Руководство FreeBSD

Аннотация

Добро пожаловать в FreeBSD! Данное руководство охватывает установку и повседневное использование *FreeBSD 15.0-RELEASE*, FreeBSD 14.4-RELEASE и 13.5-RELEASE. Эта книга стала результатом постоянной работы многих участников. Некоторые разделы могут быть устаревшими. Все желающие помочь в обновлении и расширении данного документа могут написать на электронную почту Список рассылки Проекта Документации FreeBSD.

Последняя версия данного руководства доступна на веб-сайте FreeBSD. Предыдущие версии можно получить по адресу https://docs.FreeBSD.org/doc/. Книгу можно загрузить в различных форматах и вариантах сжатия с сервера загрузок FreeBSD или одного из многочисленных зеркал. Поиск по данному руководству и другим документам доступен на странице поиска.

Содержание

Предисловие . . . 11
Целевая аудитория . . . 11
Четвертое издание . . . 11
Третье издание . . . 12
Второе издание (2004) . . . 13
Первое издание (2001) . . . 13
Организация книги . . . 14
Условные обозначения, используемые в этой книге . . . 18
Благодарности . . . 19
I: В начале . . . 20
1. Введение . . . 21
1.1. Обзор . . . 21
1.2. Добро пожаловать во FreeBSD! . . . 21
1.3. О проекте FreeBSD . . . 24
2. Установка FreeBSD . . . 29
2.1. Обзор . . . 29
2.2. Минимальные требования к оборудованию . . . 29
2.3. Задачи перед установкой . . . 29
2.4. Начало установки . . . 34
2.5. Использование bsdinstall . . . 37
2.6. Выделение дискового пространства . . . 43
2.7. Загрузка файлов дистрибутива . . . 59
2.8. Сетевые интерфейсы, учётные записи, часовой пояс, службы и защита . . . 61
2.9. Устранение неполадок . . . 85
2.10. Использование Live CD . . . 86
3. Основы FreeBSD . . . 87
3.1. Обзор . . . 87
3.2. Виртуальные консоли и терминалы . . . 87
3.3. Пользователи и базовая настройка учётных записей . . . 90
3.4. Разрешения . . . 100
3.5. Структура каталогов . . . 106
3.6. Организация диска . . . 108
3.7. Монтирование и демонтирование файловых систем . . . 115
3.8. Процессы и демоны . . . 117
3.9. Оболочки . . . 120
3.10. Текстовые редакторы . . . 124
3.11. Устройства и узлы устройств . . . 124
3.12. Справочник . . . 124

4. Установка приложений: пакеты и порты . . . 127
4.1. Обзор . . . 127
4.2. Обзор установки программного обеспечения . . . 127
4.3. Поиск программного обеспечения . . . 129
4.4. Использование pkg для управления бинарными пакетами . . . 129
4.5. Использование коллекции портов . . . 139
4.6. Сборка пакетов с poudriere . . . 148
4.7. Пост-установочные вопросы . . . 152
4.8. Работа с неработающими портами . . . 153
5. Система X Window . . . 154
5.1. Обзор . . . 154
5.2. Драйверы видеокарт . . . 154
5.3. Обзор системы X Window . . . 158
5.4. Установка сервера X.org . . . 158
5.5. Конфигурация X.org . . . 159
5.6. Использование шрифтов в системе X Window . . . 165
6. Wayland на FreeBSD . . . 171
6.1. Обзор . . . 171
6.2. Обзор Wayland . . . 171
6.3. Композитор Wayfire . . . 173
6.4. Композитор Hikari . . . 176
6.5. Композитор Sway . . . 178
6.6. Использование Xwayland . . . 179
6.7. Удалённый рабочий стол с использованием VNC . . . 182
6.8. Логин менеджер Wayland . . . 182
6.9. Полезные утилиты . . . 183
7. Сеть . . . 185
7.1. Обзор . . . 185
7.2. Настройка сети . . . 185
7.3. Проводные сети . . . 187
7.4. Беспроводные сети . . . 195
7.5. Имя сайта . . . 199
7.6. DNS . . . 200
7.7. Устранение неполадок . . . 201
II: Стандартные задачи . . . 203
8. Рабочие столы . . . 204
8.1. Обзор . . . 204
8.2. Рабочие столы . . . 204
8.3. Браузеры . . . 212
8.4. Инструменты разработки . . . 215
8.5. Настольные офисные приложения . . . 217

8.6. Просмотрщики документов . . . 219
8.7. Финансы . . . 220
9. Мультимедиа . . . 222
9.1. Обзор . . . 222
9.2. Настройка звуковой карты . . . 222
9.3. Аудиоплееры . . . 226
9.4. Видеоплееры . . . 227
9.5. Конференции и встречи . . . 230
9.6. Сканеры изображений . . . 233
10. Настройка ядра FreeBSD . . . 237
10.1. Обзор . . . 237
10.2. Зачем собирать собственное ядро? . . . 237
10.3. Поиск информации об оборудовании системы . . . 238
10.4. Файл конфигурации . . . 239
10.5. Сборка и установка собственного ядра . . . 241
10.6. Если что-то пойдет не так . . . 242
11. Печать . . . 244
11.1. Быстрый старт . . . 244
11.2. Подключение принтеров . . . 245
11.3. Языки описания страниц . . . 247
11.4. Прямая печать . . . 248
11.5. LPD (демон линейного принтера) . . . 249
11.6. Другие системы печати . . . 259
12. Двоичная совместимость с Linux . . . 260
12.1. Обзор . . . 260
12.2. Настройка бинарной совместимости с Linux . . . 260
12.3. Пользовательские окружения Linux . . . 261
12.4. Сложные темы . . . 265
13. WINE . . . 269
13.1. Обзор . . . 269
13.2. Обзор и основные концепции WINE . . . 270
13.3. Установка WINE на FreeBSD . . . 272
13.4. Запуск первой программы в WINE на FreeBSD . . . 274
13.5. Настройка установленного WINE . . . 276
13.6. Графические программы для управления WINE . . . 284
13.7. WINE в многопользовательских установках FreeBSD . . . 298
13.8. Часто задаваемые вопросы о WINE на FreeBSD . . . 300
III: Системное администрирование . . . 304
14. Конфигурация, сервисы, журналирование и управление питанием . . . 305
14.1. Обзор . . . 305
14.2. Файлы конфигурации . . . 305

14.3. Управление службами в FreeBSD . . . 310
14.4. Cron и Periodic . . . 312
14.5. Настройка системного журналирования . . . 317
14.6. Управление питанием и ресурсами . . . 327
14.7. Добавление swap-пространства . . . 333
15. Процесс загрузки FreeBSD . . . 336
15.1. Обзор . . . 336
15.2. Процесс загрузки FreeBSD . . . 336
15.3. Подсказки устройств . . . 343
15.4. Последовательность выключения . . . 344
16. Безопасность . . . 345
16.1. Обзор . . . 345
16.2. Введение . . . 345
16.3. Защита учётных записей . . . 346
16.4. Система обнаружения вторжений (IDS) . . . 352
16.5. Уровни безопасности . . . 355
16.6. Флаговые атрибуты файлов . . . 357
16.7. OpenSSH . . . 358
16.8. OpenSSL . . . 363
16.9. Kerberos . . . 368
16.10. TCP Wrappers . . . 376
16.11. Списки контроля доступа . . . 377
16.12. Capsicum . . . 379
16.13. Учёт процессов . . . 380
16.14. Ограничения ресурсов . . . 381
16.15. Мониторинг проблем безопасности приложений сторонних разработчиков . . . 384
16.16. Рекомендации по безопасности FreeBSD . . . 386
17. Клетки и контейнеры . . . 391
17.1. Обзор . . . 391
17.2. Типы клеток . . . 392
17.3. Конфигурация хоста . . . 395
17.4. Классическая клетка (Толстая клетка) . . . 404
17.5. Тонкие клетки (Thin Jails) . . . 408
17.6. Клетки VNET . . . 413
17.7. Клетки Linux . . . 417
17.8. NAT и перенаправление портов для клеток . . . 419
17.9. Сервисные клетки (Service Jails) . . . 420
17.10. Управление клетками . . . 421
17.11. Администрирование изнутри клетки . . . 428
17.12. Обновление клетки . . . 429
17.13. Ограничения ресурсов клетки . . . 433

17.14. Дополнительные параметры клетки . . . 435
17.15. Клетки и ZFS . . . 439
17.16. Менеджеры клеток и контейнеры . . . 441
18. OCI Containers . . . 442
18.1. Synopsis . . . 442
18.2. Container Image Naming and Tagging . . . 442
18.3. Official Registry Locations . . . 445
18.4. Introduction to Podman . . . 445
18.5. Preparing the System for Podman Containers . . . 445
18.6. Installing Podman . . . 446
18.7. Importing and Running Containers . . . 446
18.8. Useful Variations and Tips . . . 449
18.9. Networking and Name Resolution . . . 452
18.10. Building Custom Images . . . 452
18.11. Advanced Usage . . . 454
18.12. Building Container Images from Sources . . . 458
18.13. Cross-Architecture Testing . . . 458
18.14. Troubleshooting . . . 459
19. Принудительный контроль доступа (MAC) . . . 461
19.1. Обзор . . . 461
19.2. Ключевые термины . . . 462
19.3. Метки MAC . . . 463
19.4. Планирование конфигурации безопасности . . . 468
19.5. Доступные политики MAC . . . 469
19.6. Блокировка пользователя . . . 478
19.7. Nagios в MAC клетке . . . 479
19.8. Устранение проблем с инфраструктурой MAC . . . 482
20. Аудит событий безопасности . . . 484
20.1. Обзор . . . 484
20.2. Ключевые термины . . . 485
20.3. Настройка аудита . . . 485
20.4. Работа с журналами аудита . . . 490
21. Устройства хранения . . . 494
21.1. Обзор . . . 494
21.2. Добавление дисков . . . 494
21.3. Изменение размера и увеличение дисков . . . 495
21.4. USB-накопители . . . 498
21.5. Создание и использование CD-носителей . . . 502
21.6. Создание и использование DVD-носителей . . . 508
21.7. Создание и использование дискет . . . 514
21.8. Основы резервного копирования . . . 514

21.9. Диски в памяти . . . 519
21.10. Снимки файловой системы . . . 522
21.11. Квоты на диске . . . 523
21.12. Шифрование разделов диска . . . 527
21.13. Шифрование раздела подкачки . . . 533
21.14. Высокодоступное хранилище (HAST) . . . 535
22. GEOM: Модульная инфраструктура трансформации дискового пространства . . . 544
22.1. Обзор . . . 544
22.2. RAID0 - Striping . . . 544
22.3. RAID1 - Зеркалирование . . . 546
22.4. RAID3 - чередование на уровне байтов с выделенной чётностью . . . 557
22.5. Устройства с программным RAID . . . 558
22.6. Сетевые устройства GEOM Gate . . . 563
22.7. Маркировка дисковых устройств . . . 564
22.8. Журналирование UFS через GEOM . . . 567
23. Файловая система Z (ZFS) . . . 569
23.1. Что отличает ZFS от других . . . 569
23.2. Краткое руководство по началу работы . . . 570
23.3. Администрирование zpool . . . 577
23.4. Управление с помощью утилиты `zfs` . . . 609
23.5. Делегированное администрирование . . . 641
23.6. Встроенное шифрование ZFS . . . 642
23.7. Загрузочные окружения . . . 650
23.8. Сложные темы . . . 651
23.9. Дополнительные ресурсы . . . 656
23.10. Особенности и терминология ZFS . . . 656
24. Поддержка файловых систем . . . 669
24.1. Обзор . . . 669
24.2. Файловые системы Linux® . . . 669
24.3. Файловые системы Windows® . . . 670
24.4. Файловые системы MacOS® . . . 671
25. Виртуализация . . . 673
25.1. Обзор . . . 673
25.2. FreeBSD в качестве гостевой системы в Parallels Desktop для macOS® . . . 673
25.3. FreeBSD в качестве гостевой системы на VMware Fusion для macOS® . . . 681
25.4. FreeBSD в качестве гостевой системы в VirtualBox™ . . . 694
25.5. FreeBSD в качестве хоста с VirtualBox™ . . . 696
25.6. Виртуализация с QEMU на FreeBSD . . . 698
25.7. FreeBSD в качестве хоста с bhyve . . . 724
25.8. FreeBSD в качестве хоста Xen™ . . . 742
26. Локализация - использование и настройка i18n/L10n . . . 749

26.1. Обзор . . . 749
26.2. Использование локализации . . . 749
26.3. Поиск приложений с поддержкой i18n . . . 756
26.4. Настройка локали для определённых языков . . . 756
27. Обновление и смена версии FreeBSD . . . 760
27.1. Обзор . . . 760
27.2. Введение . . . 760
27.3. Обновление FreeBSD . . . 761
27.4. Обновление загрузочного кода . . . 770
27.5. Обновление документации . . . 770
27.6. Отслеживание ветки разработки . . . 771
27.7. Обновление FreeBSD из исходного кода . . . 775
27.8. Обновление FreeBSD пакетами . . . 782
27.9. Распространение обновлений на несколько машин . . . 789
27.10. Сборка на хостах, отличных от FreeBSD . . . 790
28. DTrace . . . 792
28.1. Обзор . . . 792
28.2. Различия в реализациях . . . 792
28.3. Включение поддержки DTrace . . . 793
28.4. Включение DTrace во внешних модулях ядра . . . 794
28.5. Использование DTrace . . . 794
29. Режим устройства USB / USB OTG . . . 798
29.1. Обзор . . . 798
29.2. Виртуальные последовательные порты USB . . . 799
29.3. Сетевые интерфейсы в режиме USB-устройства . . . 801
29.4. Виртуальное устройство хранения данных USB . . . 801
IV: Сетевые коммуникации . . . 804
30. Последовательная передача данных . . . 805
30.1. Обзор . . . 805
30.2. Терминология и оборудование для последовательной передачи данных . . . 805
30.3. Терминалы . . . 810
30.4. Входящие соединения по модему . . . 814
30.5. Исходящие соединения по модему . . . 818
30.6. Настройка последовательной консоли . . . 822
31. PPP . . . 828
31.1. Обзор . . . 828
31.2. Настройка PPP . . . 828
31.3. Устранение неполадок PPP-подключений . . . 837
31.4. Использование PPP через Ethernet (PPPoE) . . . 840
31.5. Использование PPP через ATM (PPPoA) . . . 842
32. Электронная почта . . . 846

32.1. Обзор . . . 846
32.2. Компоненты почты . . . 846
32.3. Почтовый агент DragonFly (DMA) . . . 847
32.4. Sendmail . . . 849
32.5. Изменение агента передачи почты . . . 852
32.6. Почтовые клиенты . . . 854
32.7. Сложные темы . . . 863
33. Сетевые серверы . . . 869
33.1. Обзор . . . 869
33.2. Суперсервер inetd . . . 869
33.3. Сетевая файловая система (NFS — Network File System) . . . 873
33.4. Сетевая информационная система (NIS) . . . 878
33.5. Протокол LDAP . . . 893
33.6. Протокол динамической конфигурации узла (DHCP) . . . 902
33.7. Система доменных имён (DNS) . . . 906
33.8. Сетевое взаимодействие без настройки (mDNS/DNS-SD) . . . 909
33.9. HTTP сервер Apache . . . 909
33.10. Протокол передачи файлов (FTP) . . . 917
33.11. Услуги файлов и печати для клиентов Microsoft® Windows® (Samba) . . . 918
33.12. Синхронизация времени с помощью NTP . . . 921
33.13. Настройка инициатора и цели iSCSI . . . 925
34. Межсетевые экраны . . . 931
34.1. Обзор . . . 931
34.2. Концепции межсетевого экрана . . . 932
34.3. PF . . . 934
34.4. IPFW . . . 952
34.5. IPFILTER (IPF) . . . 968
34.6. Blocklistd . . . 982
35. Сложные вопросы работы в сети . . . 988
35.1. Обзор . . . 988
35.2. Шлюзы и маршруты . . . 988
35.3. Виртуальные узлы . . . 994
35.4. Расширенная аутентификация в беспроводной сети . . . 995
35.5. Беспроводное соединение в режиме Ad-hoc . . . 1000
35.6. Раздача интернета через USB . . . 1004
35.7. Bluetooth . . . 1005
35.8. Создание моста . . . 1015
35.9. Агрегация каналов и отказоустойчивость . . . 1022
35.10. Запуск системы по сети (PXE) без использования локальных накопителей . . . 1027
35.11. Протокол общей избыточности адресов (CARP) . . . 1033
35.12. Виртуальные сети VLAN . . . 1035

V: Приложения . . . 1037
Приложение A: Получение FreeBSD . . . 1038
A.1. Зеркала . . . 1038
A.2. Используя Git . . . 1041
Приложение B: Библиография . . . 1045
B.1. Библиография FreeBSD . . . 1045
B.2. Издания о безопасности . . . 1045
B.3. История UNIX® . . . 1045
B.4. Периодические издания, журналы и газеты . . . 1046
Приложение C: Ресурсы в Интернете . . . 1047
C.1. Вебсайты . . . 1047
C.2. Списки рассылки . . . 1047
C.3. Группы новостей Usenet . . . 1050
Приложение D: Ключи OpenPGP . . . 1052
D.1. Офицеры . . . 1052
Глоссарий FreeBSD . . . 1061
Сведения об издании . . . 1082

Предисловие

Целевая аудитория

Новичок в FreeBSD обнаружит, что первая часть этой книги проведёт пользователя через процесс установки FreeBSD и мягко познакомит с концепциями и соглашениями, лежащими в основе UNIX®. Для работы с этой частью требуется лишь желание исследовать и способность усваивать новые концепции по мере их появления.

После прохождения этого этапа вторая, значительно более обширная часть Руководства представляет собой всеобъемлющий справочник по различным темам, представляющим интерес для администраторов систем FreeBSD. Некоторые из этих глав могут рекомендовать предварительное ознакомление с другими материалами, что отмечается в кратком обзоре в начале каждой главы.

Для получения дополнительных источников информации см. Библиография.

Четвертое издание

Текущая версия Руководства представляет собой совокупный результат рабочей группы, которая занимается проверкой и обновлением всего содержимого Руководства. Ниже приведены основные изменения по сравнению с четвёртым изданием Руководства.

- Справочник был преобразован из Docbook в Hugo и AsciiDoctor
- Портал документации FreeBSD был создан.
- Глава Введение была обновлена для улучшения описания истории FreeBSD и исправления незначительных опечаток.
- Глава Установка была обновлена с улучшенной аннотацией, последними изменениями в установщике, обновленными изображениями, добавленным альтернативным текстом для изображений и удалением упоминаний конкретных версий.
- Глава Основы содержит обновлённые таблицы, выводы команд и структуру каталогов в соответствии с hier(7).
- Глава Порты была обновлена и теперь упрощает поиск пакетов, обновляет примеры программного обеспечения (Nginx заменяет Apache), улучшает процесс начальной загрузки pkg(8) и добавляет новые инструкции по настройке и управлению пакетами, включая их блокировку и разблокировку.
- Глава X11 была обновлена, чтобы отразить текущее состояние графики в FreeBSD: удалены устаревшие ссылки на старые драйверы Intel, конфигурации и compiz, а инструкции по настройке окружений рабочего стола (таких как KDE Plasma и GNOME) перенесены в главу «Окружения рабочего стола», так как эти окружения теперь поддерживают не только X11, но и Wayland.
- Добавлена глава Wayland с информацией об установке и настройке Wayland в FreeBSD.
- Глава Сеть была создана для описания базовой настройки проводных и беспроводных сетей, включая имя хоста, DNS и устранение неполадок. Разделы о проводных сетях,

беспроводных сетях и IPv6 были перемещены и обновлены с улучшенными выводами команд, использованием sysrc и улучшенным синтаксисом AsciiDoc.

- Глава Рабочий стол была обновлена: добавлены улучшенные инструкции по установке KDE Plasma, GNOME, XFCE, MATE, Cinnamon и LXQT, расширены варианты браузеров, добавлен новый раздел о инструментах разработки, а также обновлены разделы о офисных приложениях, программах для просмотра документов и финансовых инструментах.
- Глава Мультимедиа была переработана с обновлениями в разделе о звуке, новыми таблицами для микшеров звука, аудиоплееров и видеоплееров, рекомендациями по автоматическому переключению на наушники, новым разделом о конференциях и встречах, а также пересмотренным разделом о сканерах изображений.
- Глава Linuxemu была улучшена за счет обновлённых инструкций по настройке базовой системы Debian/Ubuntu с использованием debootstrap.
- Глава Конфигурация была переименована для точности, с обновлениями в разделах управления службами, cron и periodic, syslog, управления питанием и подкачки. Добавлена новая запись о конфигурационных файлах, а устаревший раздел по настройке удалён.
- Глава Безопасность была обновлена с улучшениями для VPN через IPSec, защиты учётных записей, хешей паролей, sudo/doas и OpenSSH/OpenSSL. Добавлены новые разделы, посвященные IDS, уровням безопасности, флагам файлов, Capsicum, ACL NFSv4 и ограничениям ресурсов.
- Глава Клетки была обновлена и теперь включает подробности о типах клеток (толстых, тонких, VNET клеток и клеток Linux), настройке хостовой системы, вариантах сетевого взаимодействия, файле конфигурации клетки, процедурах настройки, методах обновления, ограничениях ресурсов, а также различных менеджерах клеток и решениях для контейнеров.
- Глава Почта была обновлена и теперь включает информацию о DMA, обновлениях Sendmail, инструкции по замене DMA и Sendmail на другие MTA, а также удаление разделов Dialup и Fetchmail с последующей реорганизацией главы.
- Библиография была значительно обновлена.

Третье издание

Текущая онлайн-версия Handbook представляет собой результат совместных усилий сотен участников за последние 10 лет. Ниже перечислены некоторые значительные изменения, внесённые с момента публикации третьего издания в двух томах в 2004 году:

- В документацию FreeBSD добавлен раздел WINE с информацией о запуске приложений Windows® в FreeBSD.
- В документацию FreeBSD добавлен раздел DTrace с информацией о мощном инструменте анализа производительности DTrace.
- В файле Другие файловые системы добавлена информация о не родных файловых системах в FreeBSD, таких как ZFS от Sun™.

- В документе Аудит событий безопасности добавлен раздел, посвящённый новым возможностям аудита в FreeBSD, с объяснением их использования.
- В документе FreeBSD добавлен раздел Виртуализация с информацией об установке FreeBSD на программное обеспечение для виртуализации.
- Добавлен раздел Установка FreeBSD, посвящённый установке FreeBSD с помощью новой утилиты bsdinstall.

Второе издание (2004)

Третье издание стало результатом более чем двухлетней работы преданных участников проекта FreeBSD Documentation Project. Печатное издание увеличилось до такого объёма, что потребовалось выпустить его в виде двух отдельных томов. Ниже приведены основные изменения в этом новом издании:

- Настройка и оптимизация системы была дополнена новой информацией об управлении питанием и ресурсами ACPI, системной утилите `cron`, а также дополнительными параметрами настройки ядра.
- Безопасность дополнена новой информацией о виртуальных частных сетях (VPN), списках контроля доступа (ACL) файловых систем и рекомендациях по безопасности.
- Принудительный контроль доступа — это новая глава в данном издании. В ней объясняется, что такое MAC и как этот механизм может быть использован для защиты системы FreeBSD.
- Хранение данных была дополнена новой информацией о USB-накопителях, снимках файловых систем, квотах файловых систем, файловых и сетевых файловых системах, а также о зашифрованных разделах диска.
- В раздел PPP добавлен подраздел по устранению неполадок.
- Электронная почта была дополнена новой информацией об использовании альтернативных транспортных агентов, SMTP-аутентификации, UUCP, fetchmail, procmail и других расширенных темах.
- Сетевые серверы полностью обновлена в этом издании. Эта глава включает информацию о настройке Apache HTTP Server, ftpd, а также о настройке сервера для клиентов Microsoft® Windows® с помощью Samba. Некоторые разделы из Расширенные сетевые технологии были перемещены сюда для улучшения изложения.
- Расширенные сетевые технологии дополнены новой информацией об использовании устройств Bluetooth® с FreeBSD, настройке беспроводных сетей и работе с сетями Asynchronous Transfer Mode (ATM).
- В книгу добавлен глоссарий, который служит центральным местом для определений технических терминов, используемых в тексте.
- В книгу внесён ряд визуальных улучшений в таблицы и иллюстрации.

Первое издание (2001)

Второе издание стало результатом более двух лет работы преданных членов Проекта

документации FreeBSD. Основные изменения в этом издании включали:

- Добавлен полный указатель.
- Все ASCII-рисунки заменены графическими диаграммами.
- В начало каждой главы добавлено стандартное краткое содержание, которое даёт общее представление о том, какую информацию содержит глава и какие знания ожидаются от читателя.
- Содержание было логически реорганизовано в три части: «Начало работы», «Администрирование системы» и «Приложения».
- Раздел Основы FreeBSD был расширен, и теперь включает дополнительную информацию о процессах, демонах и сигналах.
- Раздел Установка приложений: Пакеты и порты был дополнен дополнительной информацией об управлении бинарными пакетами.
- Раздел Система X Window был полностью переписан с акцентом на использование современных технологий для рабочих столов, таких как KDE и GNOME на XFree86™ 4.X.
- Раздел Процесс загрузки FreeBSD был расширен.
- Раздел Хранение данных был написан на основе двух глав: «Диски» и «Резервное копирование», которые ранее были отдельными. Мы считаем, что объединение этих тем в одну главу облегчает их понимание. Также добавлен раздел о RAID (как аппаратном, так и программном).
- Раздел Последовательные коммуникации был полностью переработан и обновлен для FreeBSD 4.X/5.X.
- Раздел PPP был значительно обновлен.
- В новую секцию Сложные вопросы работы в сети добавлено множество новых разделов.
- Раздел Электронная почта был дополнен дополнительной информацией о настройке sendmail.
- Раздел Совместимость с бинарными файлами Linux® был расширен и включает информацию об установке Oracle® и SAP® R/3®.
- В этом втором издании рассматриваются следующие новые темы:
  - Настройка и конфигурирование.
  - Мультимедиа.

Организация книги

Эта книга разделена на пять логически обособленных частей. Первая часть, *Начало работы*, посвящена установке и основам использования FreeBSD. Предполагается, что читатель будет изучать эти главы последовательно, возможно, пропуская главы на знакомые темы. Вторая часть, *Повседневные задачи*, охватывает часто используемые возможности FreeBSD. Эту часть, как и все последующие, можно читать в любом порядке. Каждая глава начинается с краткого описания, в котором указано, что рассматривается в главе и какие знания уже должны быть у читателя. Это позволяет бегло просматривать содержание и

находить интересующие главы. Третья часть, *Администрирование системы*, посвящена вопросам администрирования. Четвёртая часть, *Сетевое взаимодействие*, охватывает темы, связанные с сетями и серверами. Пятая часть содержит справочные приложения.

Введение

Представляет FreeBSD новому пользователю. Описывает историю проекта FreeBSD, его цели и модель разработки.

Установка FreeBSD

Проводит пользователя через весь процесс установки FreeBSD 9.*x* и более поздних версий с использованием bsdinstall.

Основы FreeBSD

Охватывает основные команды и функциональность операционной системы FreeBSD. Если вы знакомы с Linux® или другой разновидностью UNIX®, то, вероятно, можете пропустить эту главу.

Установка приложений: Пакеты и порты

Охватывает установку стороннего программного обеспечения с использованием инновационной "Коллекции портов" FreeBSD и стандартных бинарных пакетов.

Система X Window

Описывает X Window System в общем и использование X11 в FreeBSD в частности. Также рассматриваются популярные окружения рабочего стола, такие как KDE и GNOME.

Wayland

Описывает Wayland, сервер отображения в целом, и использование Wayland в FreeBSD в частности. Также рассматриваются популярные композиторы, такие как Wayfire, Hikari и Sway.

Приложения для рабочего стола

Перечисляет некоторые распространённые настольные приложения, такие как веб-браузеры и офисные пакеты, и описывает, как их установить в FreeBSD.

Мультимедиа

Показывает, как настроить поддержку воспроизведения звука и видео в системе. Также описываются некоторые примеры аудио- и видеоприложений.

Настройка ядра FreeBSD

Объясняет, зачем может потребоваться настройка нового ядра, и предоставляет подробные инструкции по настройке, сборке и установке кастомного ядра.

Печать

Описывает управление принтерами в FreeBSD, включая информацию о титульных страницах, учёте печати и первоначальной настройке.

Совместимость с бинарными файлами Linux®

Описывает возможности совместимости FreeBSD с Linux®. Также содержит подробные

инструкции по установке многих популярных приложений Linux®, таких как Oracle® и Mathematica®.

WINE

Описывает WINE и предоставляет подробные инструкции по установке. Также описывает, как работает WINE, как установить графический помощник, как запускать приложения Windows® на FreeBSD, а также предлагает другие советы и решения.

Настройка и Тонкая Настройка

Описывает параметры, доступные системным администраторам для настройки системы FreeBSD с целью достижения оптимальной производительности. Также описываются различные конфигурационные файлы, используемые в FreeBSD, и их расположение.

Процесс загрузки FreeBSD

Описывает процесс загрузки FreeBSD и объясняет, как управлять этим процессом с помощью параметров конфигурации.

Безопасность

Описывает множество различных инструментов, доступных для обеспечения безопасности системы FreeBSD, включая Kerberos, IPsec и OpenSSH.

Клетки

Описывает фреймворк клеток и улучшения по сравнению с традиционной поддержкой chroot в FreeBSD.

Принудительный контроль доступа

Объясняет, что такое Принудительный Контроль Доступа (Mandatory Access Control, MAC) и как этот механизм может быть использован для защиты системы FreeBSD.

Аудит событий безопасности

Описывает, что такое аудит событий в FreeBSD, как его можно установить, настроить, а также как проверять или отслеживать журналы аудита.

Устройства хранения

Описывает, как управлять носителями данных и файловыми системами в FreeBSD. Это включает физические диски, RAID-массивы, оптические и ленточные носители, диски в памяти и сетевые файловые системы.

GEOM: Модульная инфраструктура преобразования дисковых запросов

Описывает, что такое фреймворк GEOM в FreeBSD и как настроить различные поддерживаемые уровни RAID.

Платформа хранения данных OpenZFS

Описывает платформу хранения данных OpenZFS и предоставляет краткое руководство по началу работы, а также информацию о продвинутых темах, связанных с использованием OpenZFS в FreeBSD.

Другие файловые системы

Изучает поддержку неродных файловых систем в FreeBSD, таких как ext2, ext3 и ext4.

Виртуализация

Описывает, какие системы виртуализации доступны и как их можно использовать с FreeBSD.

Локализация - использование и настройка i18n/L10n

Описывает, как использовать FreeBSD на языках, отличных от английского. Рассматривает локализацию как на уровне системы, так и на уровне приложений.

Обновление и модернизация FreeBSD

Объясняет различия между FreeBSD-STABLE, FreeBSD-CURRENT и выпусками FreeBSD. Описывает, какие пользователи могут извлечь выгоду из отслеживания разработки системы, и излагает этот процесс. Рассматривает методы, которые пользователи могут использовать для обновления своей системы до последнего безопасного выпуска.

DTrace

Описывает, как настроить и использовать инструмент DTrace от Sun™ в FreeBSD. Динамическая трассировка помогает выявлять проблемы с производительностью, выполняя анализ системы в реальном времени.

Режим USB-устройства / USB OTG

Объясняет использование режима USB0-устройства и USB On The Go (USB OTG) в FreeBSD.

PPP

Описывает, как использовать PPP для подключения к удалённым системам в FreeBSD.

Электронная Почта

Объясняет различные компоненты почтового сервера и рассматривает простые вопросы настройки для наиболее популярного почтового сервера: sendmail.

Сетевые серверы

Предоставляет подробные инструкции и примеры конфигурационных файлов для настройки вашей FreeBSD-машины в качестве сервера сетевых файловых систем, сервера доменных имен, сервера сетевой информационной системы или сервера синхронизации времени.

Межсетевые экраны

Объясняет философию программных межсетевых экранов и предоставляет подробную информацию о настройке различных межсетевых экранов, доступных для FreeBSD.

Расширенные сетевые технологии

Описывает множество тем, связанных с сетями, включая совместное использование интернет-подключения с другими компьютерами в локальной сети, расширенные темы маршрутизации, беспроводные сети, Bluetooth®, ATM, IPv6 и многое другое.

Получение FreeBSD

Перечисляет различные источники для получения FreeBSD на CDROM или DVD, а также различные сайты в Интернете, которые позволяют загрузить и установить FreeBSD.

Библиография

Эта книга затрагивает множество различных тем, которые могут пробудить в читателях желание узнать больше. В библиографии приведён список отличных книг, на которые есть ссылки в тексте.

Ресурсы в Интернете

Описывает множество форумов, доступных для пользователей FreeBSD, где можно задавать вопросы и участвовать в технических обсуждениях, связанных с FreeBSD.

OpenPGP Ключи

Перечисляет PGP-отпечатки нескольких разработчиков FreeBSD.

Условные обозначения, используемые в этой книге

Для обеспечения единообразия и удобочитаемости текста в книге соблюдаются определённые соглашения.

Типографические соглашения

Курсив

Курсивный шрифт используется для имён файлов, URL-адресов, выделенного текста и первого упоминания технических терминов.

Моноширинный

Моноширинный шрифт используется для сообщений об ошибках, команд, переменных окружения, названий портов, имён хостов, имён пользователей, имён групп, имён устройств, переменных и фрагментов кода.

Жирный

Жирный шрифт используется для приложений, команд и клавиш.

Ввод пользователя

Ключи выделены **жирным шрифтом**, чтобы отличаться от остального текста. Комбинации клавиш, которые нужно нажимать одновременно, обозначаются символом + между клавишами, например:

`Ctrl` + `Alt` + `Del`

Пользователь должен одновременно нажать клавиши `Ctrl`, `Alt` и `Del`.

Клавиши, которые нужно нажимать последовательно, разделяются запятыми, например:

`Ctrl` + `X`, `Ctrl` + `S`

Это означает, что пользователь должен одновременно нажать клавиши `Ctrl` и `X`, а затем одновременно нажать клавиши `Ctrl` и `S`.

Примеры

Примеры, начинающиеся с C:\>, обозначают команду MS-DOS®. Если не указано иное, эти команды могут быть выполнены в окне "Командная строка" в современной среде Microsoft® Windows®.

```
C:\> tools\fdimage floppies\kern.flp A:
```

Примеры, начинающиеся с символа #, обозначают команды, которые должны выполняться с правами суперпользователя в FreeBSD. Вы можете войти в систему как `root` для выполнения команды или использовать обычную учётную запись и применить su(1) для получения прав суперпользователя.

```
# dd if=kern.flp of=/dev/fd0
```

Примеры, начинающиеся с символа %, обозначают команды, которые должны выполняться от имени обычной учётной записи. Если не указано иное, для установки переменных окружения и других команд оболочки используется синтаксис C-shell.

```
% top
```

Благодарности

Эта книга — результат труда многих сотен людей по всему миру. Будь то исправления опечаток или целые главы, все их вклады оказались полезными.

Несколько компаний поддержали разработку этого документа, оплачивая работу авторов на полную ставку, финансируя публикацию и т.д. В частности, BSDi (позже приобретенная Wind River Systems) оплачивала членам FreeBSD Documentation Project работу над улучшением этой книги на полной ставке вплоть до публикации первого печатного издания в марте 2000 года (ISBN 1-57176-241-8). Затем Wind River Systems оплатила работу нескольких дополнительных авторов для внесения ряда улучшений в инфраструктуру печатного вывода и добавления новых глав к тексту. Эта работа завершилась публикацией второго печатного издания в ноябре 2001 года (ISBN 1-57176-303-1). В 2003-2004 годах FreeBSD Mall, Inc оплатила работу нескольких участников над улучшением Руководства в рамках подготовки к третьему печатному изданию. Третье печатное издание было разделено на два тома. Оба тома были опубликованы как **The FreeBSD Handbook 3rd Edition Volume 1: User Guide** (ISBN 1-57176-327-9) и **The FreeBSD Handbook 3rd Edition Volume 2: Administrators Guide** (ISBN 1-57176-328-7).

Часть I: В начале

Эта часть руководства предназначена для пользователей и администраторов, которые только начинают работать с FreeBSD. В следующих главах:

- Введение в FreeBSD.
- Пошаговое руководство по установке системы.
- Основы работы с UNIX® для начинающих.
- Установка стороннего программного обеспечения из коллекции портов.
- Знакомство с системой X Window, настройка графического окружения для повышения эффективности работы.
- Введение в Wayland, новый сервер экрана для UNIX®.

Количество ссылок в тексте на следующие главы сведено к минимуму, чтобы данный раздел можно было последовательно прочитать от начала до конца без необходимости постоянного перелистывания страниц.

Глава 1. Введение

1.1. Обзор

Спасибо за ваш интерес к FreeBSD! Следующая глава охватывает различные аспекты проекта FreeBSD, такие как его история, цели, модель разработки и так далее.

Прочтите эту главу, чтобы узнать:

- Как FreeBSD соотносится с другими операционными системами.
- История проекта FreeBSD.
- Цели проекта FreeBSD.
- Основы модели разработки открытого исходного кода FreeBSD.
- Откуда произошло название «FreeBSD».

1.2. Добро пожаловать во FreeBSD!

FreeBSD — это операционная система с открытым исходным кодом, соответствующая стандартам и похожая на Unix, предназначенная для компьютеров на архитектурах x86 (как 32-, так и 64-битных), ARM, AArch64, RISC-V, POWER и PowerPC. Она предоставляет все функции, которые сегодня считаются само собой разумеющимися, такие как вытесняющая многозадачность, защита памяти, виртуальная память, многопользовательские возможности, поддержка SMP, все инструменты разработки с открытым исходным кодом для различных языков и фреймворков, а также возможности для рабочих станций, включая X Window System, KDE или GNOME. Её основные преимущества:

- *Лицензия Open Source с либеральными условиями*, которая предоставляет вам права свободно изменять и расширять её исходный код, а также включать его как в проекты с открытым исходным кодом, так и в закрытые продукты, не накладывая ограничений, характерных для копилефтных лицензий (copyleft — требование той же лицензии для производных продуктов), и избегая потенциальных проблем несовместимости лицензий.
- *Мощные TCP/IP сети* — FreeBSD реализует промышленные стандартные протоколы с постоянно растущей производительностью и масштабируемостью. Это делает её отличным выбором как для серверов, так и для маршрутизации/файрволинга — и действительно, многие компании и поставщики используют её именно для этих целей.
- *Полностью интегрированная поддержка OpenZFS*, включая root-on-ZFS, ZFS Boot Environments, управление отказами, делегирование административных задач, поддержку клеток (jail), документацию, специфичную для FreeBSD, и поддержку системного установщика.
- *Расширенные функции безопасности,* от системы принудительного контроля доступа до механизмов песочницы и возможностей Capsicum.
- *Более 30 тысяч предварительно собранных пакетов* для всех поддерживаемых архитектур, а также Коллекция портов, которая позволяет легко создавать собственные

настраиваемые пакеты.

- *Документация* - в дополнение к Руководству и книгам различных авторов, охватывающим темы от системного администрирования до внутреннего устройства ядра, существуют также man(1) страницы, доступные не только для пользовательских демонов, утилит и конфигурационных файлов, но и для API драйверов ядра (раздел 9) и отдельных драйверов (раздел 4).
- *Простая и целостная структура хранилища исходного кода и система сборки* — FreeBSD использует единый репозиторий для всех своих компонентов, как относящихся к ядру, так и пользовательскому окружению. Вместе с унифицированной, легко настраиваемой системой сборки и хорошо продуманным процессом разработки это позволяет легко интегрировать FreeBSD в инфраструктуру сборки решений сторонних разработчиков или производных продуктов.
- *Верность философии Unix*, предпочитая модульность вместо монолитных "всё в одном" демонов с жёстко заданным поведением.
- *Двоичная совместимость* с Linux, позволяющая запускать многие Linux-приложения без необходимости виртуализации.

FreeBSD основана на релизе 4.4BSD-Lite от Computer Systems Research Group (CSRG) Калифорнийского университета в Беркли и продолжает славные традиции разработки систем BSD. Помимо отличной работы, проделанной CSRG, проект FreeBSD вложил многие тысячи человеко-часов в расширение функциональности и тонкую настройку системы для достижения максимальной производительности и надёжности в условиях реальных нагрузок. FreeBSD предлагает производительность и надёжность, сопоставимые с другими решениями с открытым исходным кодом и коммерческими предложениями, в сочетании с передовыми функциями, недоступными больше нигде.

1.2.1. Что может FreeBSD?

Применение FreeBSD действительно ограничено только собственной фантазией. От разработки программного обеспечения до автоматизации производства, от управления запасами до азимутальной коррекции удалённых спутниковых антенн — если что-то можно сделать с коммерческим продуктом UNIX®, то, скорее всего, это можно сделать и с FreeBSD! FreeBSD также значительно выигрывает от тысяч высококачественных приложений, разработанных исследовательскими центрами и университетами по всему миру, которые часто доступны по очень низкой цене или вообще бесплатно.

Поскольку исходный код FreeBSD также свободно доступен, система может быть адаптирована для специальных приложений или проектов в практически беспрецедентной степени и способами, которые обычно недоступны в операционных системах от большинства крупных коммерческих поставщиков. Вот лишь несколько примеров областей, в которых люди в настоящее время используют FreeBSD:

- *Интернет-сервисы:* Мощный стек TCP/IP, встроенный в FreeBSD, делает её идеальной платформой для различных интернет-сервисов, таких как:
  - Веб-серверы
  - IPv4 и IPv6 маршрутизация

    - Межсетевые экраны и шлюзы NAT («маскарадинг IP»)
    - Серверы FTP
    - Почтовые серверы
    - Серверы хранения данных
    - Серверы виртуализации
    - И ещё…
- *Образование:* Студенты, изучающие информационные технологии или смежную инженерную специальность, получат выгоду от FreeBSD. Нет лучшего способа познать операционные системы, компьютерную архитектуру и сети, чем получить практический опыт работы с FreeBSD, изучив её внутреннее устройство. Множество свободно доступных САПР, математических и графических пакетов также делают FreeBSD крайне полезной для тех, кто в первую очередь использует компьютер для решения *других* задач!
- *Исследования:* Благодаря доступности исходного кода всей системы FreeBSD представляет собой отличную платформу для исследований в области операционных систем, а также других разделов компьютерных наук. Свободная доступность FreeBSD также позволяет удалённым группам сотрудничать в разработке идей или совместных проектов, не беспокоясь о специальных лицензионных соглашениях или ограничениях на обсуждение в открытых форумах.
- *Сеть:* Нужен новый маршрутизатор? Сервер имён (DNS)? Межсетевой экран, чтобы защитить вашу внутреннюю сеть от несанкционированного доступа? FreeBSD может легко превратить неиспользуемый компьютер, пылящийся в углу, в продвинутый маршрутизатор с возможностями сложной фильтрации пакетов.
- *Встроенные системы:* FreeBSD представляет собой отличную платформу для создания встроенных систем. С поддержкой архитектур ARM, AArch64 и PowerPC, в сочетании с надёжным сетевым стеком, передовыми функциями и разрешительной лицензией BSD, FreeBSD служит отличной основой для построения встроенных маршрутизаторов, межсетевых экранов и других устройств.
- *Рабочий стол:* FreeBSD представляет собой отличный выбор в качестве недорогого решения для рабочего стола с использованием свободно доступных серверов X11 и Wayland. FreeBSD предлагает на выбор множество открытых окружений рабочего стола, включая стандартные графические интерфейсы GNOME и KDE. FreeBSD даже может загружаться «без диска» с центрального сервера, что делает отдельные рабочие станции ещё дешевле и проще в администрировании.
- *Разработка ПО:* Базовая система FreeBSD включает полный набор инструментов для разработки, в том числе компиляторы C/C++ и отладчики. Поддержка многих других языков программирования также доступна через коллекции портов и пакетов.

FreeBSD доступна для бесплатной загрузки образов или установки в большинстве облачных сервисов. Дополнительную информацию о получении FreeBSD см. в Получение FreeBSD.

1.2.2. Кто использует FreeBSD?

FreeBSD известна своими возможностями веб-сервера. Список отзывов компаний, которые

используют FreeBSD в своих продуктах и услугах, можно найти на сайте Фонда FreeBSD. Wikipedia также ведёт список продуктов, основанных на FreeBSD.

1.3. О проекте FreeBSD

Следующий раздел содержит справочную информацию о проекте, включая краткую историю, цели проекта и модель разработки проекта.

1.3.1. Краткая история FreeBSD

Проект FreeBSD зародился в начале 1993 года, отчасти как детище последних трёх координаторов Неофициальных наборов патчей 386BSD: Нейта Уильямса, Рода Граймса и Джордана Хаббарда.

Изначальной целью было создание промежуточного снимка состояния 386BSD для исправления ряда проблем, которые не могли быть решены с помощью механизма наборов патчей. В связи с этим ранним рабочим названием проекта было 386BSD 0.5 или 386BSD Interim.

386BSD была операционной системой Билла Джолица, которая на тот момент сильно страдала от почти годичного отсутствия внимания. Поскольку набор патчей с каждым днём становился всё более громоздким, они решили помочь Биллу, выпустив этот промежуточный «очищенный» снимок состояния. Однако эти планы внезапно рухнули, когда Билл Джолиц неожиданно отозвал свою поддержку проекта, не предложив никаких ясных альтернатив.

Трое сочли, что цель по-прежнему стоит усилий, даже без поддержки Билла, и поэтому они дали проекту название «FreeBSD», предложенное Дэвидом Гринманом. Первоначальные задачи были определены после консультаций с текущими пользователями системы, и, когда стало ясно, что проект, возможно, станет реальностью, Джордан связался с Walnut Creek CDROM, стараясь найти каналы распространения FreeBSD тем, у кого не было лёгкого доступа к интернету. Walnut Creek CDROM не только поддержали идею распространения FreeBSD на CD, но и предоставили проекту машину для работы и быстрый интернет-канал. Без почти беспрецедентной веры Walnut Creek CDROM в тогда ещё совершенно неизвестный проект, вряд ли FreeBSD так быстро достигла бы таких высот, на которых она находится сегодня.

Первый релиз FreeBSD на CD-ROM (и в целом в сети) был FreeBSD 1.0, выпущенный в декабре 1993 года. Он основывался на ленте 4.3BSD-Lite ("Net/2") от U.C. Berkeley, с множеством компонентов, также предоставленных 386BSD и Free Software Foundation. Это был довольно успешный первый релиз, за которым последовал весьма успешный FreeBSD 1.1, выпущенный в мае 1994 года.

Примерно в это же время на горизонте сгустились неожиданные тучи, когда Novell и Калифорнийский университет в Беркли урегулировали свой затянувшийся судебный спор о юридическом статусе ленты Berkeley Net/2. По условиям соглашения Калифорнийский университет признал, что три файла из Net/2 содержали «обременённый» код и должны быть удалены, так как являлись собственностью Novell, которая, в свою очередь, ранее приобрела их у AT&T. В обмен Беркли получил «благословение» Novell на то, что выпуск

4.4BSD-Lite, когда он наконец выйдет, будет объявлен свободным от ограничений, а все существующие пользователи Net/2 будут настоятельно рекомендованы к переходу на него. Это касалось и FreeBSD, и проекту дали срок до конца июля 1994 года, чтобы прекратить распространение своей версии на основе Net/2. Согласно условиям соглашения, проекту разрешили сделать один последний выпуск до истечения срока — им стал FreeBSD 1.1.5.1.

Затем FreeBSD приступила к сложной задаче буквально переизобретения себя на основе совершенно нового и довольно неполного набора компонентов 4.4BSD-Lite. Хотя были удалены только три файла, связанных с разделяемой памятью и семафорами System V, в дистрибутив BSD было внесено множество других изменений и исправлений ошибок, поэтому объединение всех наработок FreeBSD с 4.4BSD-Lite оказалось огромной работой. Проекту потребовалось время до ноября 1994 года, чтобы осуществить этот переход, и в декабре он выпустил FreeBSD 2.0 для всего мира. Несмотря на то, что релиз всё ещё был сыроват, он оказался весьма успешным, а в июне 1995 года последовал более стабильный и простой в установке выпуск FreeBSD 2.0.5.

С тех пор FreeBSD выпустила серию релизов, каждый из которых улучшал стабильность, скорость и функциональность предыдущей версии.

На данный момент долгосрочные проекты разработки продолжают вестись в ветке 16.0-CURRENT (main), а снимки состояния 16.0 регулярно публикуются на сервере снимков по мере продвижения работы.

1.3.2. Цели проекта FreeBSD

Цели проекта FreeBSD заключаются в предоставлении программного обеспечения, которое может использоваться для любых целей без каких-либо ограничений. Многие из нас вложили значительные усилия в разработку кода (и проекта) и, конечно, не отказались бы от небольшого финансового вознаграждения время от времени, но мы точно не готовы настаивать на этом. Мы считаем, что наша главная «миссия» — предоставлять код всем желающим, независимо от целей, чтобы код получил максимально широкое распространение и принес максимальную пользу. Это, по нашему мнению, одна из фундаментальных целей свободного программного обеспечения, которую мы горячо поддерживаем.

В нашем исходном коде программное обеспечение, распространяемое под лицензией GNU General Public License (GPL) или Library General Public License (LGPL), имеет несколько больше ограничений, хотя, по крайней мере, в сторону обеспечения доступа, а не наоборот. Однако из-за дополнительных сложностей, которые могут возникнуть при коммерческом использовании ПО под GPL, мы предпочитаем программное обеспечение, предоставленное под более свободной лицензией BSD, когда это разумный вариант.

1.3.3. Модель разработки FreeBSD

Разработка FreeBSD — это очень открытый и гибкий процесс, буквально созданный из вклада тысяч людей по всему миру, что можно увидеть в нашем списке участников. Инфраструктура разработки FreeBSD позволяет этим тысячам участников сотрудничать через Интернет. Мы постоянно ищем новых добровольцев, и тем, кто хочет более тесно вовлечься, стоит ознакомиться со статьёй о вкладе в FreeBSD.

Полезная информация о проекте FreeBSD и процессе его разработки, независимо от того, работаете ли вы самостоятельно или в тесном сотрудничестве:

Репозитории Git

В течение нескольких лет центральное дерево исходных кодов FreeBSD поддерживалось с помощью CVS (Concurrent Versions System), свободно распространяемой системы контроля версий. В июне 2008 года проект перешёл на использование SVN (Subversion). Переход был признан необходимым, так как технические ограничения CVS становились всё более очевидными из-за быстрого роста дерева исходных кодов и объёма уже накопленной истории. Репозитории проекта документации и коллекции портов также были перенесены с CVS на SVN в мае 2012 года и июле 2012 года соответственно. В декабре 2020 года проект перенёс репозитории исходных кодов и документации на Git, а коллекция портов последовала их примеру в апреле 2021 года. Дополнительную информацию о получении репозитория FreeBSD `src/` можно найти в разделе Получение исходных кодов, а подробности о получении коллекции портов FreeBSD — в разделе Использование коллекции портов.

Список коммиттеров

Коммиттеры — это люди, у которых есть доступ на *запись* в Git-репозиторий и которые уполномочены вносить изменения в исходный код FreeBSD (термин «коммиттер» происходит от команды `commit`, используемой в системе контроля версий для добавления новых изменений в репозиторий). Любой может отправить отчёт об ошибке в link:https://bugs.FreeBSD.org/submit/ [Базу данных ошибок]. Перед отправкой отчёта можно воспользоваться списками рассылки FreeBSD, IRC-каналами или форумами, чтобы убедиться, что проблема действительно является ошибкой.

Команда разработчиков FreeBSD

Команда core FreeBSD была бы эквивалентна совету директоров, если бы проект FreeBSD был компанией. Основная задача команды core — убедиться, что проект в целом находится в хорошем состоянии и движется в правильном направлении. Приглашение преданных и ответственных разработчиков в нашу группу коммиттеров — одна из функций команды core, как и привлечение новых членов команды core по мере того, как другие уходят. Текущая команда core была избрана из числа кандидатов-коммиттеров в мае и июне 2024 года. Выборы проводятся каждые 2 года.

Как и большинство разработчиков, большинство членов основной команды также являются добровольцами в разработке FreeBSD и не получают финансовой выгоды от проекта, поэтому «обязательство» не следует ошибочно истолковывать как «гарантированную поддержку». Приведённая выше аналогия с «советом директоров» не совсем точна, и, возможно, правильнее будет сказать, что это люди, которые вопреки здравому смыслу посвятили свою жизнь FreeBSD!

Фонд FreeBSD

Фонд FreeBSD (FreeBSD Foundation) — это некоммерческая организация 501(c)(3), базирующаяся в США, которая занимается поддержкой и продвижением проекта FreeBSD и его сообщества по всему миру. Фонд финансирует разработку программного

обеспечения через гранты на проекты и предоставляет сотрудников для оперативного решения срочных проблем, а также внедрения новых функций и возможностей. Фонд закупает оборудование для улучшения и поддержки инфраструктуры FreeBSD, а также финансирует работу специалистов для повышения тестового покрытия, непрерывной интеграции и автоматизации. Фонд продвигает FreeBSD, участвуя в технических конференциях и мероприятиях по всему миру. Кроме того, Фонд проводит мастер-классы, разрабатывает учебные материалы и презентации для привлечения новых пользователей и разработчиков в проект FreeBSD. Фонд также представляет проект FreeBSD при заключении контрактов, лицензионных соглашений и других юридических договоров, требующих участия официального юридического лица.

Внешние участники

И последнее, но не менее важное: самая большая группа разработчиков — это сами пользователи, которые практически постоянно предоставляют нам обратную связь и исправления ошибок. Основной способ следить за разработкой базовой системы FreeBSD — подписаться на список рассылки Список рассылки FreeBSD, посвящённый техническим обсуждениям, где обсуждаются подобные вопросы. Для портирования сторонних приложений используется список Список рассылки, посвящённый Портам FreeBSD. Для документации — Список рассылки Проекта Документации FreeBSD. Дополнительную информацию о различных списках рассылки FreeBSD можно найти в разделе Ресурсы в Интернете.

Список участников FreeBSD длинный и постоянно растёт, так почему бы не присоединиться к нему, внеся свой вклад в FreeBSD, уже сегодня? Предоставление кода — не единственный способ!

Вкратце, наша модель разработки организована как свободный набор концентрических кругов. Централизованная модель создана для удобства *пользователей* FreeBSD, которым предоставляется простой способ отслеживания единой централизованной кодовой базы, а не для того, чтобы исключить потенциальных участников! Наше стремление — предоставить стабильную операционную систему с обширным набором согласованных прикладных программ, которые пользователи могут легко устанавливать и использовать — данная модель отлично справляется с этой задачей.

Мы просим от тех, кто хочет присоединиться к нам в качестве разработчиков FreeBSD, лишь такой же преданности, которую проявляют нынешние участники проекта для его дальнейшего успеха!

1.3.4. Сторонние программы

Помимо базовых дистрибутивов, FreeBSD предлагает коллекцию портированного программного обеспечения, включающую тысячи популярных программ. Список портов варьируется от HTTP-серверов до игр, языков программирования, редакторов и почти всего, что между ними. Насчитывается около 37000 портов; вся Коллекция портов занимает примерно 3 GB. Чтобы скомпилировать порт, достаточно перейти в каталог программы, которую вы хотите установить, ввести `make install`, а система сделает всё остальное. Полный исходный дистрибутивный файл для каждого порта загружается динамически, поэтому требуется достаточно места на диске для сборки только нужного порта.

Почти каждый порт также доступен в виде предварительно скомпилированного «пакета», который можно установить простой командой (`pkg install`) для тех, кто не хочет компилировать порты из исходного кода. Дополнительная информация о пакетах и портах доступна в Установка приложений: Пакеты и Порты.

1.3.5. Дополнительная документация

Все поддерживаемые версии FreeBSD предоставляют возможность в установщике установить дополнительную документацию в /usr/local/share/doc/freebsd во время первоначальной настройки системы. Документация также может быть установлена позже с использованием пакетов:

```
# pkg install en-freebsd-doc
```

Для локализованных версий замените «en» на префикс нужного языка. Учтите, что некоторые локализованные версии могут быть устаревшими и содержать информацию, которая больше не является корректной или актуальной. Читайте локально установленные руководства в веб-браузере, используя следующие URL-адреса:

Руководство FreeBSD

`/usr/local/share/doc/freebsd/en/books/handbook/handbook_en.pdf`

Часто задаваемые вопросы по FreeBSD (FAQ)

`/usr/local/share/doc/freebsd/en/books/faq/faq_en.pdf`

Актуальную документацию всегда можно найти на Портале документации.

Все товарные знаки являются собственностью их законных владельцев.

Глава 2. Установка FreeBSD

2.1. Обзор

FreeBSD поддерживает различные архитектуры, включая amd64, ARM®, RISC-V® и PowerPC®. В зависимости от архитектуры и платформы, различные образы могут быть скачены для установки или непосредственного запуска FreeBSD.

Типы образов:

- Диски образов виртуальных машин, такие как `qcow2`, `vmdk`, `vhd`, и образы raw-устройств. Это не установочные образы, а образы с предустановленной FreeBSD, готовые к выполнению задач после установки. Образы виртуальных машин также часто используются в облачных средах.
- Образы SD-карт для встраиваемых систем, таких как Raspberry Pi. Эти файлы необходимо распаковать и записать как сырой образ на SD-карту, с которой будет загружаться плата.
- Установочные образы для загрузки с ISO или USB-устройства, чтобы установить FreeBSD на диск для обычной настольной, портативной или серверной системы.

Остальная часть этой главы описывает третий случай, объясняя, как установить FreeBSD с использованием текстовой программы установки под названием `bsdinstall`. Между установщиком и тем, что показано здесь, могут быть незначительные различия, поэтому используйте эту главу как общее руководство, а не как точную инструкцию.

Прочтите эту главу, чтобы узнать:

- Как получить образы FreeBSD и создать установочные носители FreeBSD.
- Как запустить bsdinstall.
- Вопросы, которые задаст bsdinstall, их значение и как на них отвечать.
- Как решить проблемы неудачной установки.
- Как получить доступ к свежей версии FreeBSD перед установкой системы.

2.2. Минимальные требования к оборудованию

Требования к оборудованию для установки FreeBSD зависят от архитектуры и версии. Поддерживаемые аппаратные архитектуры и устройства для выпуска FreeBSD перечислены на странице FreeBSD Release Information. На странице FreeBSD download page также приведены рекомендации по выбору правильного образа для различных архитектур.

2.3. Задачи перед установкой

После того как будет подтверждено, что система соответствует минимальным требованиям для установки FreeBSD, необходимо загрузить установочный файл и подготовить установочный носитель.

Рассмотрите возможность использования Виртуализации, если вы хотите использовать FreeBSD на системе, где уже установлена другая операционная система.

Прежде чем перейти к установке, убедитесь, что система готова, проверив пункты из этого контрольного списка:

1. **Резервное копирование важных данных**

   Перед установкой любой операционной системы **всегда** создавайте резервную копию всех важных данных. Не храните резервную копию на системе, на которую производится установка. Вместо этого сохраните данные на съемный диск, такой как USB-накопитель, другую систему в сети или в онлайн-сервис резервного копирования. Проверьте резервную копию перед началом установки, чтобы убедиться, что она содержит все необходимые файлы. После того как установщик отформатирует диск системы, все данные, хранящиеся на этом диске, будут потеряны.

2. **Выберите место для установки FreeBSD**

   Если FreeBSD будет единственной операционной системой, этот шаг можно пропустить. Но если FreeBSD будет находиться на диске вместе с другой операционной системой, определите, какой диск или раздел будет использоваться для FreeBSD.

   В архитектурах i386 и amd64 диски могут быть разделены на несколько разделов с использованием одной из двух схем разделения. Традиционная *Главная загрузочная запись* (MBR) содержит таблицу разделов, определяющую до четырёх *основных разделов*. По историческим причинам FreeBSD называет эти основные разделы *слайсами*. Один из этих основных разделов может быть преобразован в *расширенный раздел*, содержащий несколько *логических разделов*. *Таблица разделов GUID* (GPT) — это более новый и простой метод разделения диска. Стандартные реализации GPT позволяют создавать до 128 разделов на диске, что устраняет необходимость в логических разделах.

   Загрузчик FreeBSD требует наличия либо первичного, либо GPT-раздела. Если все первичные или GPT-разделы уже заняты, необходимо освободить один для FreeBSD. Чтобы создать раздел без удаления существующих данных, используйте инструмент для изменения размера разделов, чтобы уменьшить существующий раздел и создать новый раздел, используя освободившееся пространство.

   Альтернативой изменению существующих разделов диска системы является использование Виртуализации, которая позволяет одновременно запускать несколько операционных систем без необходимости изменения разделов.

   Различные бесплатные и коммерческие инструменты для изменения размера разделов перечислены в статье Список программ для работы с разделами диска на Википедии. GParted Live — это бесплатный Live CD, включающий редактор разделов

GParted.

При правильном использовании утилиты для сжатия дисков могут безопасно освободить место для создания нового раздела. Поскольку существует вероятность выбора неправильного раздела, всегда создавайте резервные копии важных данных и проверяйте их целостность перед изменением разделов диска.

Разделы диска с разными операционными системами позволяют установить несколько операционных систем на один компьютер.

3. **Соберите информацию о сети**

   Некоторые методы установки FreeBSD требуют наличия сетевого подключения для загрузки файлов установки. После одного из этапов установки программа-установщик предложит настроить сетевые интерфейсы системы.

   Если в сети есть DHCP-сервер, его можно использовать для автоматической настройки сети. Если DHCP недоступен, следующую информацию о сети для системы необходимо получить у локального сетевого администратора или интернет-провайдера:

   Необходимая сетевая информация

   a. IP-адрес
   b. Маска подсети
   c. IP-адрес шлюза по умолчанию
   d. Доменное имя сети
   e. IP-адреса DNS-серверов сети

4. **Проверьте наличие исправлений в FreeBSD**

   Хотя проект FreeBSD стремится к тому, чтобы каждая версия FreeBSD была максимально стабильной, иногда в процессе могут возникать ошибки. В очень редких случаях эти ошибки влияют на процесс установки. Когда такие проблемы обнаруживаются и исправляются, они отмечаются на странице FreeBSD Errata для каждой версии. Перед установкой проверьте список errata, чтобы убедиться в отсутствии проблем, которые могут повлиять на установку.

   Информация и список ошибок для всех выпусков доступны на странице Информация о релизе FreeBSD.

2.3.1. Подготовьте установочный носитель

Установщик FreeBSD — это не приложение, которое можно запустить из другой операционной системы. Вместо этого скачайте файл установки FreeBSD, запишите его на носитель, соответствующий его типу и размеру (CD, DVD или USB), и загрузите систему для установки с этого носителя.

Файлы для установки FreeBSD доступны на странице скачивания FreeBSD. Имя каждого файла установки включает версию выпуска FreeBSD, архитектуру и тип файла.

Файлы для установки доступны в нескольких форматах, сжатые с помощью xz(1) или несжатые. Форматы различаются в зависимости от архитектуры компьютера и типа носителя.

Типы файлов установки:

- `-bootonly.iso` — это самый маленький установочный файл, так как он содержит только установщик. Для установки требуется работающее интернет-соединение, так как установщик загрузит файлы, необходимые для завершения установки FreeBSD. Этот файл следует записать на оптический носитель.
- `-disc1.iso` — этот файл содержит все необходимые файлы для установки FreeBSD, её исходных кодов и Коллекции портов. Данный файл следует записать на оптический носитель.
- `-dvd1.iso` — этот файл содержит все необходимые файлы для установки FreeBSD, его исходные коды и Коллекцию портов. Также в него включён набор популярных бинарных пакетов для установки оконного менеджера и некоторых приложений, что позволяет установить полноценную систему с носителя без подключения к Интернету. Этот файл следует записать на оптический диск.
- `-memstick.img` — этот файл содержит все необходимые файлы для установки FreeBSD, её исходных кодов и Коллекции портов. Запишите этот файл на USB-накопитель, как показано в Запись образа на USB.
- `-mini-memstick.img` — как и `-bootonly.iso`, не содержит файлов для установки, но загружает их по мере необходимости. Требуется работающее интернет-подключение во время установки. Этот образ должен быть записан на USB-накопитель, как показано в Запись образа на USB.

После загрузки файла образа загрузите как минимум один файл *контрольной суммы (CHECKSUM)* из того же каталога. Доступны два файла *контрольной суммы*, названные в соответствии с номером выпуска и архитектурой. Например: `CHECKSUM.SHA256-FreeBSD-15.1-RELEASE-amd64` и `CHECKSUM.SHA512-FreeBSD-15.1-RELEASE-amd64`.

После загрузки одного из файлов (или обоих) вычислите *контрольную сумму* для файла образа и сравните её с указанной в файле *контрольной суммы*. Обратите внимание, что необходимо сравнивать вычисленную *контрольную сумму* с правильным файлом, так как они соответствуют разным алгоритмам: SHA256 и SHA512. FreeBSD предоставляет утилиты sha256(1) и sha512(1), которые можно использовать для вычисления *контрольной суммы*. В других операционных системах существуют аналогичные программы.

Проверка *контрольной суммы* в FreeBSD может быть выполнена автоматически с помощью sha256sum(1) (и sha512sum(1)) путем выполнения:

```
% sha256sum -c CHECKSUM.SHA256-FreeBSD-15.1-RELEASE-amd64 --ignore-missing
FreeBSD-15.1-RELEASE-amd64-dvd1.iso: OK
```

Хеш-суммы должны полностью совпадать. Если хеш-суммы не совпадают, образ файла повреждён и его необходимо загрузить заново.

Запись образа на USB-накопитель

Файл `*memstick.img` представляет собой *образ* полного содержимого USB-накопителя. Его *нельзя* просто скопировать на целевое устройство как файл. Существует несколько программ для записи `*.img` на USB-накопитель. В этом разделе описаны две такие утилиты.

Прежде чем продолжить, создайте резервную копию всех важных данных на USB-накопителе. Эта процедура удалит все существующие данные на носителе.

Процедура. Использование `dd` для записи образа

Этот пример использует `/dev/da0` в качестве целевого устройства, на которое будет записан образ. Будьте **очень внимательны** при выборе правильного устройства, так как эта команда уничтожит все существующие данные на указанном целевом устройстве.

1. Утилита командной строки доступна в системах BSD, Linux® и Mac OS®. Чтобы записать образ с помощью `dd`, вставьте USB-накопитель и определите его имя устройства. Затем укажите имя загруженного файла установки и имя устройства USB-накопителя. В этом примере записывается образ установки amd64 на первое USB-устройство в существующей системе FreeBSD.

   ```
   # dd if=FreeBSD-15.1-RELEASE-amd64-memstick.img of=/dev/da0 bs=1M conv=sync
   ```

   Если команда завершается с ошибкой, убедитесь, что USB-накопитель не смонтирован и что указано имя диска, а не раздела.

   Некоторые операционные системы могут потребовать выполнения этой команды с sudo(8). Синтаксис dd(1) немного различается на разных платформах; например, Mac OS® требует указания `bs=1m` в нижнем регистре. Системы, такие как Linux®, могут буферизировать запись. Чтобы принудительно завершить все операции записи, используйте sync(8).

Процедура. Запись образа с использованием Windows®

Убедитесь, что указали правильную букву диска, так как все существующие данные на указанном диске будут перезаписаны и уничтожены.

1. **Получение Image Writer для Windows®**

`Image Writer for Windows®` — это бесплатное приложение, которое позволяет корректно записать файл образа на USB-накопитель. Загрузите его с официальной страницы win32diskimager и распакуйте в нужную папку.

2. **Запись образа с помощью Image Writer**

   Дважды щелкните по значку Win32DiskImager, чтобы запустить программу. Убедитесь, что буква диска, указанная в разделе `Device`, соответствует диску с флеш-накопителем. Нажмите на значок папки и выберите образ, который нужно записать на флеш-накопитель. Нажмите **[Save]**, чтобы подтвердить имя файла образа. Убедитесь, что все правильно и что никакие папки на флеш-накопителе не открыты в других окнах. Когда все готово, нажмите **[Write]**, чтобы записать файл образа на флеш-накопитель.

2.4. Начало установки

По умолчанию установка не вносит никаких изменений на диск(и) до следующего сообщения:

```
Your changes will now be written to disk. If you
have chosen to overwrite existing data, it will
be PERMANENTLY ERASED. Are you sure you want to
commit your changes?
```

Установку можно прервать в любой момент до этого предупреждения. Если есть опасения, что что-то настроено неправильно, просто выключите компьютер до этого момента, и никакие изменения не будут внесены в диски системы.

В этом разделе описывается, как загрузить систему с установочного носителя, подготовленного в соответствии с инструкциями в пункте Подготовка установочного носителя. При использовании загрузочной USB-флешки подключите её к компьютеру перед включением. При загрузке с CD или DVD включите компьютер и вставьте носитель при первой возможности. Настройка системы для загрузки с подключённого носителя зависит от архитектуры.

2.4.1. Меню загрузчика FreeBSD

После загрузки системы с установочного носителя появится меню, подобное следующему:

Рисунок 1. Меню загрузчика FreeBSD

По умолчанию меню будет ждать десять секунд ввода пользователя перед загрузкой установщика FreeBSD или, если FreeBSD уже установлена, перед загрузкой системы. Чтобы приостановить таймер загрузки для просмотра вариантов, нажмите `Пробел`. Для выбора варианта нажмите соответствующую подсвеченную цифру, символ или клавишу. Доступны следующие варианты.

- `Загрузка в многопользовательском режиме (Boot Multi User)`: Это продолжит процесс загрузки FreeBSD. Если таймер загрузки был приостановлен, нажмите `1`, `B` (в верхнем или нижнем регистре) или `Enter`.
- `Загрузка в однопользовательском режиме (Boot Single User)`: Этот режим может быть использован для исправления существующей установки FreeBSD, как описано в “Однопользовательский режим”. Для входа в этот режим нажмите `2` или `S` в верхнем или нижнем регистре.
- `Выход в загрузчик (Escape to loader prompt)`: Это загрузит систему в режим восстановления с ограниченным набором низкоуровневых команд. Данный режим описан в разделе “Этап три”. Для загрузки в этом режиме нажмите `3` или `Esc`.
- `Перезагрузка (Reboot)`: Перезагружает систему.
- `Cons`: Позволяет продолжить установку через `видео`, `последовательный порт`, `двухконсольную конфигурацию (основная консоль – последовательный порт)` или `двухконсольную конфигурацию (основная консоль – видео)`
- `Ядро (Kernel)`: Загружает другое ядро.
- `Параметры загрузки (Boot Options)`: Открывает меню, показанное на рисунке Меню параметров загрузки FreeBSD и описанное после него.

Рисунок 2. Меню параметров загрузки FreeBSD

Меню параметров загрузки разделено на две части. Первая часть позволяет либо вернуться в главное меню загрузки, либо сбросить все изменённые параметры к значениям по умолчанию.

Следующий раздел позволяет переключать доступные опции в состояние Вкл или Выкл, нажимая выделенную цифру или символ соответствующей опции. Система всегда будет загружаться с использованием текущих настроек этих опций, пока они не будут изменены. С помощью этого меню можно переключать несколько опций:

- ACPI: Если система зависает во время загрузки, попробуйте переключить этот параметр в положение Выкл. Этот параметр присутствует только в том случае, если поддержка ACPI доступна, но не обязательна.
- Безопасный режим (Safe Mode): Если система всё ещё зависает во время загрузки, даже когда Поддержка ACPI установлена в Выкл, попробуйте установить этот параметр в Вкл.
- Однопользовательский (Single User): Установите этот параметр в Вкл, чтобы исправить существующую установку FreeBSD, как описано в “Однопользовательский режим”. После устранения проблемы верните значение Выкл.
- Подробно (Verbose): Установите этот параметр в Вкл, чтобы видеть более подробные сообщения в процессе загрузки. Это может быть полезно при диагностике проблем с оборудованием.

После выбора необходимых параметров нажмите `1` или `Backspace`, чтобы вернуться в главное меню загрузки, затем нажмите `Enter` для продолжения загрузки FreeBSD. Появится серия сообщений загрузки, пока FreeBSD выполняет обнаружение аппаратных устройств и загружает программу установки. После завершения загрузки будет отображено приветственное меню, показанное на рисунке Приветственное меню.

Рисунок 3. Приветственное меню

Нажмите `Enter`, чтобы выбрать вариант по умолчанию **[Install]** и перейти к установке. В остальной части этой главы описывается, как использовать этот установщик. В противном случае используйте стрелки вправо или влево или выделенную цветом букву, чтобы выбрать нужный пункт меню. Кнопка **[Shell]** позволяет получить доступ к оболочке FreeBSD для использования командной строки и подготовки дисков перед установкой. Вариант **[Live CD]** позволяет попробовать FreeBSD перед установкой. Версия Live CD описана в пункте Использование Live CD.

Для просмотра загрузочных сообщений, включая информацию об обнаруженных аппаратных устройствах, нажмите `S` (верхний или нижний регистр) и затем `Enter`, чтобы перейти в оболочку. В командной строке оболочки введите `more /var/run/dmesg.boot` и используйте пробел для прокрутки сообщений. По завершении введите `exit`, чтобы вернуться в меню приветствия.

2.5. Использование bsdinstall

В этом разделе показана последовательность меню bsdinstall и тип информации, которая будет запрошена перед установкой системы. Используйте клавиши со стрелками, чтобы выделить пункт меню, затем `Space` для выбора или отмены выбора этого пункта. По завершении нажмите `Enter`, чтобы сохранить выбор и перейти к следующему экрану.

2.5.1. Меню выбора раскладки клавиатуры

Перед началом процесса bsdinstall загрузит файлы раскладки клавиатуры, как показано на рисунке Загрузка раскладки клавиатуры.

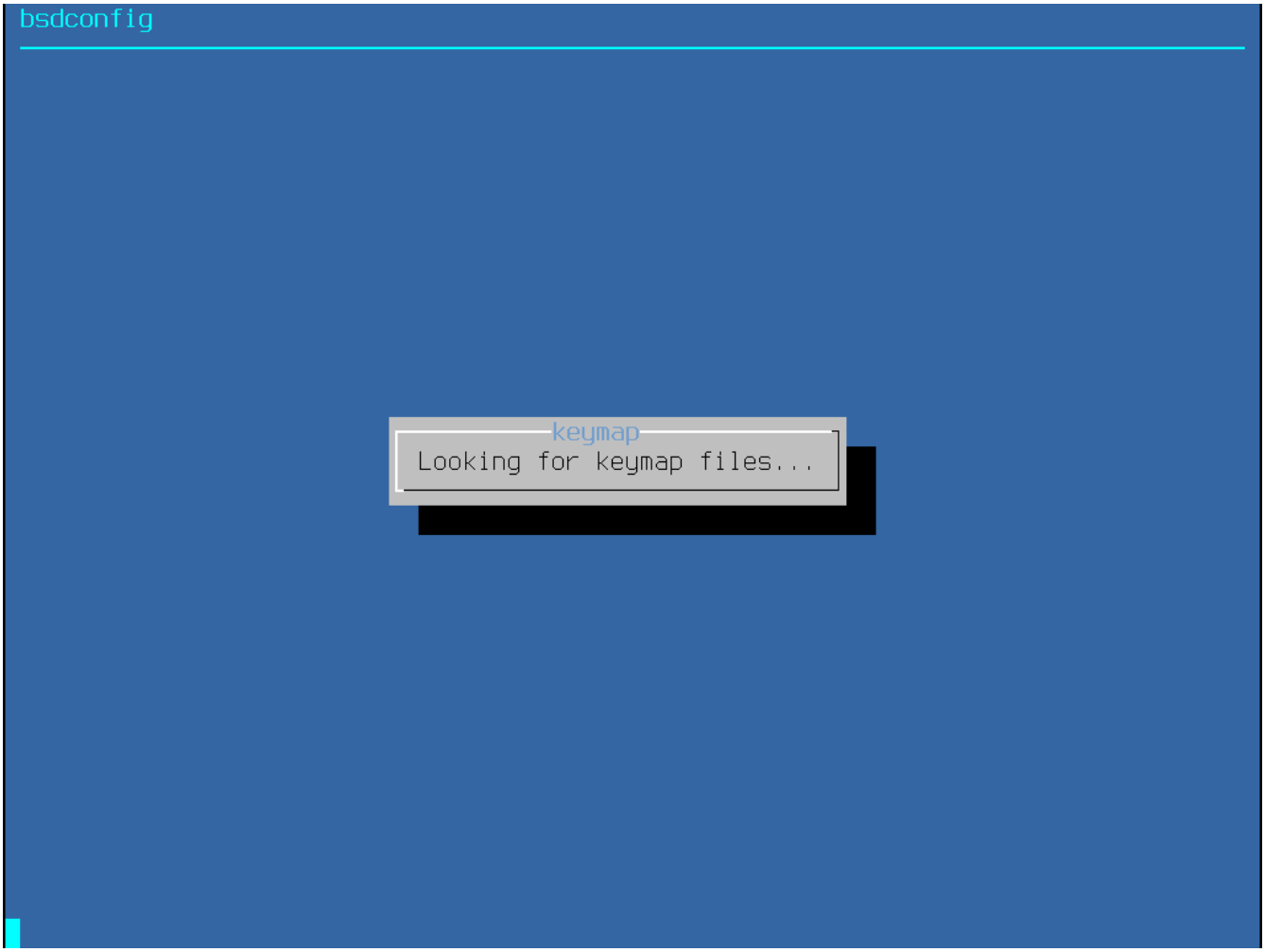


Рисунок 4. Загрузка раскладки клавиатуры

После загрузки раскладок клавиатур bsdinstall отображает меню, показанное на рисунке Меню выбора раскладки клавиатуры. Используйте стрелки вверх и вниз, чтобы выбрать раскладку, наиболее точно соответствующую клавиатуре, подключённой к системе. Нажмите `Enter`, чтобы сохранить выбор.

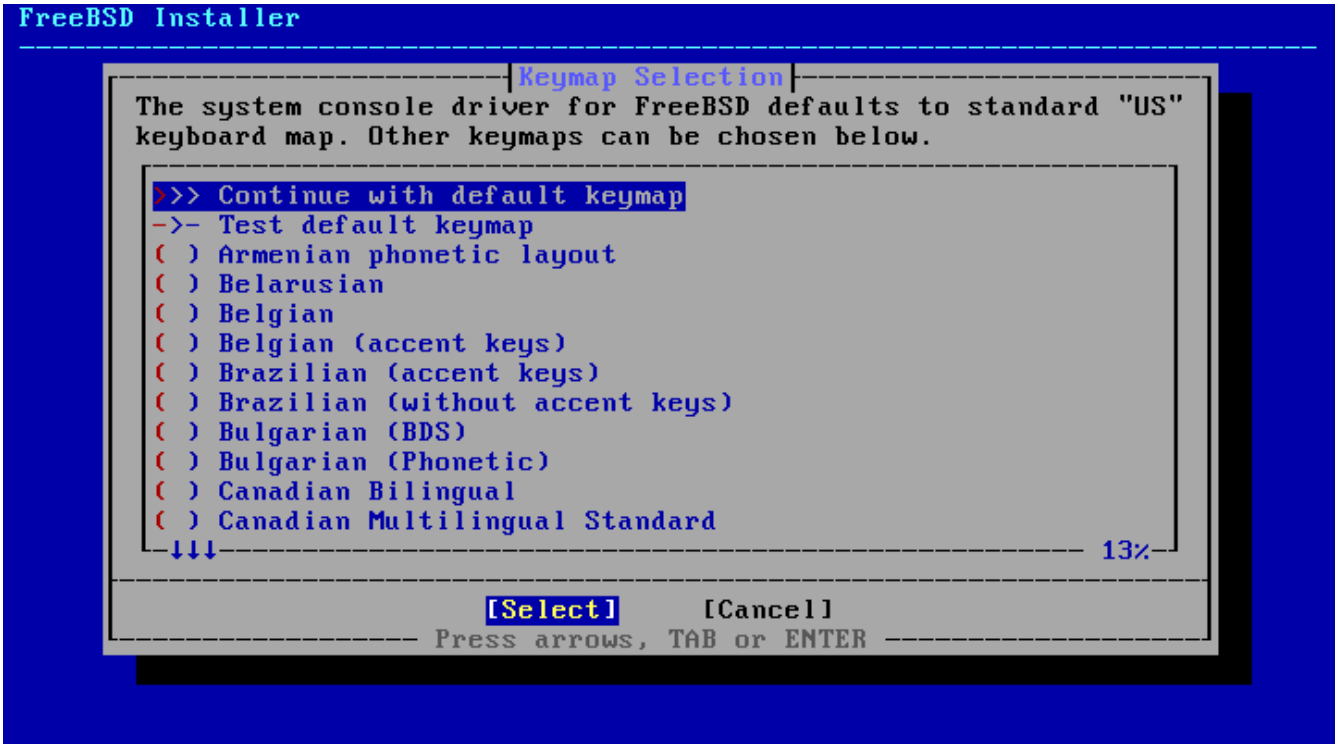


Рисунок 5. Меню выбора раскладки клавиатуры

Нажатие `Esc` выйдет из этого меню и использует раскладку по умолчанию. Если выбор раскладки неочевиден, United States of America ISO-8859-1 также является безопасным вариантом.

Кроме того, при выборе другой раскладки клавиатуры пользователь может проверить её и убедиться в правильности перед продолжением, как показано на рисунке Меню проверки раскладки клавиатуры.

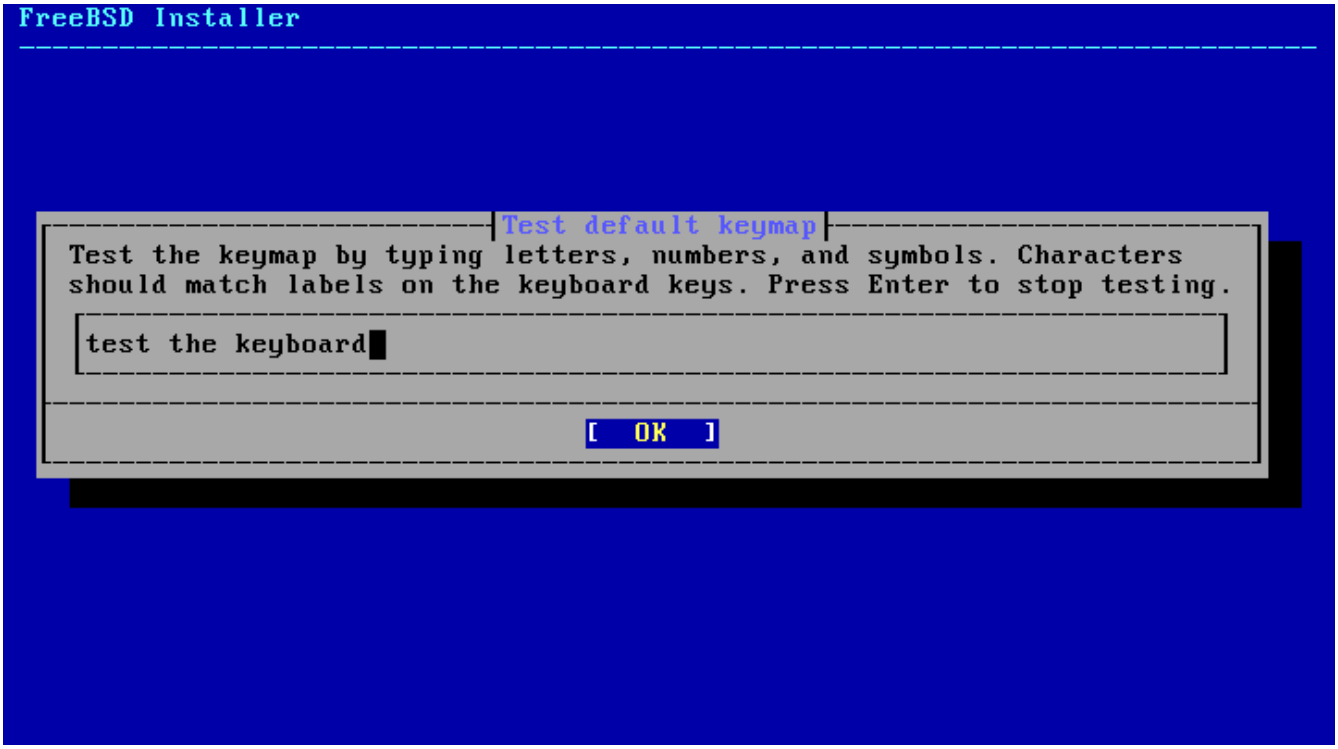


Рисунок 6. Меню тестирования раскладки клавиатуры

2.5.2. Установка имени хоста

Следующее меню bsdinstall используется для установки имени хоста вновь устанавливаемой системы.

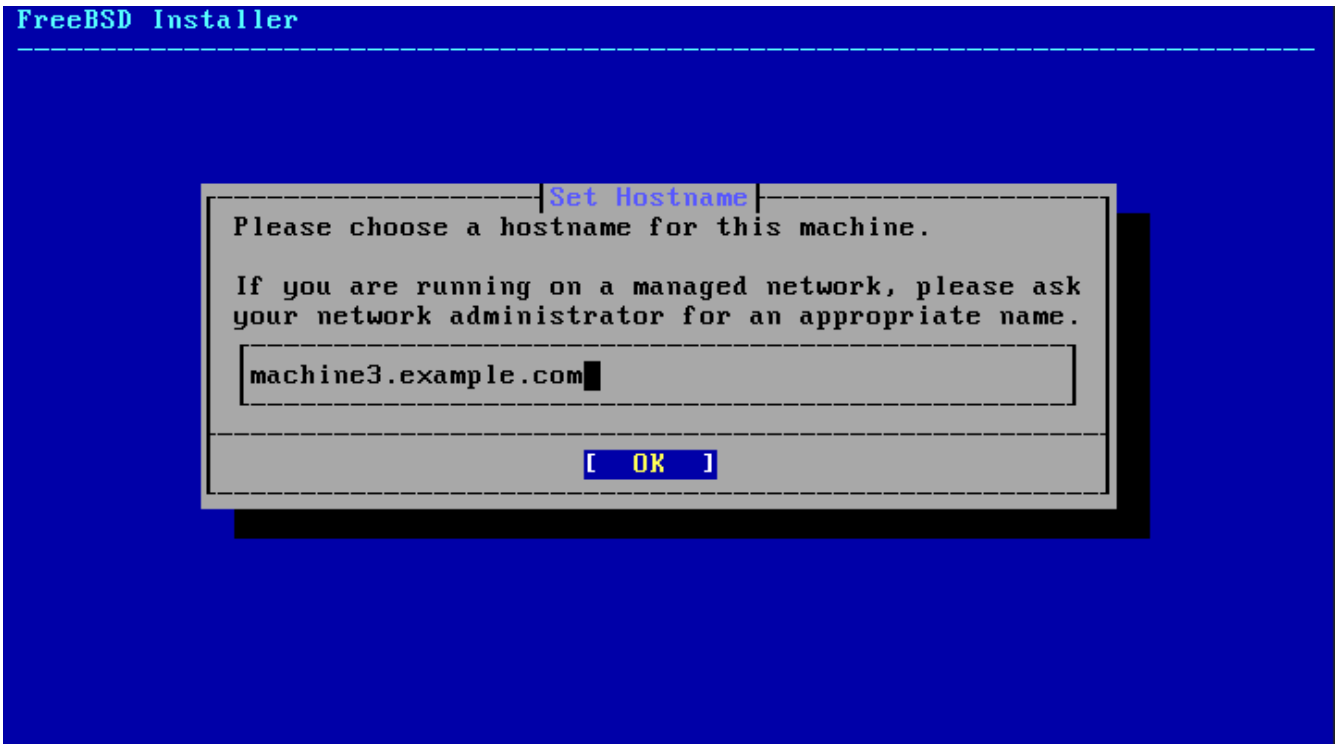


Рисунок 7. Установка имени хоста

Введите имя хоста, уникальное для сети. Оно должно быть полным доменным именем, например `machine3.example.com`.

2.5.3. Выбор типа установки

В следующем меню bsdinstall спросит, следует ли устанавливать базовую систему с использованием традиционных наборов дистрибутива (distribution sets) или нового метода, основанного на пакетах.

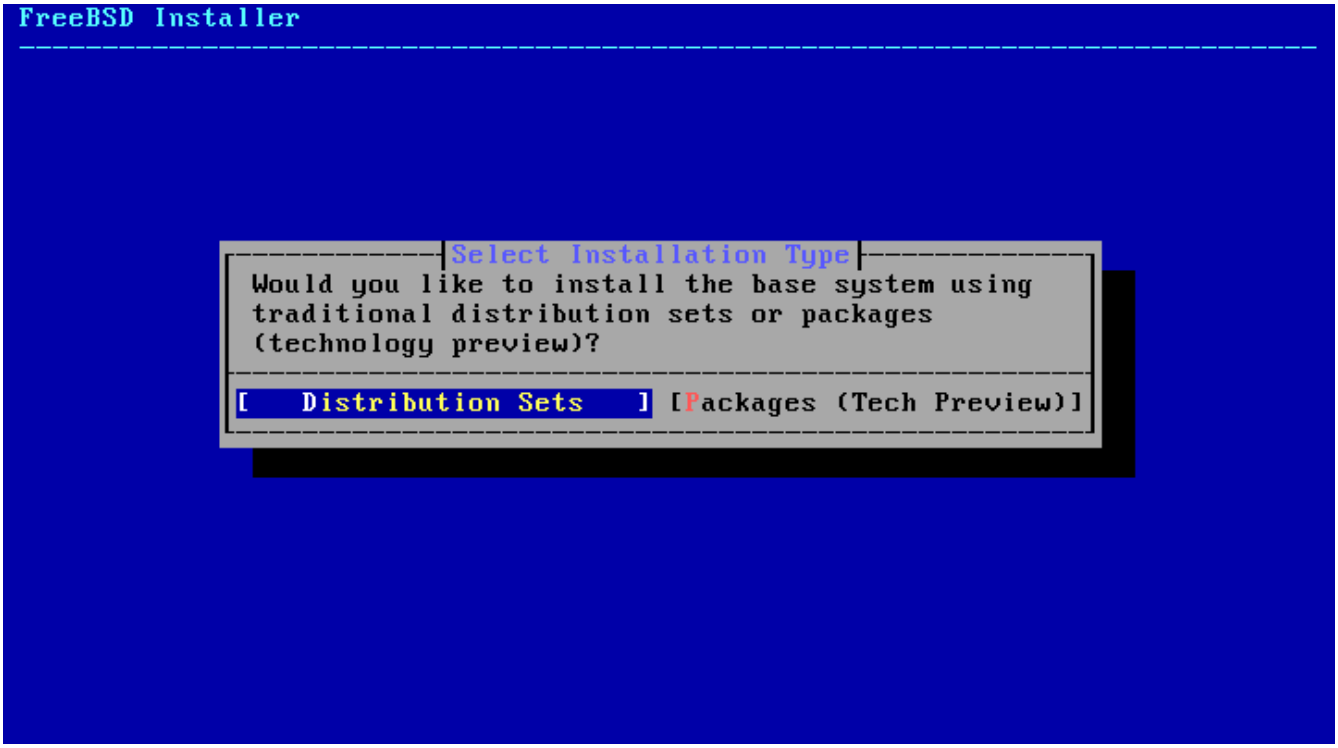


Рисунок 8. Выбор типа установки

Начиная с FreeBSD 15.0, установщик предлагает два различных подхода для развёртывания и последующего обновления базовой системы. Оба варианта устанавливают одно и то же ядро и пользовательское окружение, но принципиально различаются в способах доставки и сопровождения с течением времени.

- `Distribution Sets` (Наборы дистрибутива) — классический метод установки, используемый с первых дней существования проекта. Базовая система распаковывается из коллекции сжатых tar-архивов. Обновления выполняются исключительно с помощью утилиты freebsd-update(8). Этот метод полностью поддерживается, стабилен и остаётся методом, используемым по умолчанию для большинства установочных носителей в серии 15.x.
- `Packages (Tech Preview)` (Пакеты — техническая предварительная версия) — устанавливает базовую систему как набор отдельных пакетов из репозитория `FreeBSD-base`. После установки вся операционная система — как базовая, так и любое стороннее программное обеспечение — может быть обновлена одной командой `pkg upgrade`. Этот метод, известный как `pkgbase`, объединяет управление системой в рамках инфраструктуры pkg(8).

Вариант `Packages (Tech Preview)` (Пакеты — техническая предварительная версия) предлагается для тестирования и раннего внедрения. Хотя ожидается, что этот метод станет методом, используемым по умолчанию в FreeBSD 16.0, пользователям, которые предпочитают консервативное, проверенное временем решение для обновлений, следует выбирать `Distribution Sets` (Наборы дистрибутива) для производственных сред в течение жизненного цикла 15.x.

2.5.4. Выбор компонентов для установки

Затем `bsdinstall` предложит выбрать дополнительные компоненты для установки.

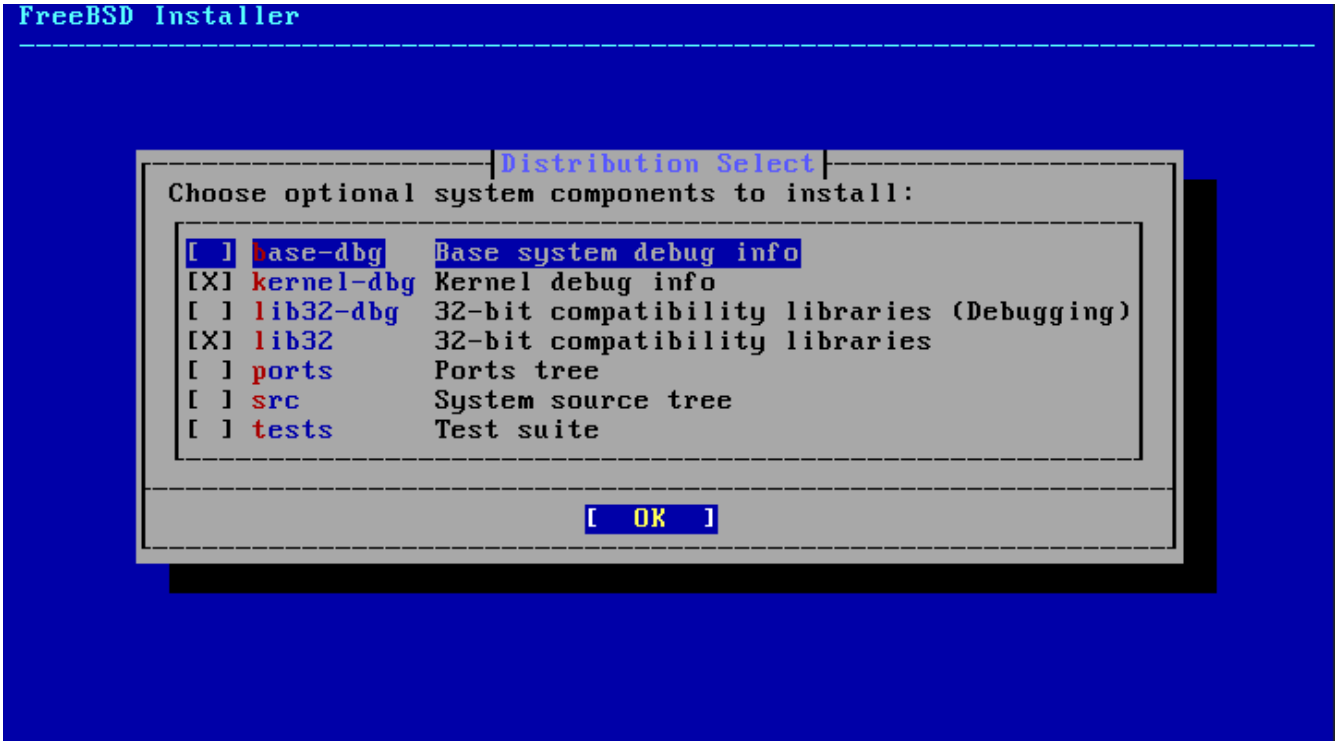


Рисунок 9. Выбор компонентов для установки

Выбор компонентов для установки в значительной степени зависит от предполагаемого использования системы и доступного дискового пространства. Базовая система FreeBSD, включающая ядро и пользовательское окружение (*base system*), устанавливается всегда. В зависимости от архитектуры некоторые из этих компонентов могут отсутствовать:

- `base-dbg` - Базовые инструменты, такие как cat и ls, среди многих других, с активированными отладочными символами.
- `kernel-dbg` - Ядро и модули с включёнными отладочными символами.
- `lib32-dbg` - Совместимые библиотеки для запуска 32-битных приложений на 64-битной версии FreeBSD с активированными отладочными символами.
- `lib32` - Совместимые библиотеки для запуска 32-битных приложений на 64-битной версии FreeBSD.
- `ports` - Коллекция портов FreeBSD представляет собой набор файлов, автоматизирующих загрузку, компиляцию и установку сторонних программных пакетов. В разделе Установка приложений: Пакеты и Порты рассматривается, как использовать Коллекцию портов.

Программа установки не проверяет наличие достаточного места на диске. Выбирайте этот вариант, только если доступно достаточно места на жёстком диске. Коллекция портов FreeBSD занимает около 3 GB места на диске.

- `src` - Полный исходный код FreeBSD, включая как ядро, так и пользовательское

пространство. Хотя он не требуется для большинства приложений, он может быть необходим для сборки драйверов устройств, модулей ядра или некоторых приложений из коллекции портов. Также он используется для разработки самой FreeBSD. Полное дерево исходных кодов занимает 1 ГБ дискового пространства, а перекомпиляция всей системы FreeBSD требует дополнительно 5 ГБ пространства.

- `tests` — Набор тестов FreeBSD.

2.5.5. Установка по сети

Меню, показанное на рисунке Установка из сети, появляется только при установке с `-bootonly.iso` или `-mini-memstick.img`, так как эти носители не содержат копий файлов установки. Поскольку файлы установки должны быть получены через сетевое соединение, это меню указывает на необходимость предварительной настройки сетевого интерфейса. Если это меню появляется на любом этапе процесса, не забудьте следовать инструкциям из раздела Настройка сетевых интерфейсов.

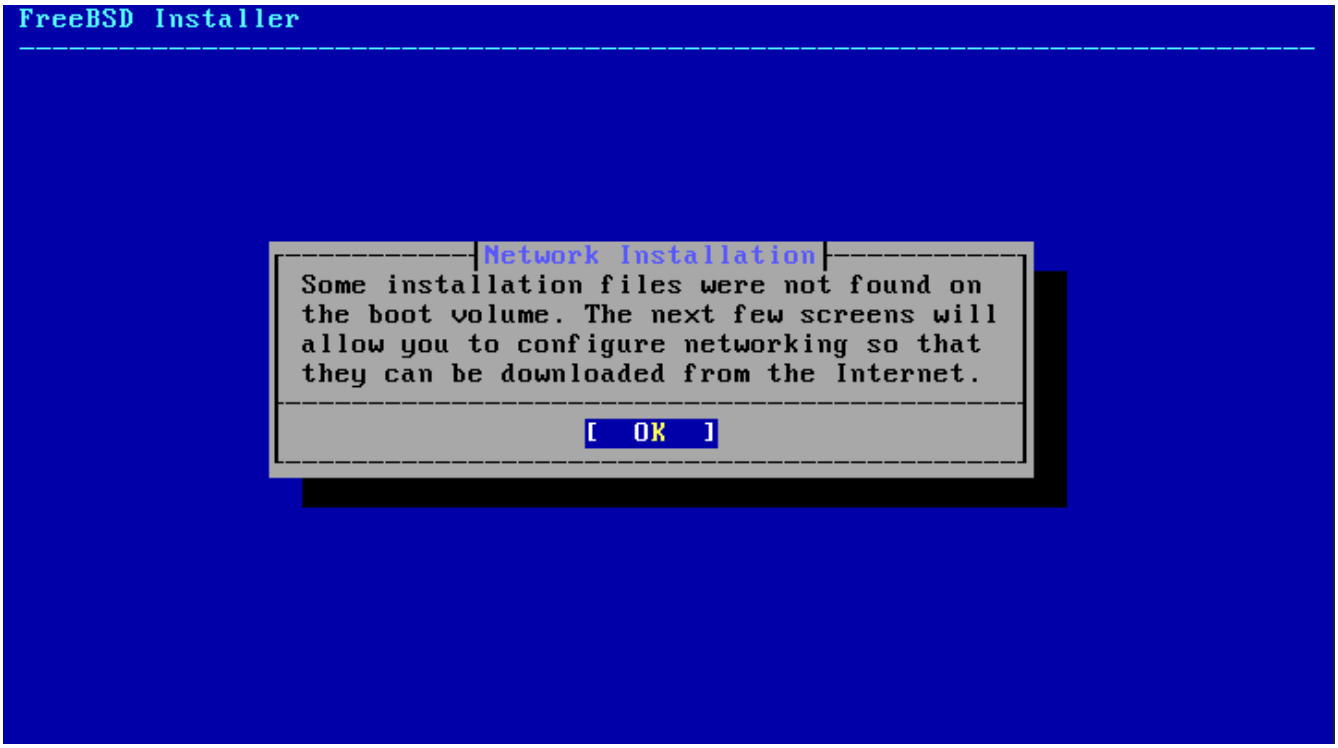


Рисунок 10. Установка по сети

2.6. Выделение дискового пространства

Следующее меню используется для определения метода распределения дискового пространства.

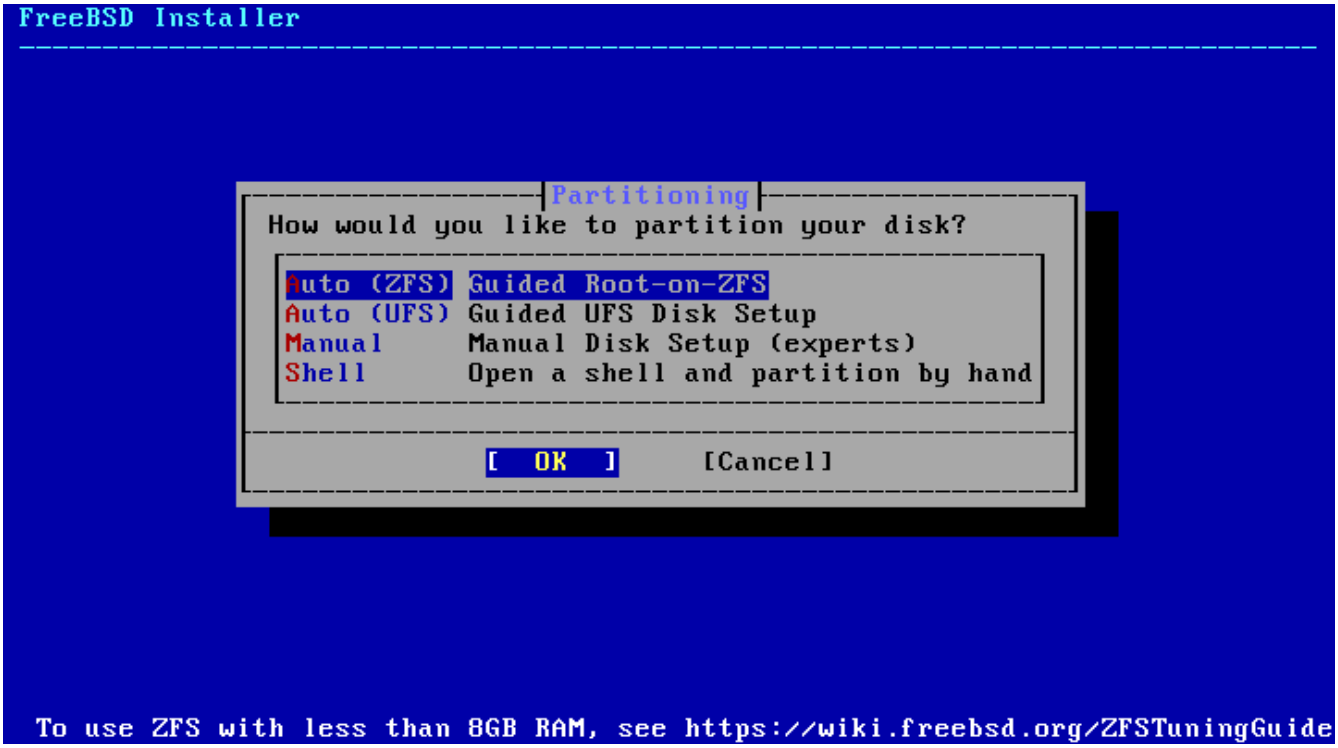


Рисунок 11. Варианты разметки разделов

bsdinstall предоставляет пользователю четыре метода распределения дискового пространства:

- `Auto (ZFS)` — автоматическое разбиение на разделы создаёт систему с корневым разделом на ZFS и возможностью использования шифрования GELI для *загрузочных окружений*.
- `Auto (UFS)` — автоматическое разбиение диска с использованием файловой системы `UFS`.
- `Ручное (Manual)` разбиение позволяет опытным пользователям создавать настраиваемые разделы с помощью параметров меню.
- `Оболочка (Shell)` открывает командную оболочку, где опытные пользователи могут создавать настраиваемые разделы с помощью утилит командной строки, таких как gpart(8), fdisk(8) и bsdlabel(8).

Этот раздел описывает, что следует учитывать при разметке разделов диска. Затем демонстрируется, как использовать различные методы разметки.

2.6.1. Проектирование разметки разделов

По умолчанию схема разметки разделов для файловых систем включает одну файловую систему для всей системы. При использовании `UFS` может быть целесообразно рассмотреть использование нескольких файловых систем, если у вас достаточно места на диске или несколько дисков. При разметке файловых систем учитывайте, что жёсткие диски передают данные быстрее с внешних дорожек по сравнению с внутренними. Таким образом, небольшие и часто используемые файловые системы должны располагаться ближе к внешней части диска, а крупные разделы, такие как `/usr`, следует размещать ближе к внутренней части диска. Рекомендуется создавать разделы в следующем порядке: `/`, раздел подкачки, `/var` и `/usr`.

Размер раздела `/var` зависит от предполагаемого использования машины. Этот раздел используется для хранения почтовых ящиков, файлов журналов и очередей печати. Почтовые ящики и файлы журналов могут достигать неожиданно больших размеров в зависимости от количества пользователей и срока хранения журналов. В среднем большинству пользователей редко требуется более одного гигабайта свободного места на диске в `/var`.

Иногда в `/var/tmp` требуется много дискового пространства. При установке нового программного обеспечения утилиты управления пакетами извлекают временную копию пакетов в `/var/tmp`. У крупных пакетов, таких как Firefox или LibreOffice, могут возникнуть сложности во время установки, если в `/var/tmp` недостаточно места на диске.

Раздел `/usr` содержит множество файлов, поддерживающих систему, включая коллекцию портов FreeBSD и исходный код системы. Для этого раздела рекомендуется выделить не менее 2 гигабайт пространства. Также учтите, что домашние каталоги пользователей по умолчанию размещаются в `/usr/home`, но могут быть расположены на другом разделе. По умолчанию `/home` является символической ссылкой на `/usr/home`.

При выборе размера разделов учитывайте требования к пространству. Нехватка места в одном разделе при почти полном отсутствии использования другого может создать проблемы.

Как правило, размер раздела подкачки должен быть примерно в два раза больше объёма физической памяти (RAM). Системам с малым объёмом RAM (меньше для конфигураций с большим объёмом памяти) может быть полезно иметь больше область подкачки. Слишком маленький объём подкачки может привести к неэффективности работы кода сканирования страниц виртуальной памяти и создать проблемы в будущем при добавлении памяти.

На больших системах с несколькими SCSI-дисками или несколькими IDE-дисками, работающими на разных контроллерах, рекомендуется настраивать раздел подкачки на каждом диске, вплоть до четырёх дисков. Разделы подкачки должны быть примерно одинакового размера. Ядро может обрабатывать разделы произвольного размера, но внутренние структуры данных масштабируются до 4-кратного размера наибольшего раздела подкачки. Поддержание разделов подкачки примерно одинакового размера позволит ядру оптимально распределять пространство подкачки по дискам. Большие размеры подкачки могут вызвать предупреждение ядра о суммарном объёме настроенной подкачки. Лимит можно увеличить, выделив больше памяти для отслеживания распределения подкачки, как указано в сообщении с предупреждением. Это может облегчить восстановление после сбоя программы без необходимости перезагрузки системы.

Правильное разделение системы на разделы предотвращает распространение фрагментации, возникающей в небольших разделах с высокой нагрузкой на запись, на преимущественно читаемые разделы. Размещение разделов с высокой нагрузкой на запись ближе к краю диска повышает производительность ввода-вывода в тех разделах, где это наиболее критично. Хотя производительность ввода-вывода в крупных разделах также может быть важной, их смещение ближе к краю диска не даст значительного прироста производительности по сравнению с перемещением `/var` к краю.

2.6.2. Разметка диска с использованием UFS с помощью мастера

При выборе этого метода отобразится меню с доступными дисками. Если подключено несколько дисков, выберите тот, на который будет установлена FreeBSD.

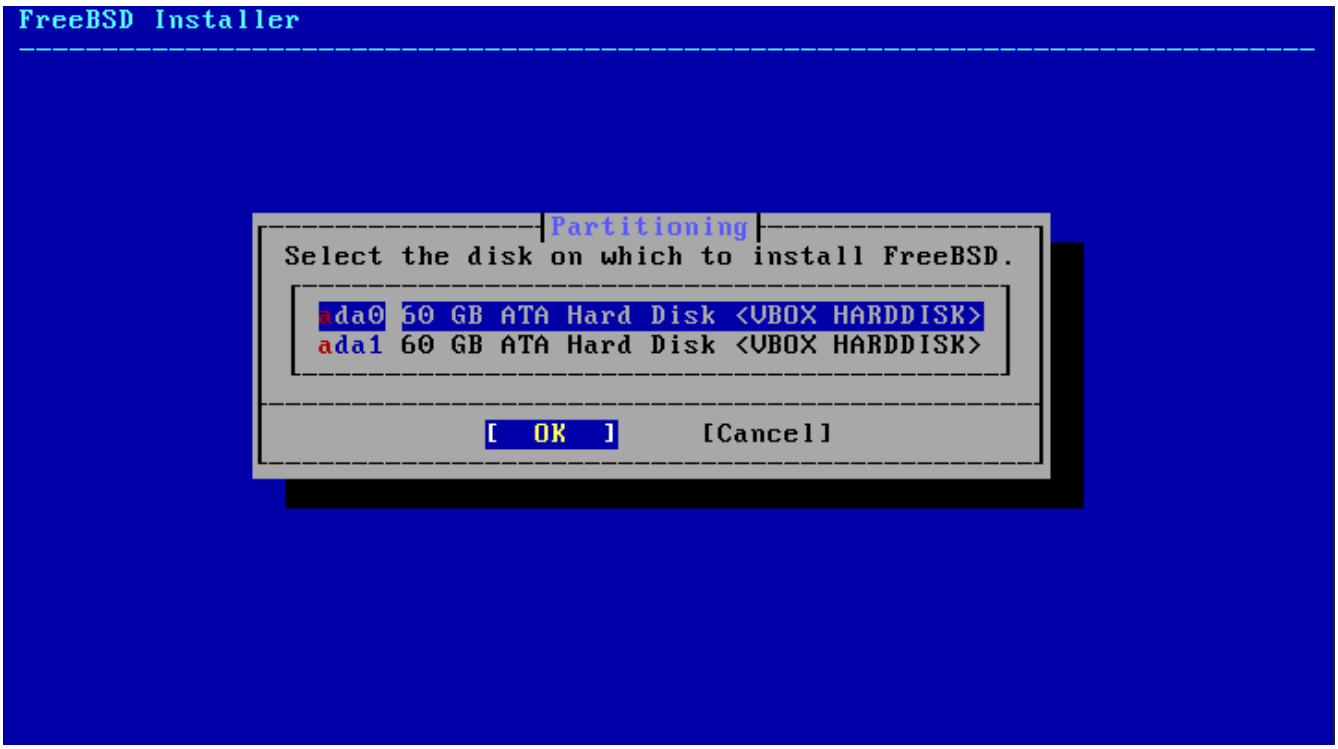


Рисунок 12. Выбор из нескольких дисков

После выбора диска в следующем меню предлагается установить систему на весь диск или создать раздел в свободном пространстве. Если выбран **[Весь диск]**, автоматически создаётся общая схема разделов, занимающая весь диск. При выборе **[Раздел]** создаётся схема разделов из неиспользуемого пространства на диске.

Рисунок 13. Выбор всего диска или раздела

Следующее меню показывает список доступных типов схем разделов. GPT обычно является наиболее подходящим выбором для компьютеров amd64. Более старые компьютеры, несовместимые с GPT, должны использовать MBR. Остальные схемы разделов, как правило, применяются для редких или устаревших компьютеров. Дополнительная информация доступна в таблице Схемы разделов.

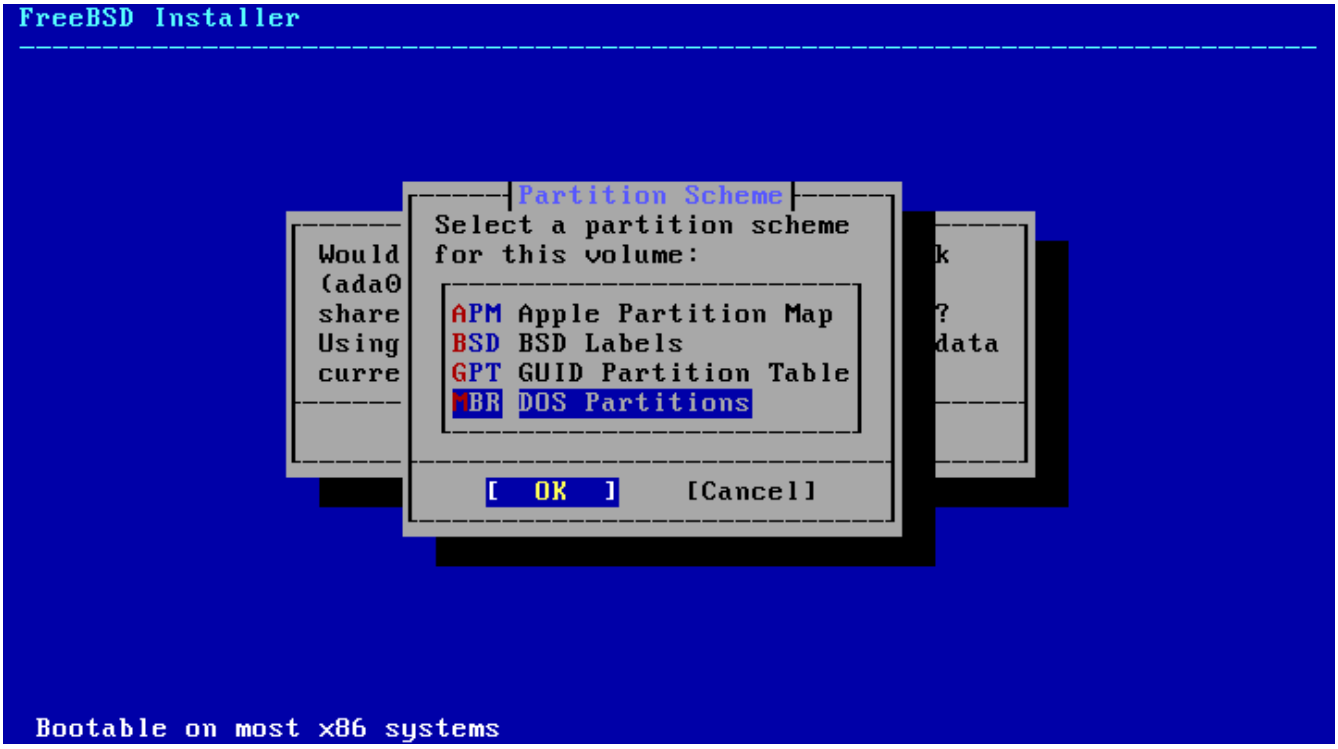


Рисунок 14. Выбор схемы разделов

После создания разметки разделов просмотрите её, чтобы убедиться, что она соответствует требованиям установки. Выбор **[Отменить (Revert)]** вернёт разделы к исходным значениям. Нажатие **[Автоматически (Auto)]** воссоздаст автоматические разделы FreeBSD. Разделы также можно создавать, изменять или удалять вручную. Когда разметка разделов будет правильной, выберите **[Готово (Finish)]** для продолжения установки.

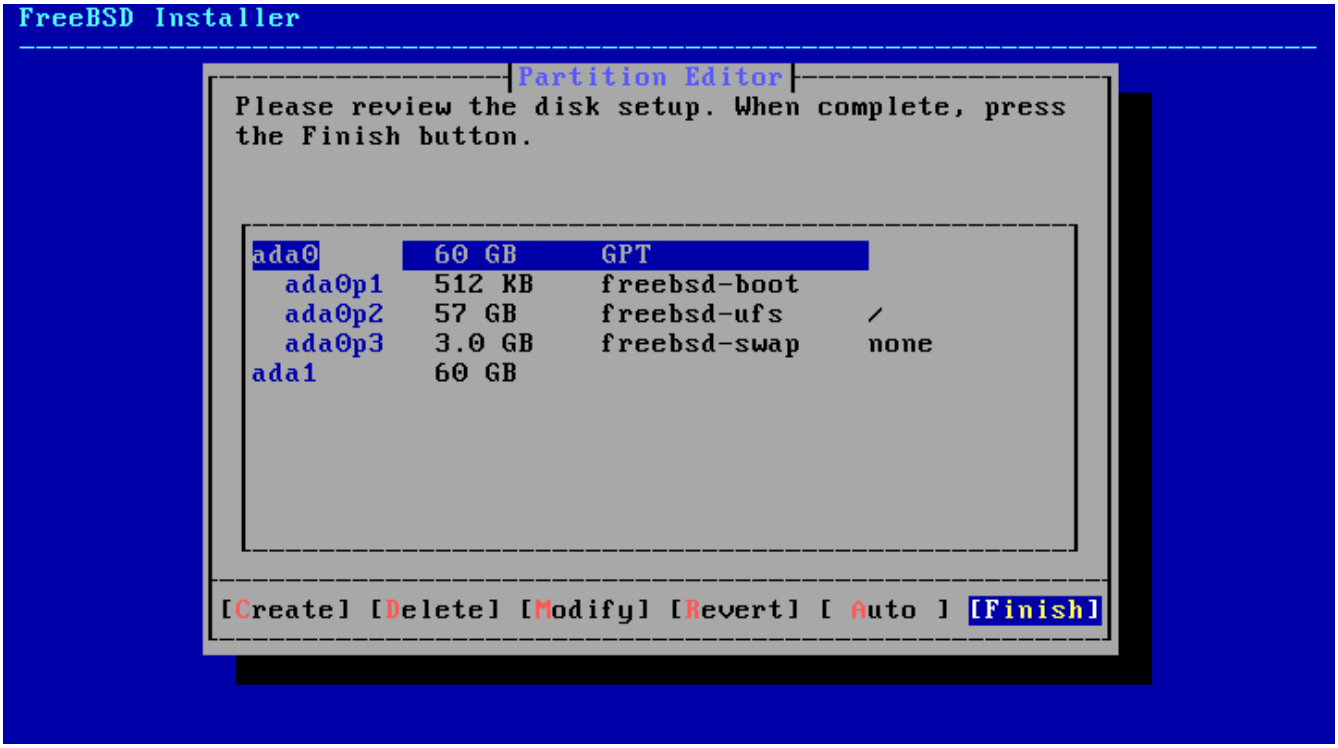


Рисунок 15. Проверка созданных разделов

После настройки дисков в следующем меню предоставляется последняя возможность внести изменения перед форматированием выбранных накопителей. Если изменения необходимы, выберите **[Назад]**, чтобы вернуться в главное меню разметки. **[Отменить & Выйти]** завершает работу установщика без внесения изменений в накопитель. В противном случае выберите **[Применить]**, чтобы начать процесс установки.

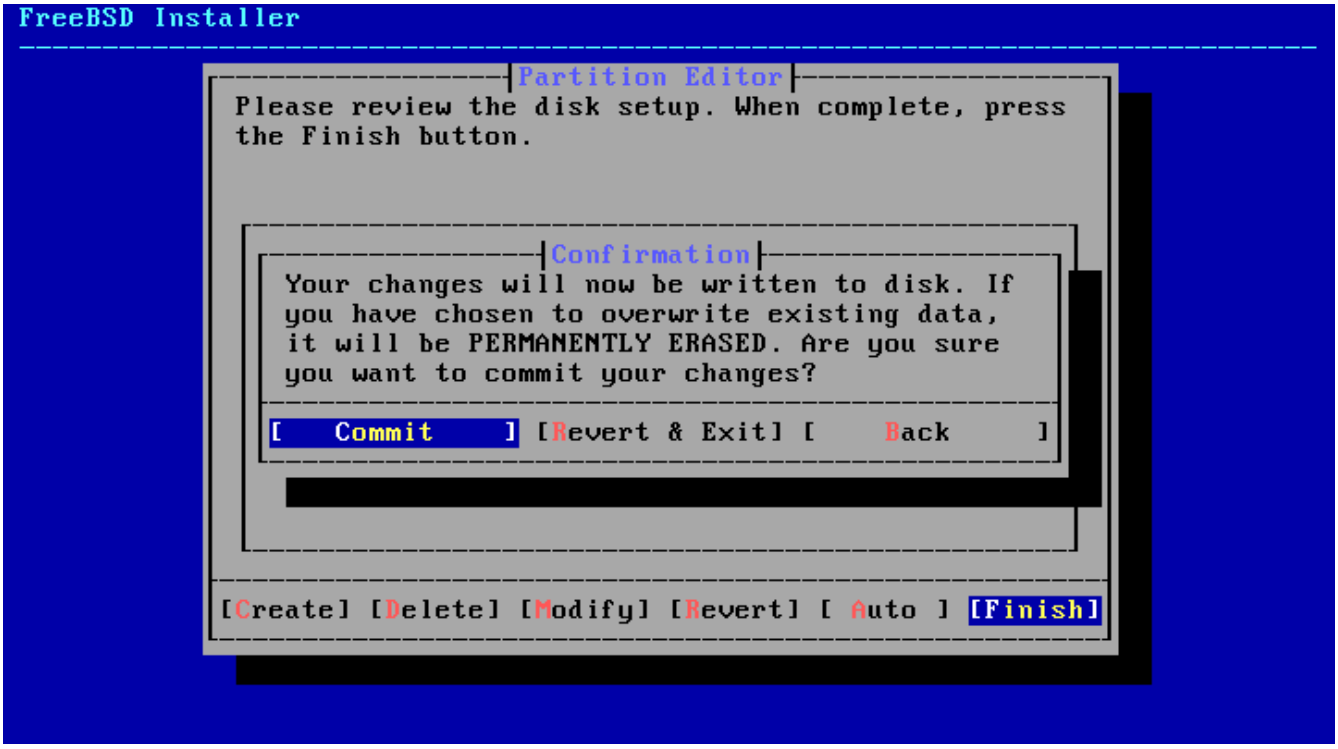


Рисунок 16. Окончательное подтверждение

Для продолжения процесса установки перейдите к разделу Загрузка файлов дистрибутива.

2.6.3. Ручное разбиение на разделы

Выбор этого метода открывает редактор разделов:

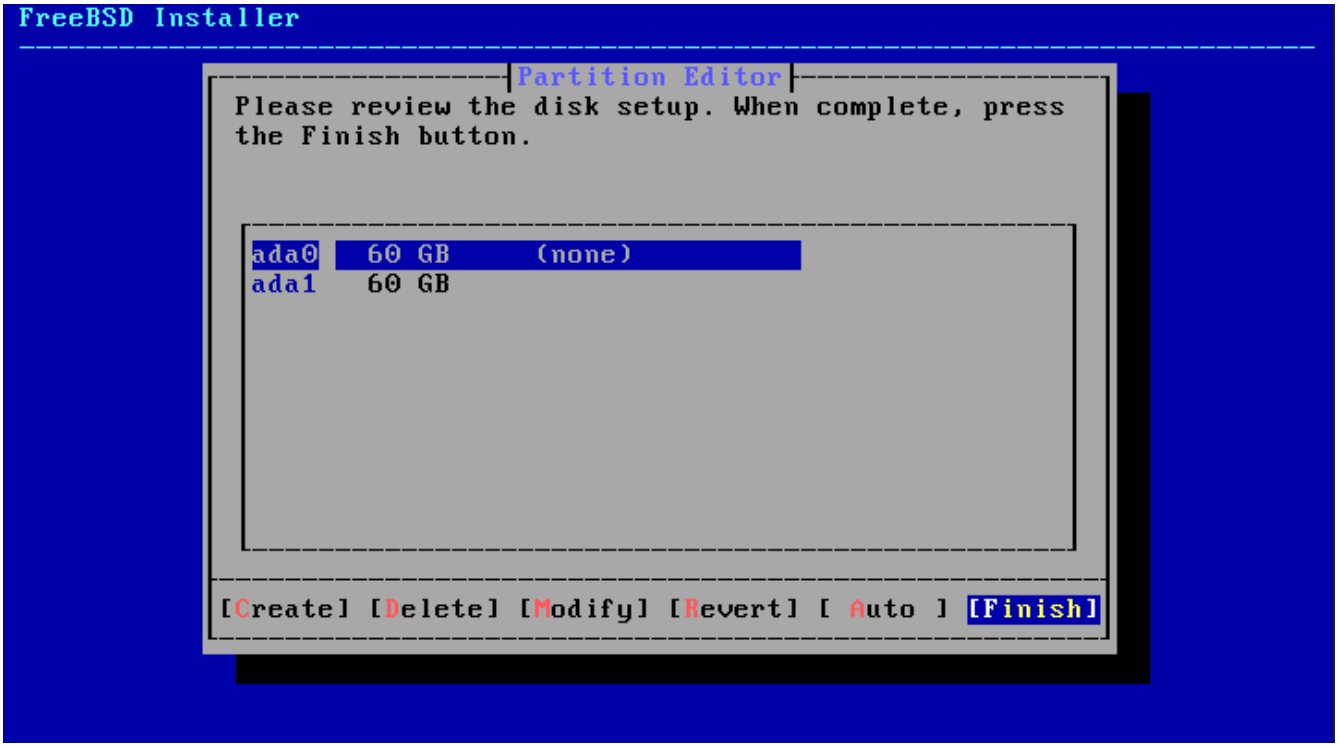


Рисунок 17. Ручное создание разделов

Выделите диск для установки (в данном примере `ada0`) и нажмите **[Создать (Create)]**, чтобы отобразить меню доступных схем разделов:

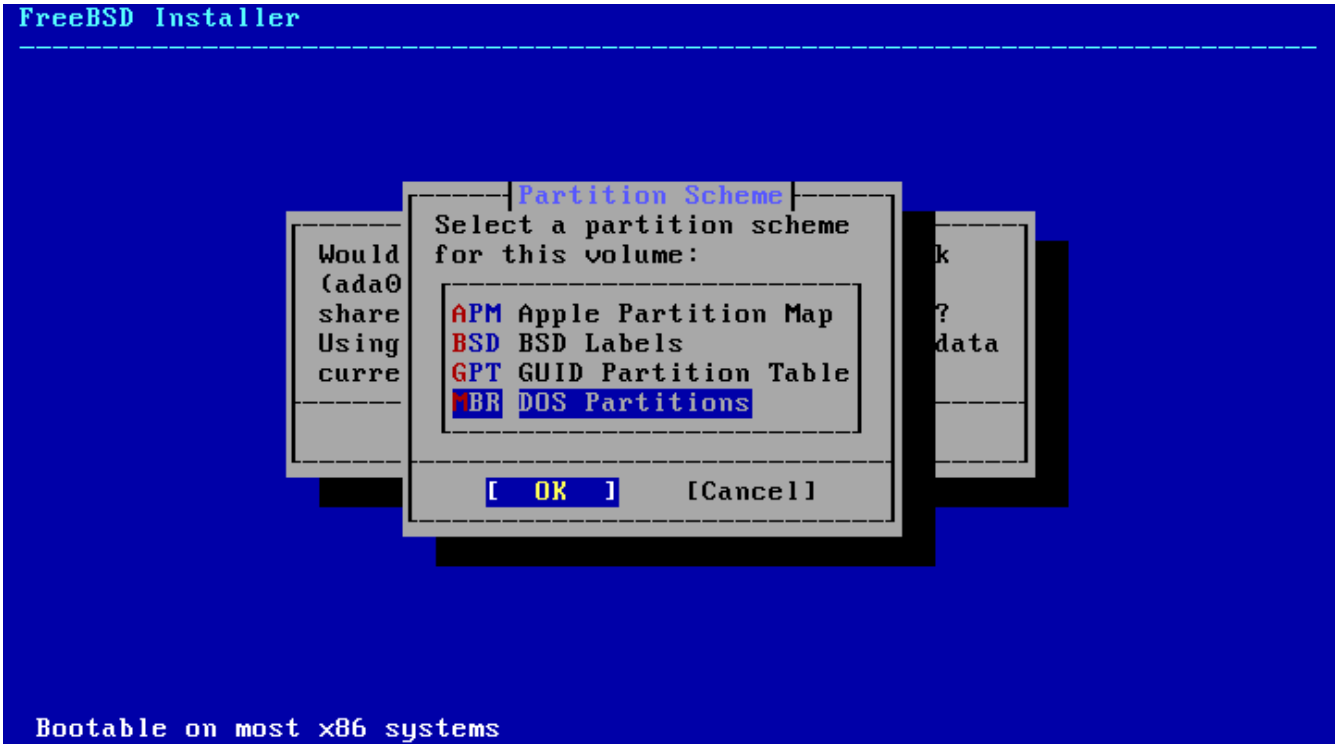


Рисунок 18. Ручное создание разделов

Для компьютеров на архитектуре amd64 обычно наиболее подходящим выбором является GPT. Старые компьютеры, несовместимые с GPT, должны использовать MBR. Остальные

схемы разделов, как правило, применяются для редких или устаревших компьютеров.

Таблица 1. Схемы разделов

Сокращение	Описание
APM	Apple Partition Map, используется в PowerPC®.
BSD	Метка BSD без MBR, иногда называемая *опасно выделенным режимом*, так как не-BSD утилиты для работы с дисками могут её не распознать.
GPT	GUID Partition Table.
MBR	Master Boot Record.

После выбора и создания схемы разделов снова выберите **[Создать]**, чтобы создать разделы. Клавиша `Tab` используется для перехода между полями (после перебора **[<OK>]**, **[<Параметры>]** и **[<Отмена>]**).

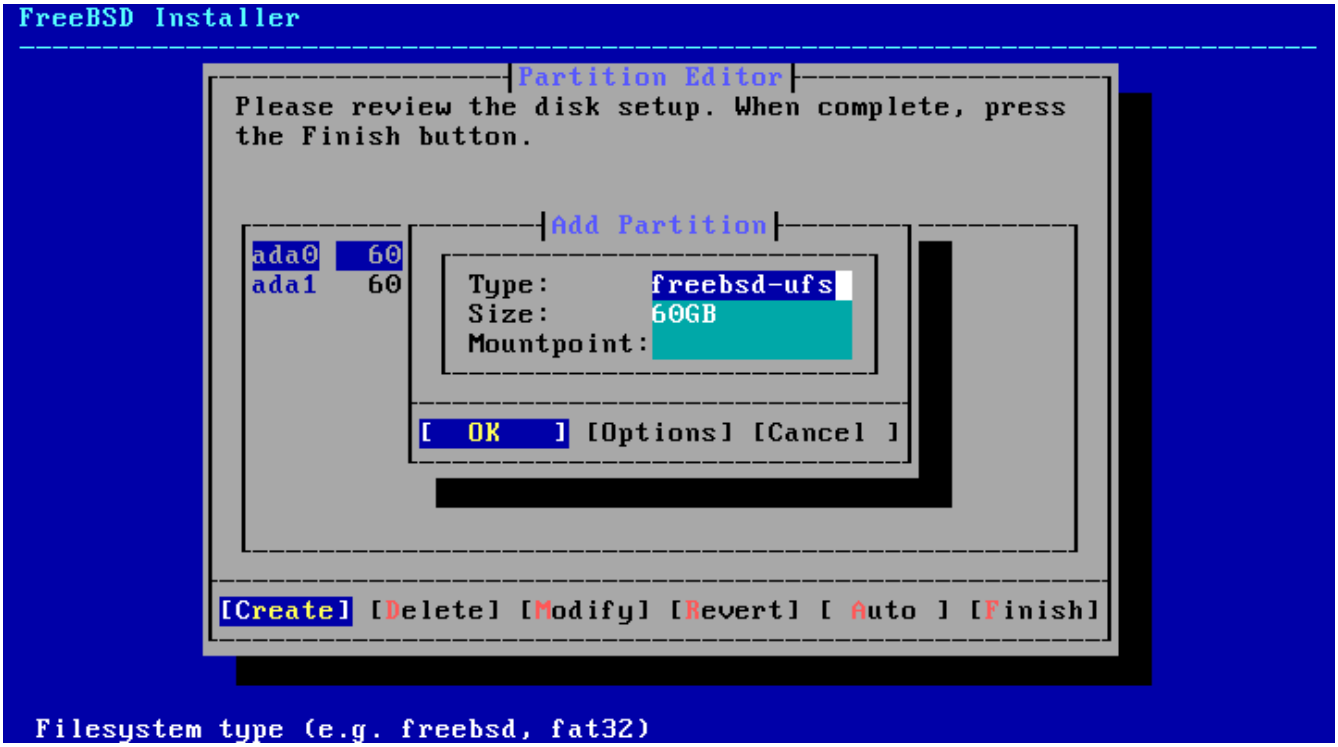


Рисунок 19. Ручное создание разделов

Стандартная установка FreeBSD с использованием GPT включает как минимум три раздела, включая либо UFS, либо ZFS:

- `freebsd-boot` или `efi` - Содержит загрузочный код FreeBSD.
- `freebsd-ufs` - Файловая система FreeBSD UFS.
- `freebsd-zfs` - Файловая система ZFS в FreeBSD. Дополнительная информация о ZFS доступна в Файловая система ZFS (ZFS).
- `freebsd-swap` - область подкачки FreeBSD.

Обратитесь к gpart(8) для описания доступных типов разделов GPT.

Можно создать несколько разделов файловой системы. Некоторые предпочитают традиционную схему с отдельными разделами для `/`, `/var`, `/tmp` и `/usr`.

Обратите внимание, что `/tmp` можно добавить позже как файловую систему в памяти (tmpfs(5)) на системах с достаточным объёмом оперативной памяти.

См. Создание традиционных разделов файловой системы с разделением для примера.

`Размер (Size)` может быть указан с общепринятыми сокращениями: *K* для килобайт, *M* для мегабайт или *G* для гигабайт.

Правильное выравнивание секторов обеспечивает наилучшую производительность, а создание разделов с размерами, кратными 4 КБ, помогает гарантировать выравнивание на дисках с секторами размером 512 байт или 4 КБ. Как правило, использование размеров разделов, кратных 1 МБ или 1 ГБ, — это самый простой способ убедиться, что каждый раздел начинается с адреса, кратного 4 КБ. Есть одно исключение: раздел *freebsd-boot* для загрузки из BIOS не должен превышать 512 КБ из-за ограничений старого загрузочного кода. Для загрузки с поддержкой UEFI такого ограничения нет.

Для раздела, который будет содержать файловую систему, требуется `точка монтирования (Mountpoint)`. Если создаётся только один раздел UFS, точкой монтирования должен быть `/`.

`Метка (Label)` — это имя, по которому раздел будет известен. Имена или номера дисков могут измениться, если диск подключён к другому контроллеру или порту, но метка раздела остаётся неизменной. Использование меток вместо имён дисков и номеров разделов в файлах, таких как `/etc/fstab`, делает систему более устойчивой к изменениям оборудования. Метки GPT отображаются в `/dev/gpt/` при подключении диска. Другие схемы разделения имеют свои возможности для меток, и их метки отображаются в разных каталогах в `/dev/`.

Используйте уникальные метки для каждого раздела, чтобы избежать конфликтов из-за одинаковых меток. Можно добавить несколько букв из имени компьютера, его назначения или местоположения. Например, используйте `labroot` или `rootfslab` для корневого раздела UFS на компьютере с именем `lab`.

Пример 1. Создание традиционных разделов файловой системы с разделением

Для традиционной схемы разделов, где `/`, `/var`, `/tmp` и `/usr` являются отдельными файловыми системами на своих разделах, создайте схему разделов GPT, затем создайте разделы, как показано ниже. Указанные размеры разделов типичны для целевого диска размером 20 ГБ. Если на целевом диске доступно больше места, могут быть полезны увеличенные разделы подкачки или `/var`. Метки, указанные здесь, имеют префикс `ex` (от "example"), но читателям следует использовать другие уникальные значения меток, как описано выше.

По умолчанию `gptboot` в FreeBSD ожидает, что первый UFS-раздел будет разделом `/`.

Тип раздела	Размер	Точка монтирования	Label
freebsd-boot	512K		
freebsd-ufs	2G	/	exrootfs
freebsd-swap	4G		exswap
freebsd-ufs	2G	/var	exvarfs
freebsd-ufs	1G	/tmp	extmpfs
freebsd-ufs	принять значение по умолчанию (оставшаяся часть диска)	/usr	exusrfs

После создания пользовательских разделов выберите **[Завершить (Finish)]**, чтобы продолжить установку и перейти к разделу Загрузка файлов дистрибутива.

2.6.4. Разметка диска с использованием Root-on-ZFS с помощью мастера

Этот режим разметки работает только с целыми дисками и полностью сотрёт все данные на диске. Основное меню настройки ZFS предоставляет несколько вариантов управления созданием пула.

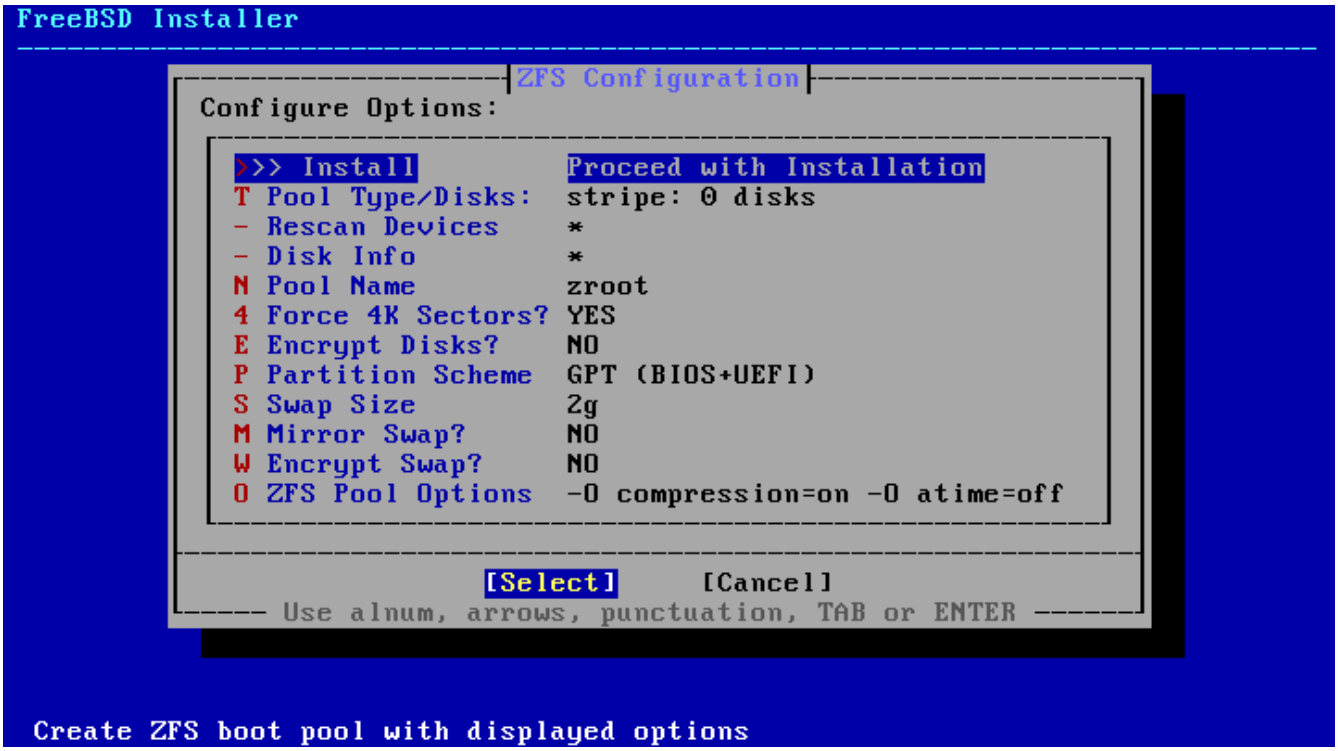


Рисунок 20. Меню разметки ZFS

Вот краткое описание пунктов этого меню:

- `Установить (Install)` - Приступить к установке с выбранными параметрами.
- `Тип пула/диски (Pool Type/Disks)` - Настройте `Тип пула` и диск(и), которые будут составлять пул. Автоматический установщик ZFS в настоящее время поддерживает создание только одного vdev верхнего уровня, за исключением режима stripe. Для создания более сложных пулов воспользуйтесь инструкциями в Разметка в режиме оболочки, чтобы создать пул.
- `Сканировать диски (Rescan Devices)` - Обновить список доступных дисков.
- `Информация о диске (Disk Info)` - Это меню позволяет просматривать информацию о каждом диске, включая таблицу разделов и другие данные, такие как модель устройства и серийный номер, если они доступны.
- `Имя пула (Pool Name)` - Укажите имя пула. По умолчанию используется имя *zroot*.
- `Принудительно использовать секторы 4K? (Force 4K Sectors?)` — Принудительное использование секторов размером 4K. По умолчанию установщик автоматически создаёт разделы, выровненные по границам 4K, и принудительно устанавливает использование секторов 4K в ZFS. Это безопасно даже для дисков с секторами размером 512 байт и имеет дополнительное преимущество: пулы, созданные на дисках с 512-байтными секторами, смогут в будущем работать с дисками, имеющими секторы 4K, — как для расширения хранилища, так и для замены вышедших из строя дисков. Нажмите `Enter`, чтобы выбрать активацию или отказ от неё.
- `Шифровать диски (Encrypt Disks)?` - Шифрование дисков позволяет пользователю зашифровать диски с помощью GELI. Дополнительная информация о шифровании дисков доступна в “Шифрование дисков с помощью geli”. Нажмите клавишу `Enter`, чтобы выбрать, активировать его или нет.
- `Схема разделов (Partition Scheme)` - Выберите схему разделов. GPT рекомендуется в большинстве случаев. Нажмите клавишу `Enter` для выбора между различными вариантами.
- `Размер подкачки (Swap Size)` - Установите объём области подкачки.
- `Зеркалировать подкачку (Mirror Swap)?` - Определяет, нужно ли зеркалировать раздел подкачки между дисками. Учтите, что включение зеркалирования swap может нарушить работу дампов аварийных завершений. Нажмите `Enter`, чтобы активировать или отклонить этот параметр.
- `Шифровать раздел подкачки (Encrypt Swap)?` - Определяет, следует ли шифровать раздел подкачки. При каждом запуске системы раздел подкачки будет зашифрован временным ключом, который удаляется после перезагрузки. Нажмите `Enter`, чтобы выбрать, активировать эту функцию или нет. Дополнительная информация о шифровании раздела подкачки приведена в Шифрование раздела подкачки.
- `ZFS Pool Options` (Параметры пула ZFS) — задаёт дополнительные свойства пула, применяемые при создании корневого пула ZFS. Дополнительная информация об этих параметрах приведена в zpool-create(8) и zpoolprops(7).

Выберите `T`, чтобы настроить `Тип пула (Pool Type)` и диски, которые будут входить в пул.

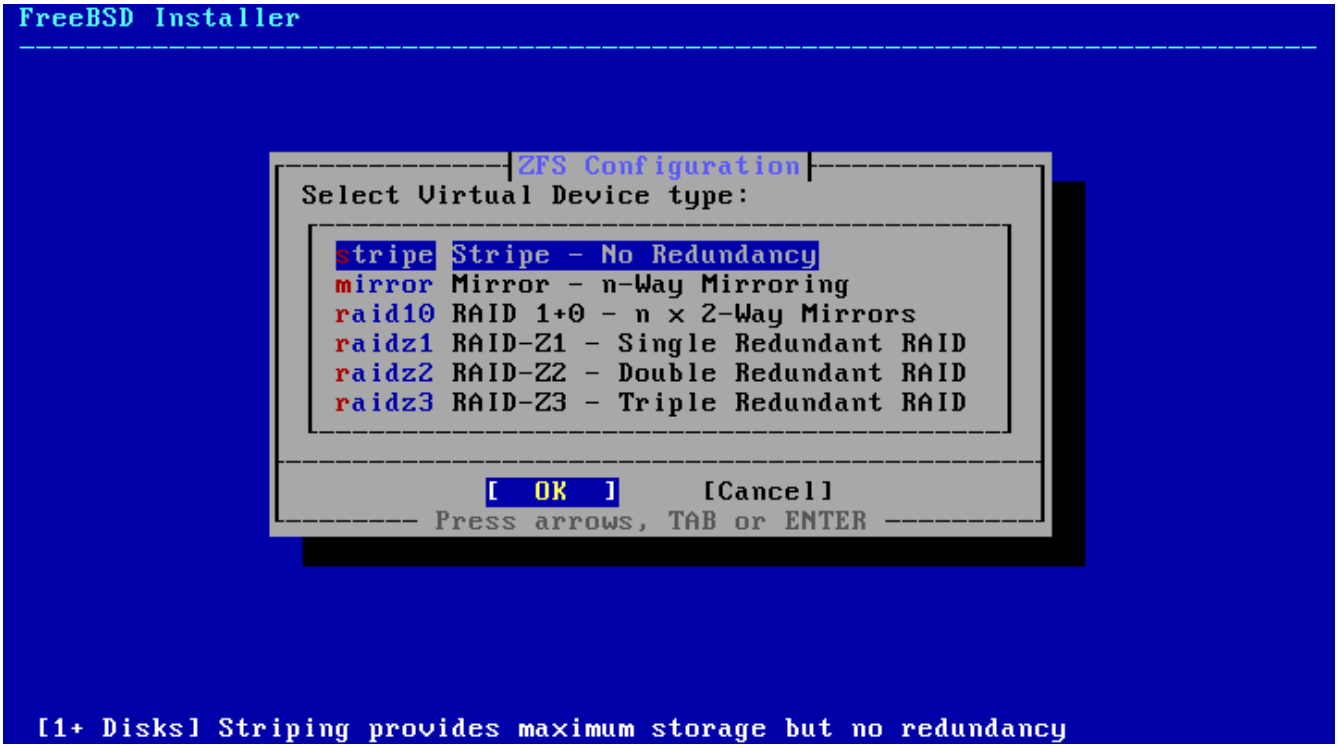


Рисунок 21. Тип пула ZFS

Вот сводка по `Типу пула`, который можно выбрать в этом меню:

- `stripe` - Чередование (striping) обеспечивает максимальный объём хранилища из всех подключённых устройств, но не предоставляет избыточности. Если выйдет из строя хотя бы один диск, данные в пуле будут потеряны безвозвратно.
- `mirror` - Зеркалирование сохраняет полную копию всех данных на каждом диске. Зеркалирование обеспечивает высокую производительность чтения, поскольку данные считываются со всех дисков параллельно. Производительность записи ниже, так как данные должны быть записаны на все диски в пуле. Допускает отказ всех дисков, кроме одного. Для этого варианта требуется как минимум два диска.
- `raid10` - Чередующиеся зеркала. Обеспечивает наилучшую производительность, но наименьший объём хранилища. Для этого варианта требуется четное количество дисков, минимум четыре.
- `raidz1` - RAID с одинарной избыточностью. Позволяет одновременный отказ одного диска. Для этого варианта требуется как минимум три диска.
- `raidz2` - Двухдисковая избыточная RAID. Позволяет одновременно отказать двум дискам. Для этой конфигурации требуется как минимум четыре диска.
- `raidz3` - Трехкратно избыточный RAID. Позволяет одновременно отказать трем дискам. Для этого варианта требуется как минимум пять дисков.

После выбора `Типа пула` отображается список доступных дисков, и пользователю предлагается выбрать один или несколько дисков для создания пула. Затем конфигурация проверяется, чтобы убедиться, что выбрано достаточное количество дисков. Если проверка не пройдена, выберите **[<Изменить выбор>]**, чтобы вернуться к списку дисков, или **[<Назад>]**, чтобы изменить `Тип пула`.

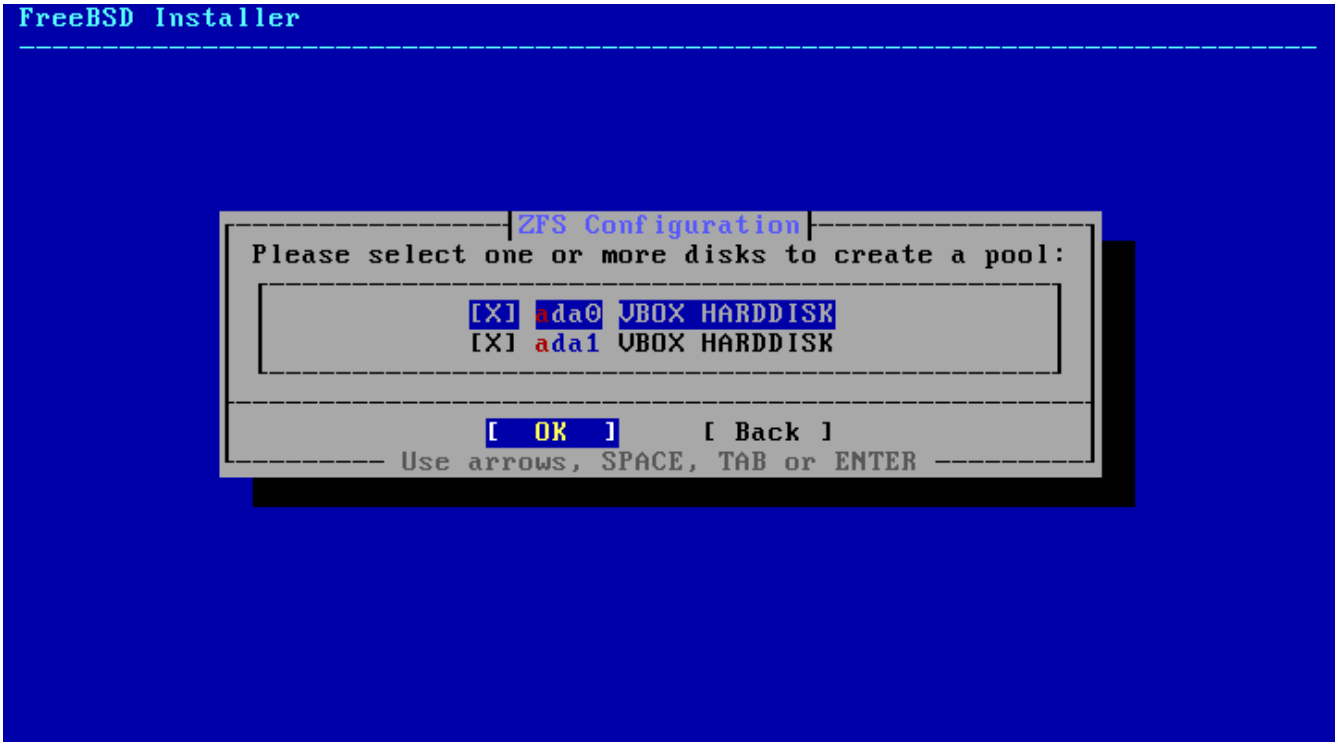


Рисунок 22. Выбор диска

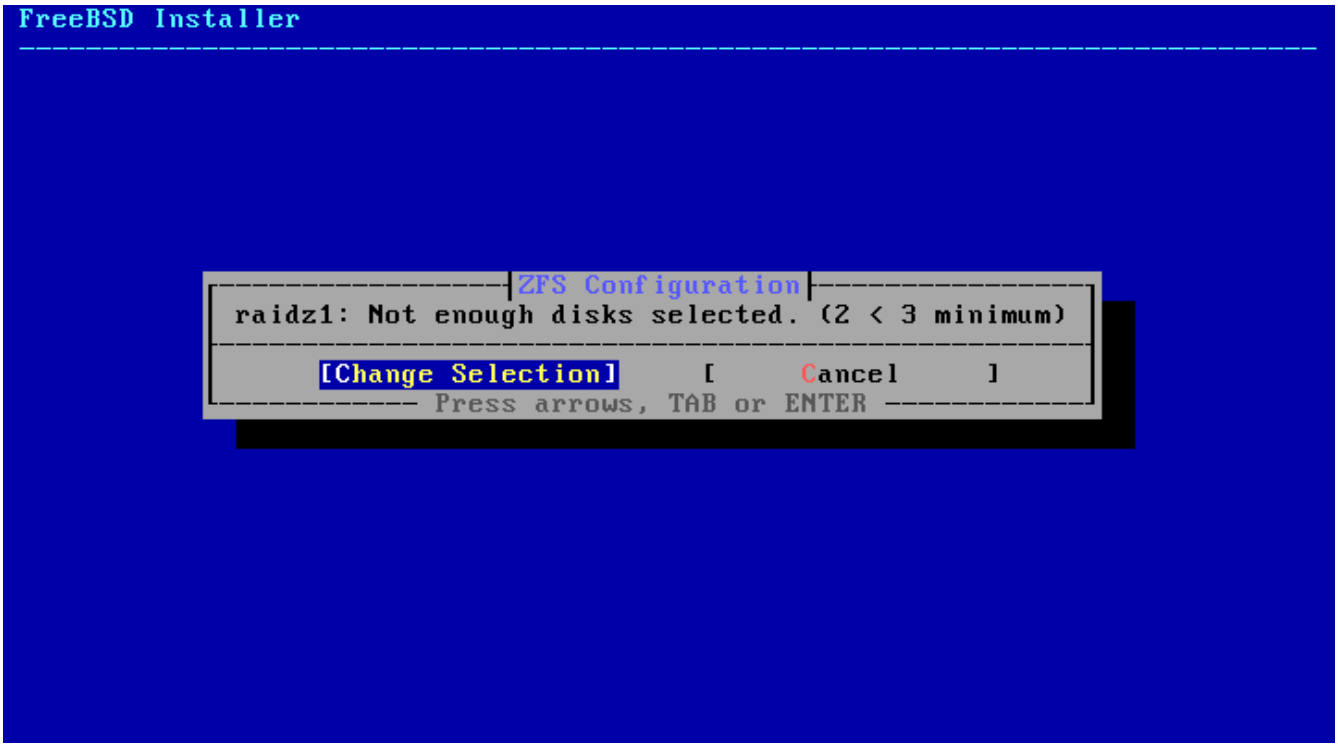


Рисунок 23. Неверный выбор

Если один или несколько дисков отсутствуют в списке или если диски были подключены после запуска установщика, выберите **[- Повторное сканирование устройств (Rescan Devices)]**, чтобы обновить список доступных дисков.

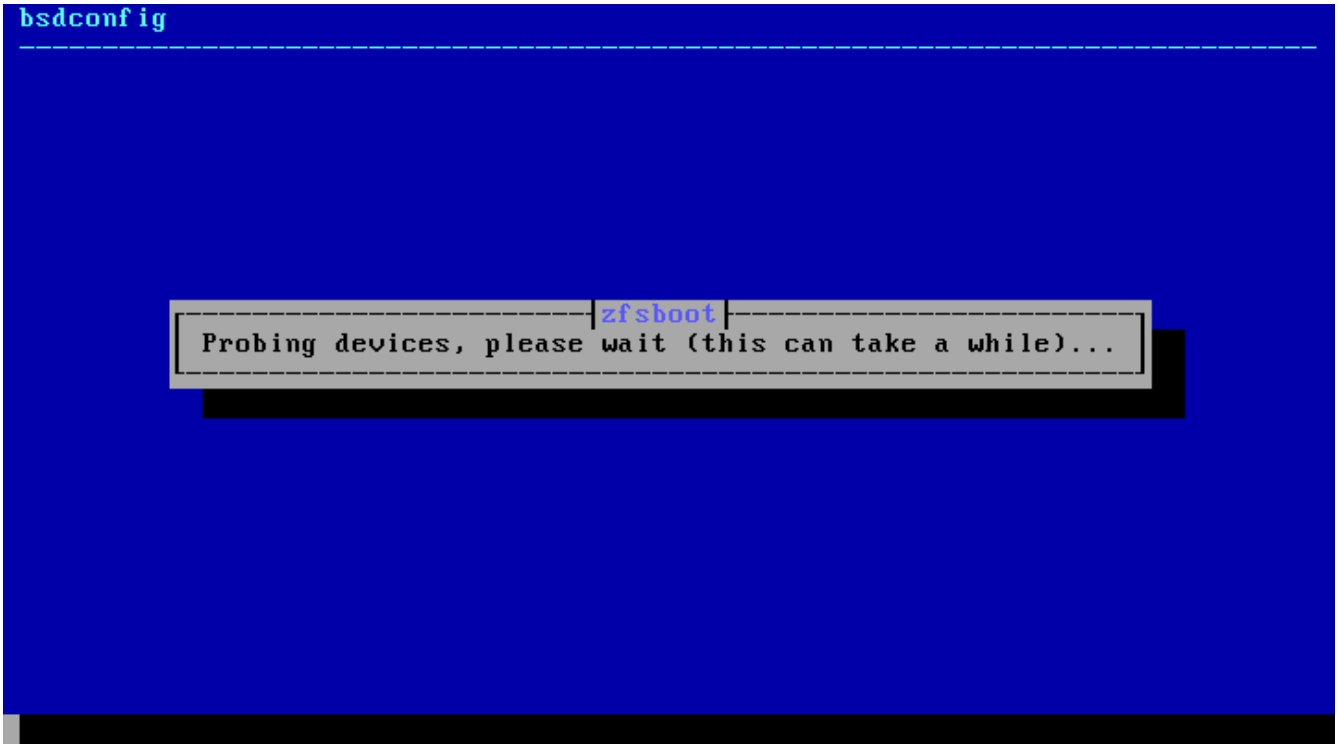


Рисунок 24. Пересканировать Устройства

Чтобы случайно не стереть не тот диск, можно использовать меню **[- Disk Info]** для просмотра информации о каждом диске, включая таблицу разделов и другие данные, такие как модель устройства и серийный номер, если они доступны.

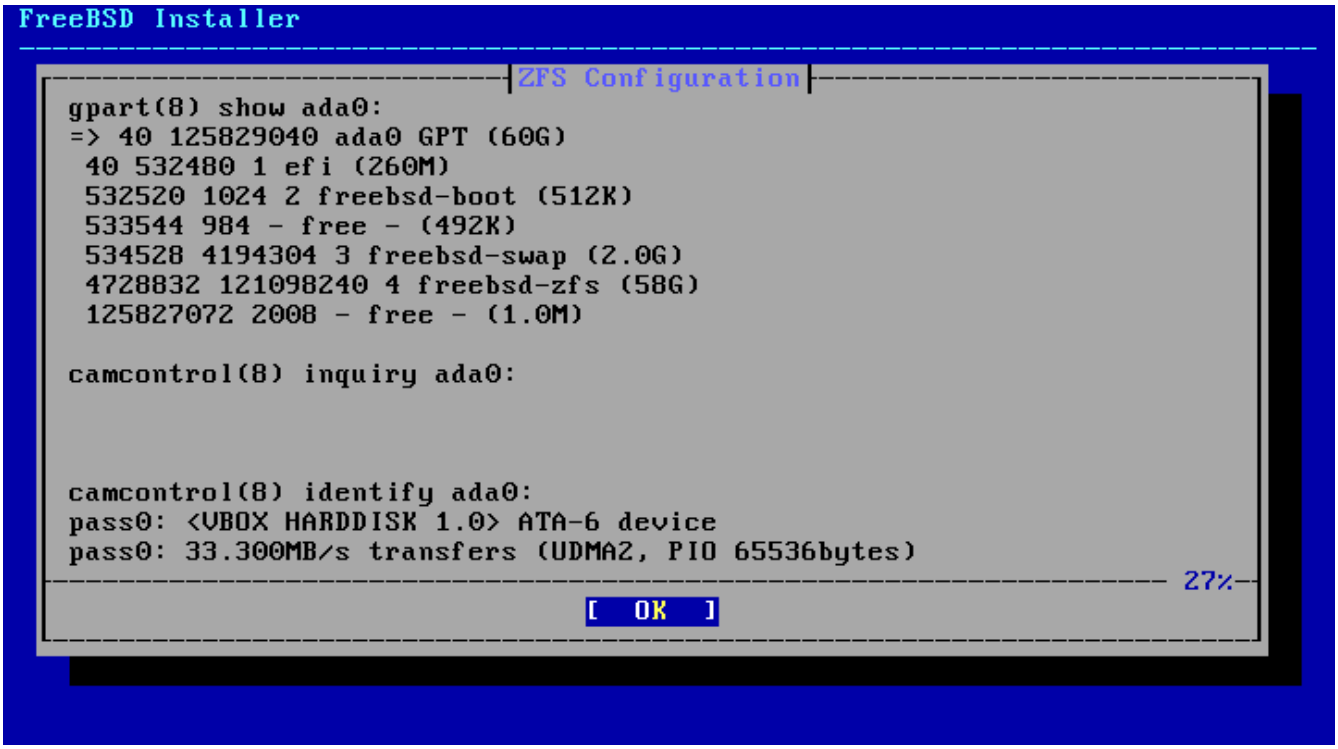


Рисунок 25. Анализ диска

Выберите `N` для настройки `Имени пула (Pool Name)`. Введите желаемое имя, затем выберите **[<OK>]**, чтобы установить его, или **[<Отмена>]**, чтобы вернуться в главное меню и оставить имя по умолчанию.

Рисунок 26. Имя пула

Выберите `S`, чтобы установить размер раздела подкачки. Введите желаемый размер раздела подкачки, затем выберите **[<OK>]** для подтверждения или **[<Cancel>]**, чтобы вернуться в главное меню и оставить значение по умолчанию.

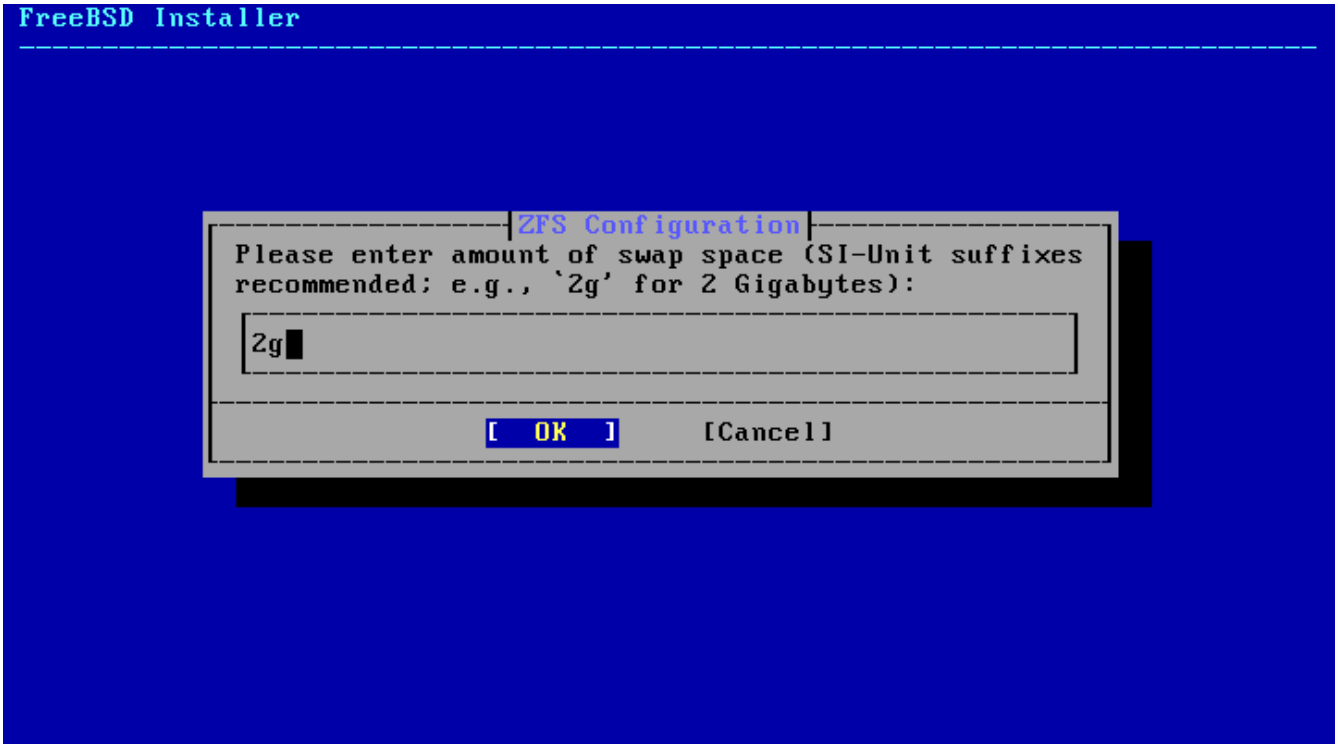


Рисунок 27. Размер раздела подкачки

После установки всех необходимых значений выберите в верхней части меню опцию **[>>> Установить]**. Установщик предоставит последнюю возможность отменить процесс перед уничтожением содержимого выбранных дисков для создания пула ZFS.

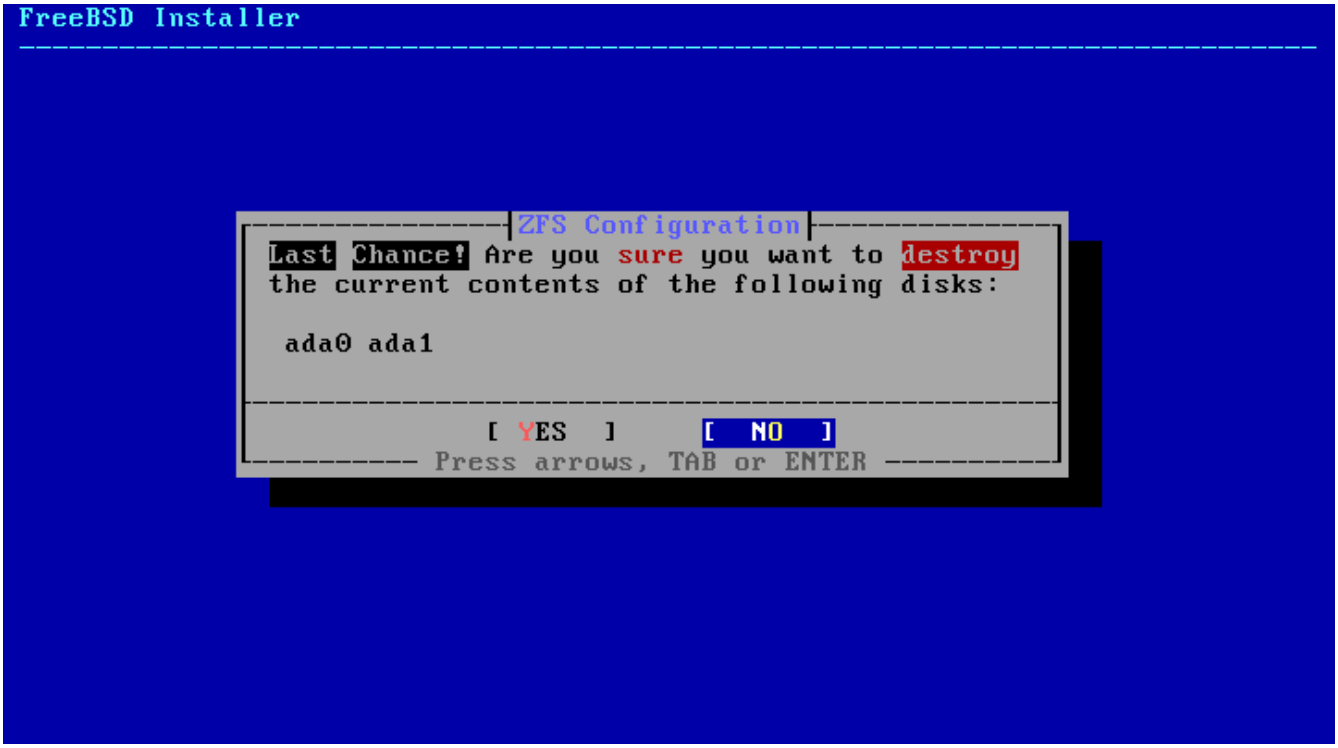


Рисунок 28. Последний шанс

Если включено шифрование дисков GELI, установщик дважды запросит парольную фразу, используемую для шифрования дисков. Затем начнётся инициализация шифрования.

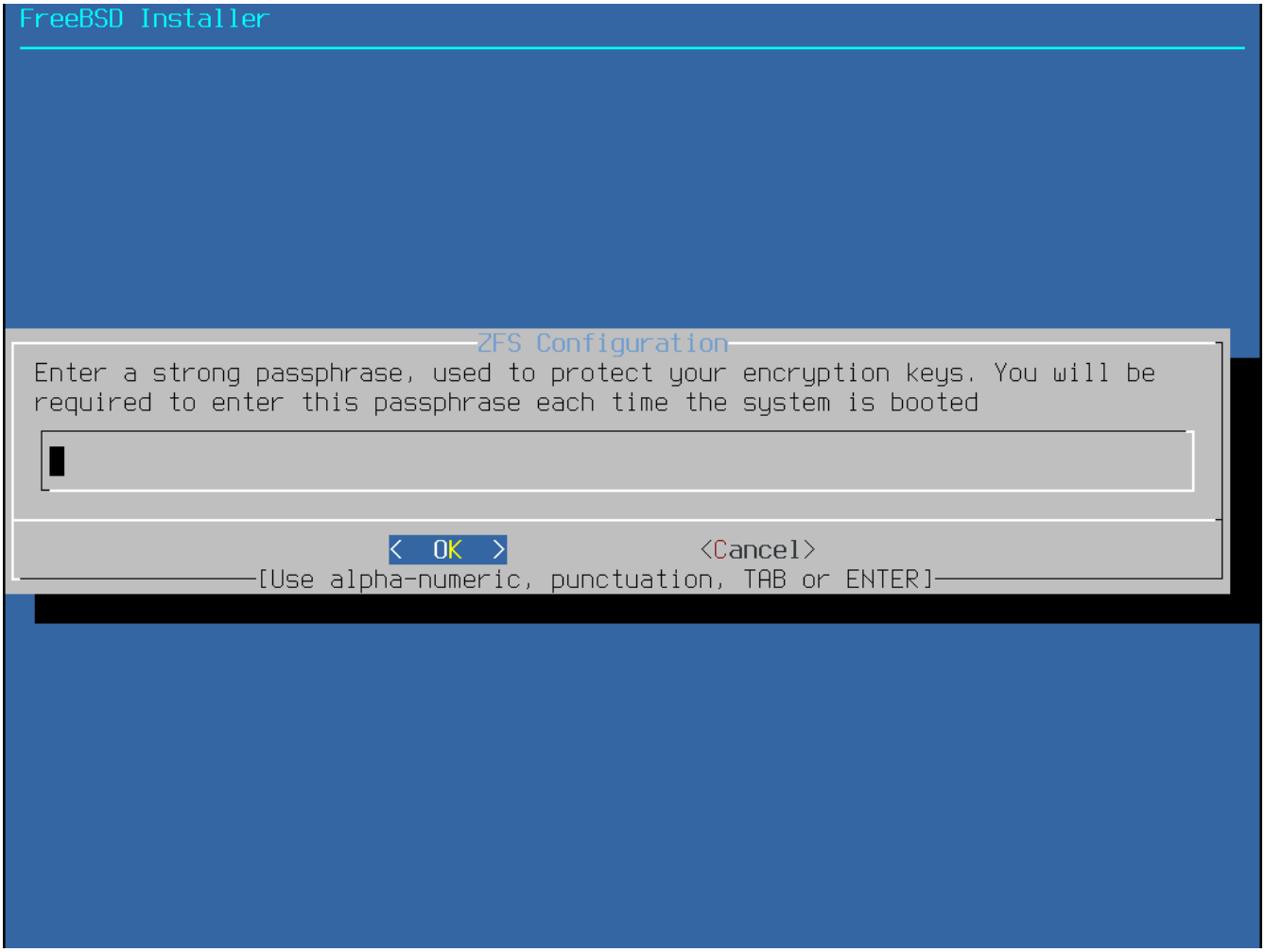


Рисунок 29. Пароль для шифрования диска

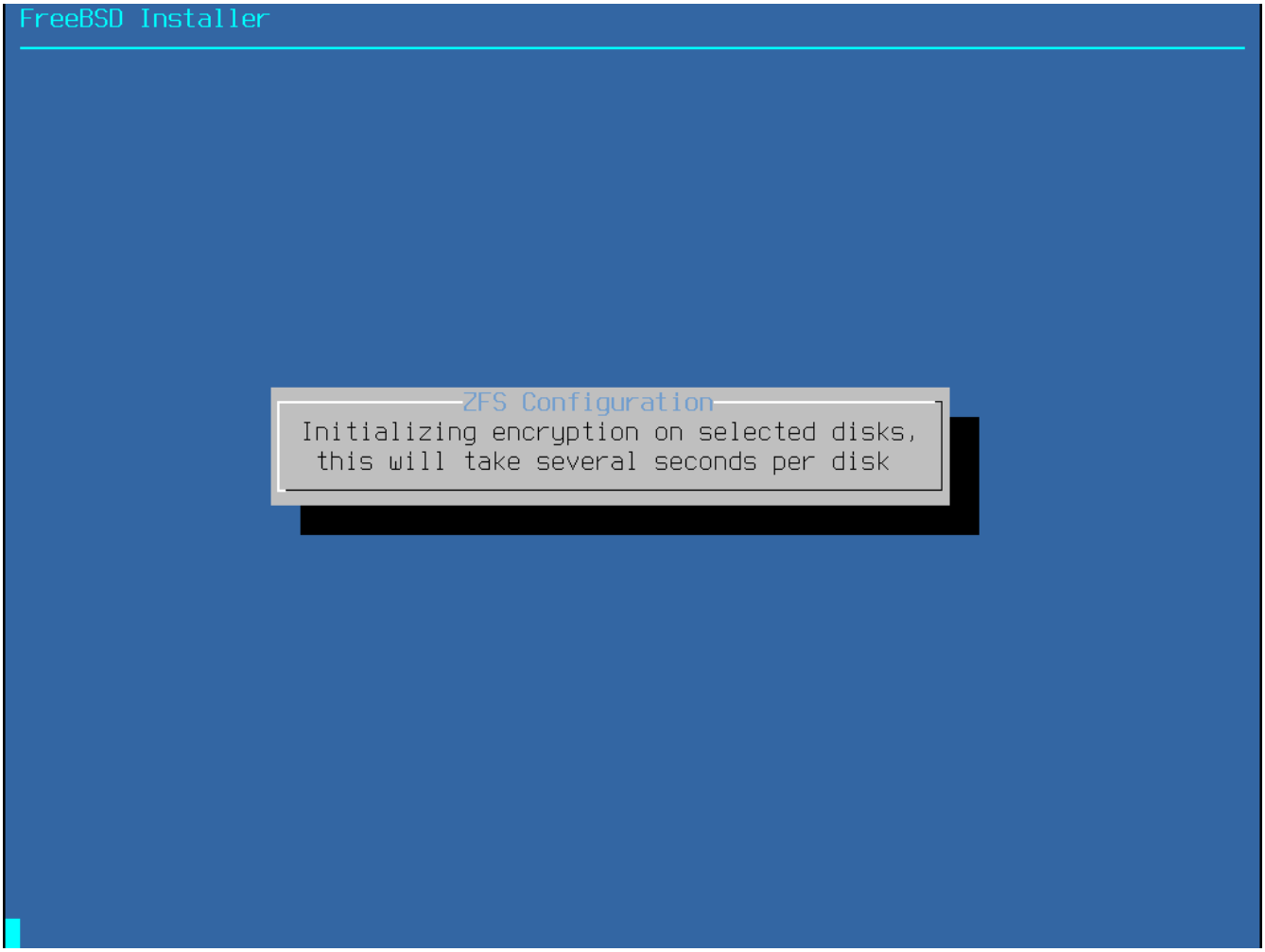


Рисунок 30. Инициализация шифрования

Установка затем продолжается в обычном режиме. Для продолжения установки перейдите к разделу Получение файлов дистрибутива.

2.6.5. Режим разметки разделов в оболочке

При создании сложных установок меню разметки `bsdinstall` может не предоставлять необходимого уровня гибкости. Опытные пользователи могут выбрать опцию **[Shell]** в меню разметки, чтобы вручную разметить диски, создать файловую систему(ы), заполнить `/tmp/bsdinstall_etc/fstab` и смонтировать файловые системы в `/mnt`. После выполнения этих действий введите `exit`, чтобы вернуться в `bsdinstall` и продолжить установку.

2.7. Загрузка файлов дистрибутива

Время установки может варьироваться в зависимости от выбранных дистрибутивов, носителя установки и скорости компьютера. Серия сообщений будет показывать ход выполнения.

Сначала установщик форматирует выбранный диск(и) и инициализирует разделы. Затем, в случае `bootonly media` или `mini memstick`, он загружает выбранные компоненты:

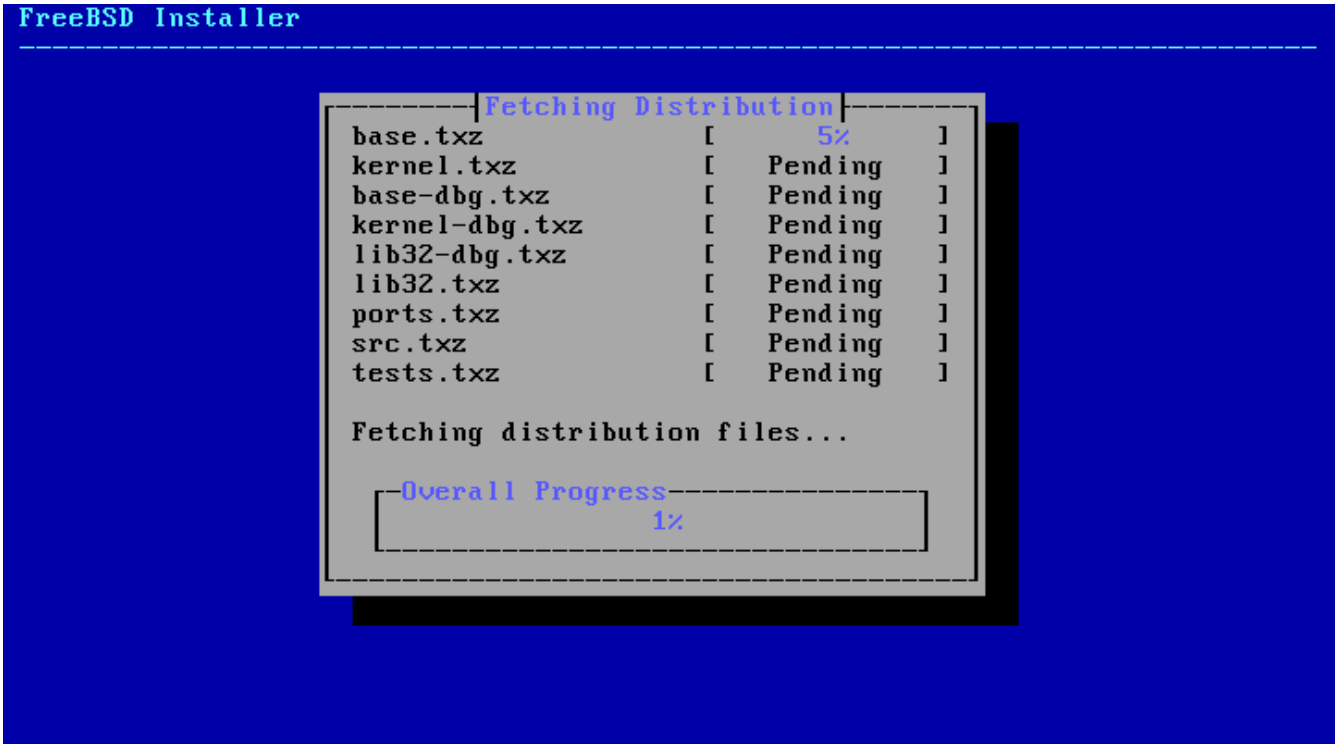


Рисунок 31. Загрузка файлов дистрибутива

Затем проверяется целостность файлов дистрибутива, чтобы убедиться, что они не были повреждены при загрузке или неправильно прочитаны с установочного носителя:

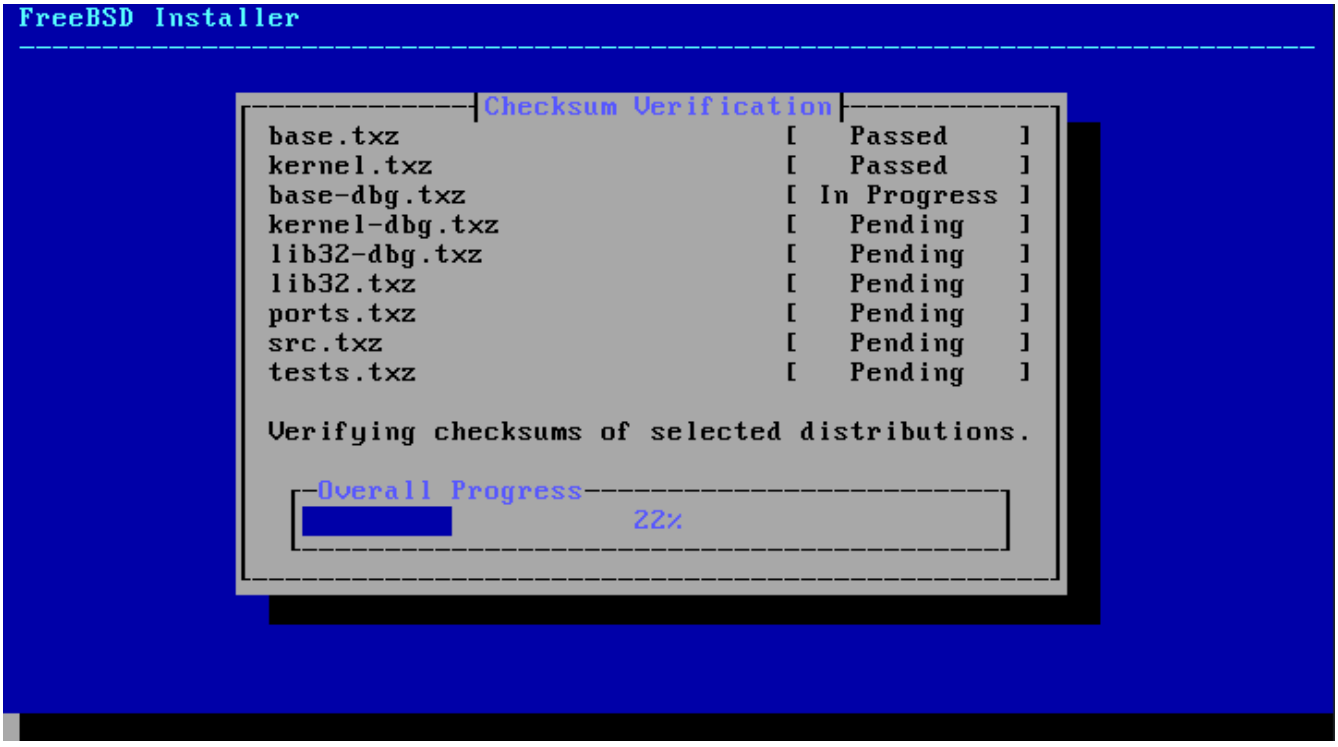


Рисунок 32. Проверка файлов дистрибутива

Наконец, проверенные файлы дистрибутива извлекаются на диск:

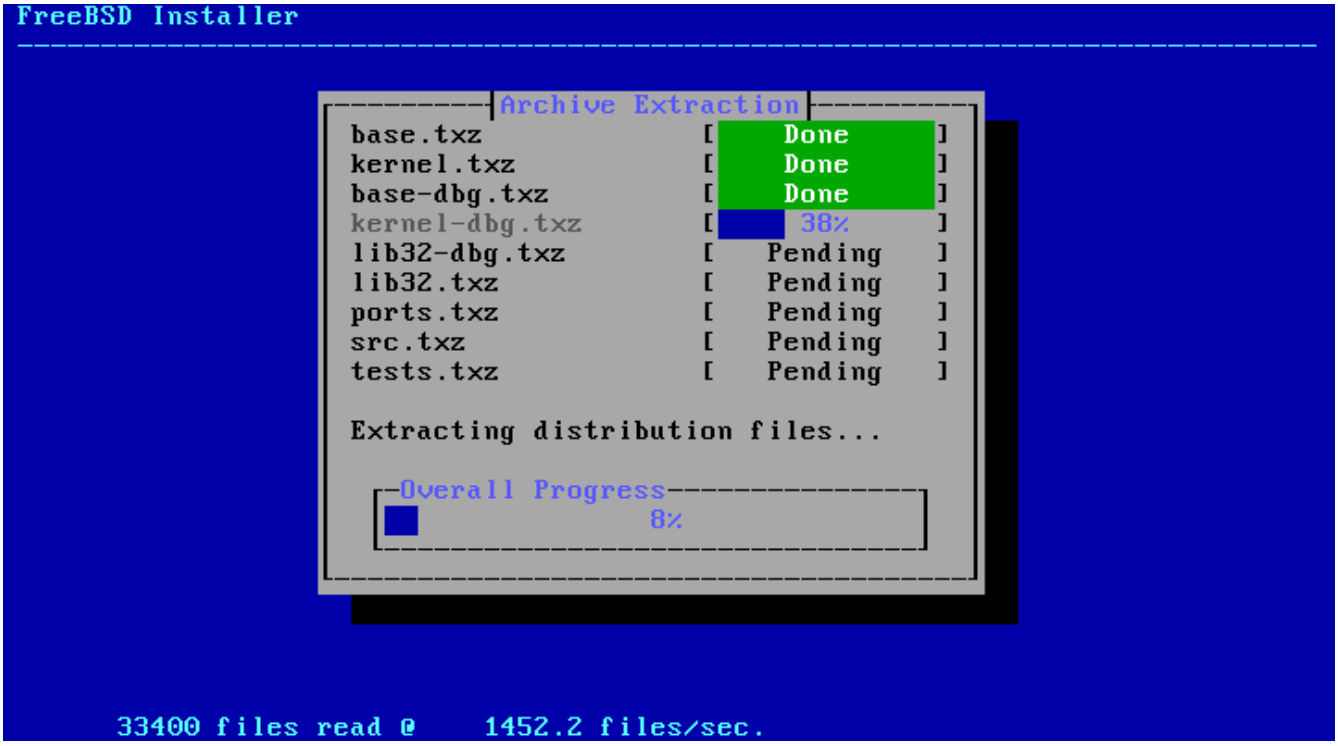


Рисунок 33. Извлечение файлов дистрибутива

После извлечения всех запрошенных файлов дистрибутива `bsdinstall` отображает первый экран настройки после установки. Доступные параметры пост-конфигурации описаны в следующем разделе.

2.8. Сетевые интерфейсы, учётные записи, часовой пояс, службы и защита

2.8.1. Установка пароля `root`

Сначала необходимо установить пароль `root`. При вводе пароля символы не отображаются на экране. Пароль нужно ввести дважды, чтобы избежать ошибок при наборе.

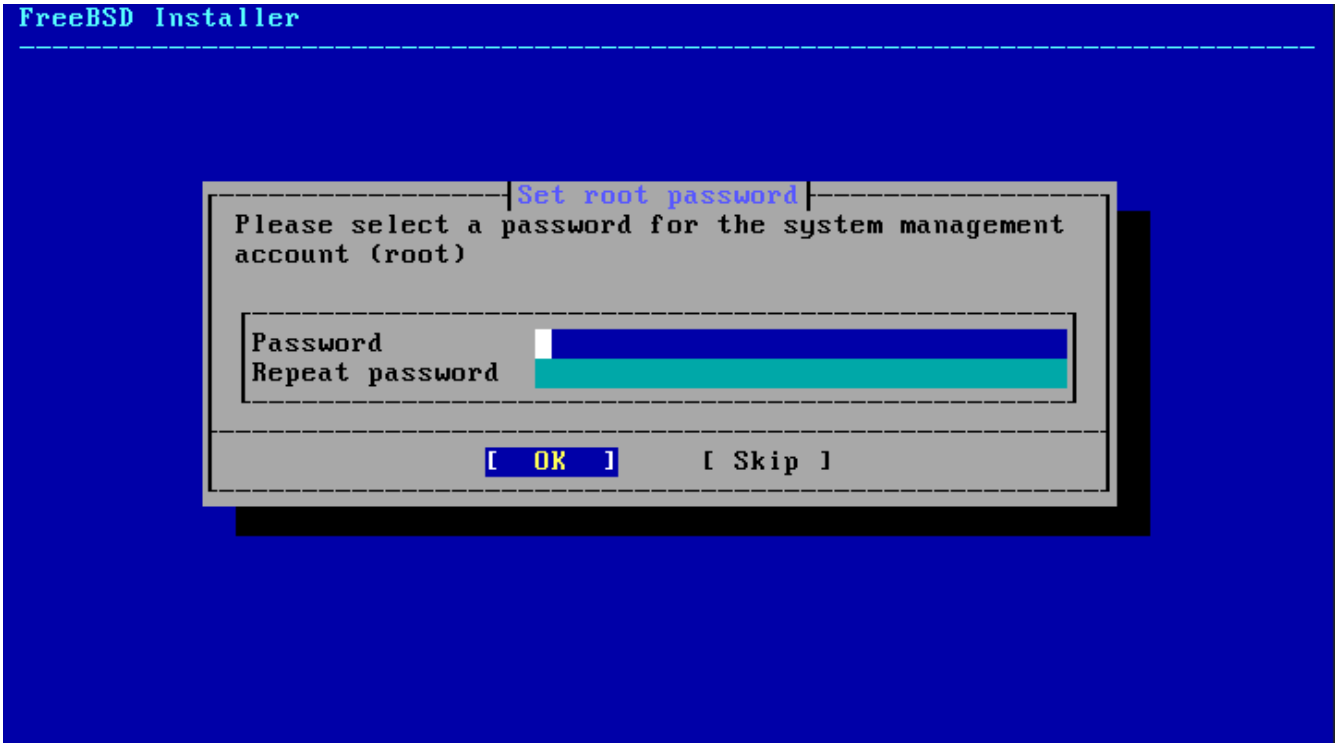


Рисунок 34. Установка пароля `root`

2.8.2. Настройка сетевых интерфейсов

Далее приведён список сетевых интерфейсов, обнаруженных на компьютере. Выберите интерфейс для настройки.

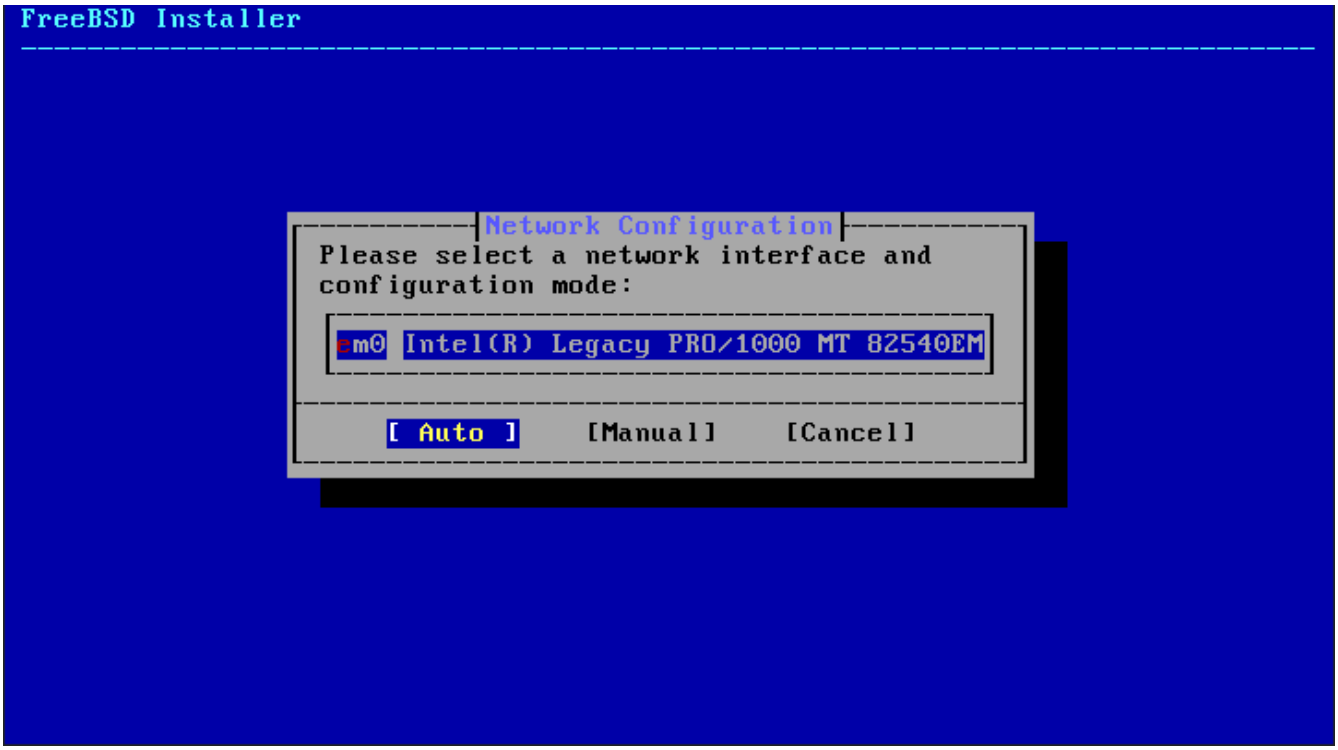


Рисунок 35. Выберите сетевой интерфейс

Если выбран интерфейс Ethernet, установщик перейдет сразу к меню, показанному в Выбор IPv4-сети. Если выбран беспроводной сетевой интерфейс, система выполнит поиск точек доступа:

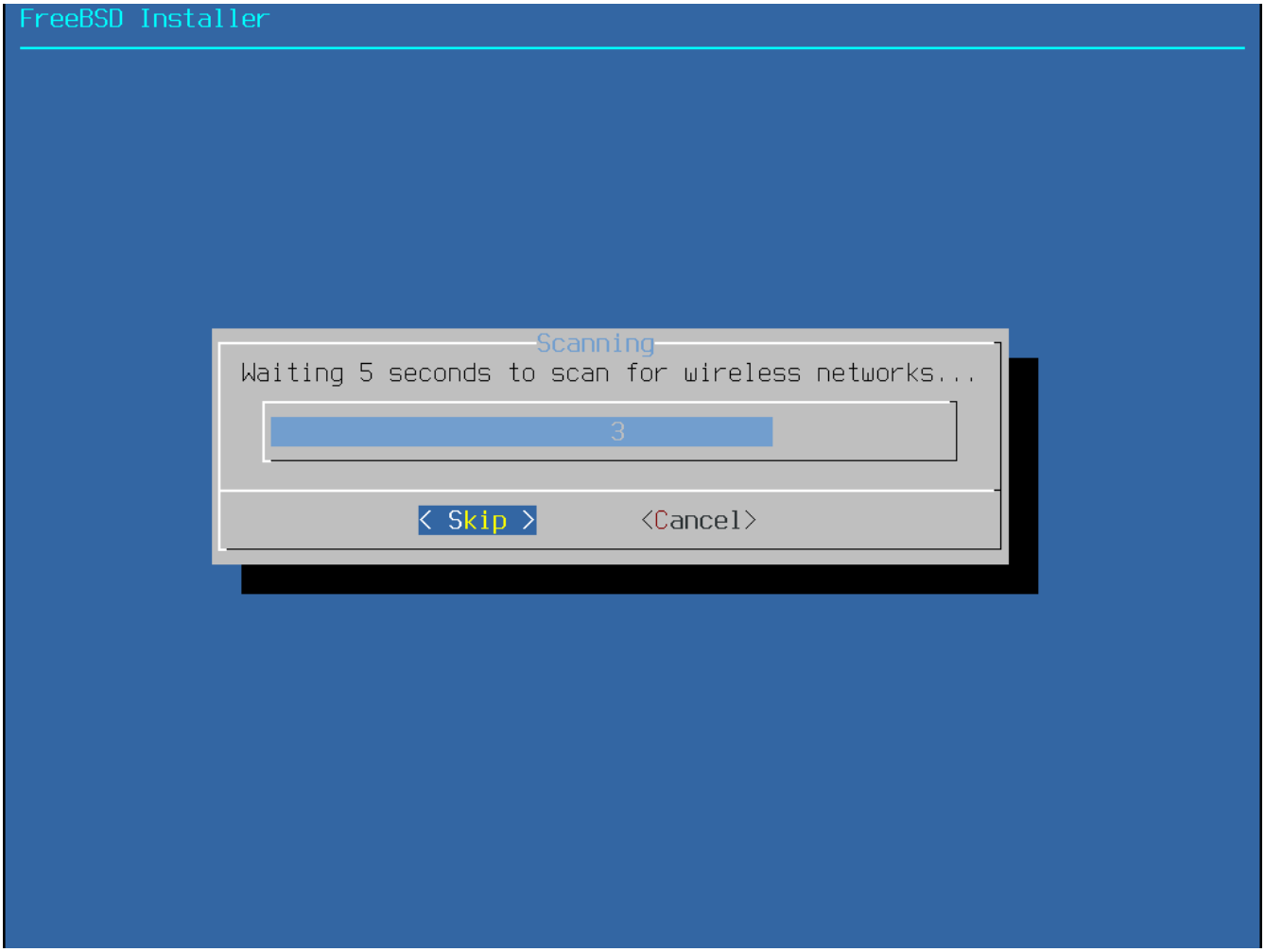


Рисунок 36. Сканирование беспроводных точек доступа

Беспроводные сети идентифицируются по имени Service Set Identifier (SSID) — короткому уникальному названию каждой сети. Обнаруженные при сканировании SSID перечислены ниже, вместе с описанием доступных типов шифрования для каждой сети. Если нужный SSID не отображается в списке, выберите **[Повторить сканирование (Rescan)]**, чтобы выполнить сканирование снова. Если нужная сеть по-прежнему не отображается, проверьте подключение антенны или попробуйте переместить компьютер ближе к точке доступа. После каждого изменения выполняйте повторное сканирование.

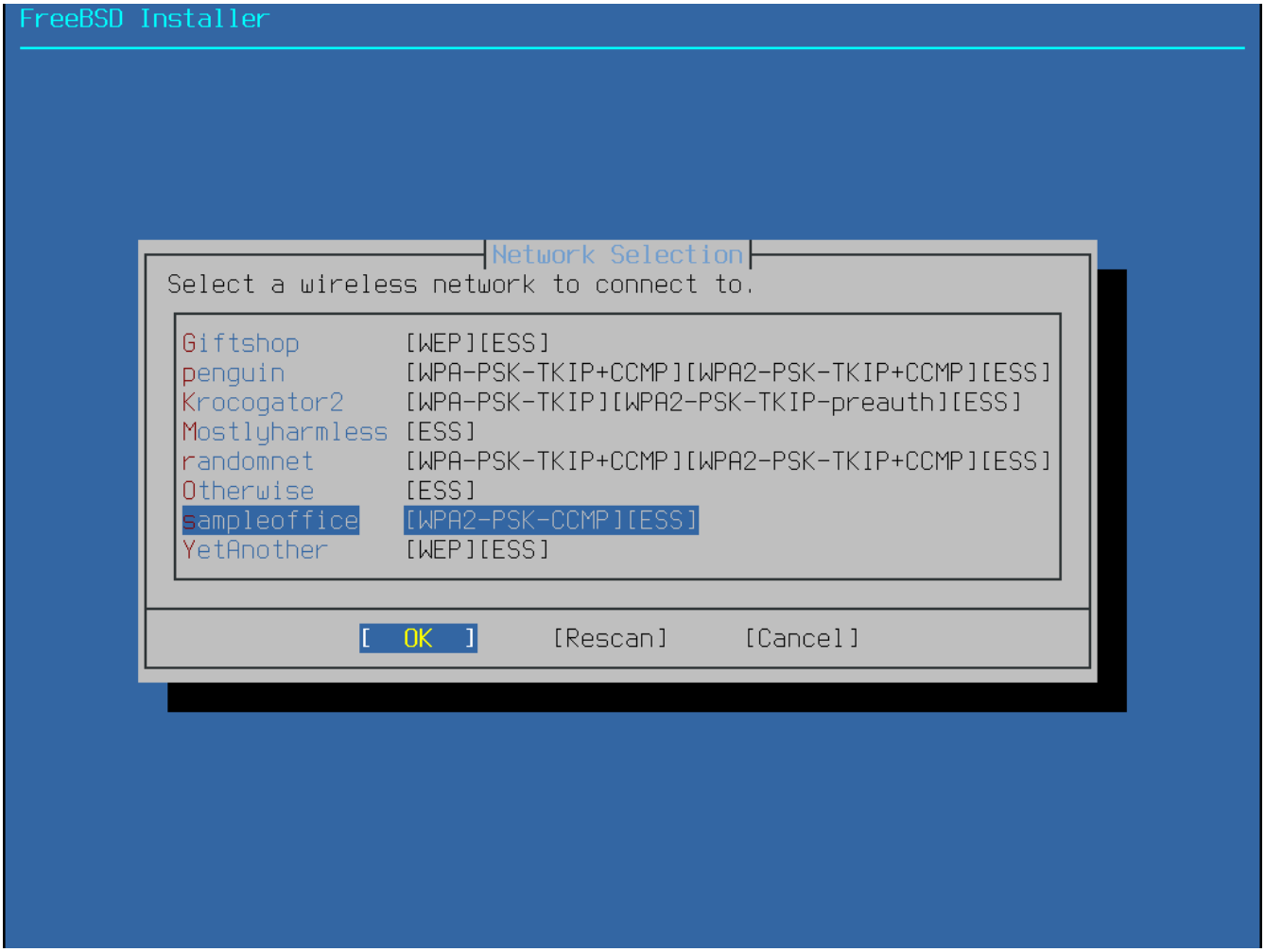


Рисунок 37. Выбор беспроводной сети

Далее введите информацию для шифрования, чтобы подключиться к выбранной беспроводной сети. Настоятельно рекомендуется использовать шифрование WPA2 вместо устаревших типов, таких как WEP, которые обеспечивают низкий уровень безопасности. Если сеть использует WPA2, введите пароль, также известный как Pre-Shared Key (PSK). В целях безопасности вводимые символы отображаются звездочками.

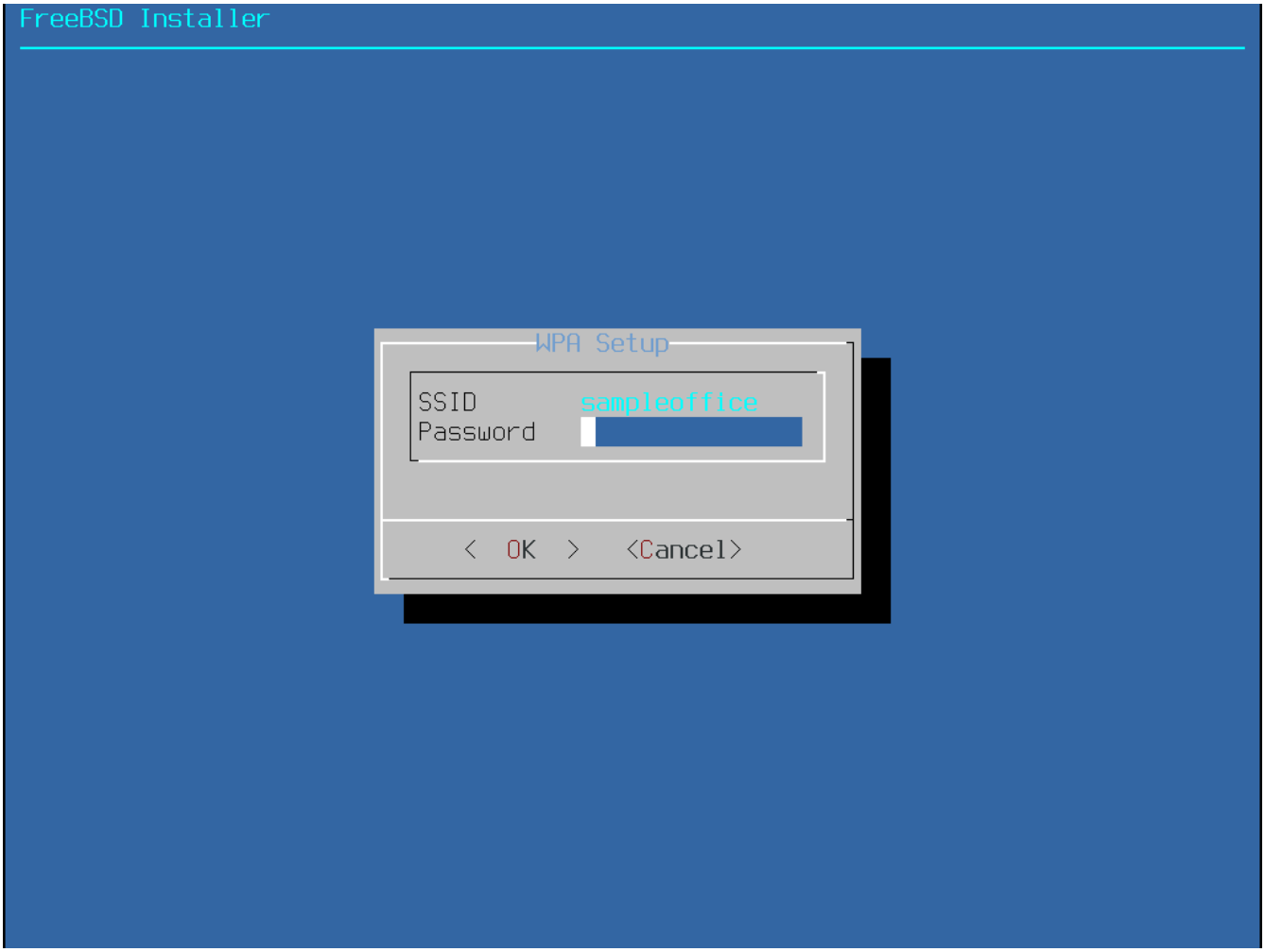


Рисунок 38. Настройка WPA2

Затем выберите, нужно ли настраивать IPv4-адрес на Ethernet или беспроводном интерфейсе:

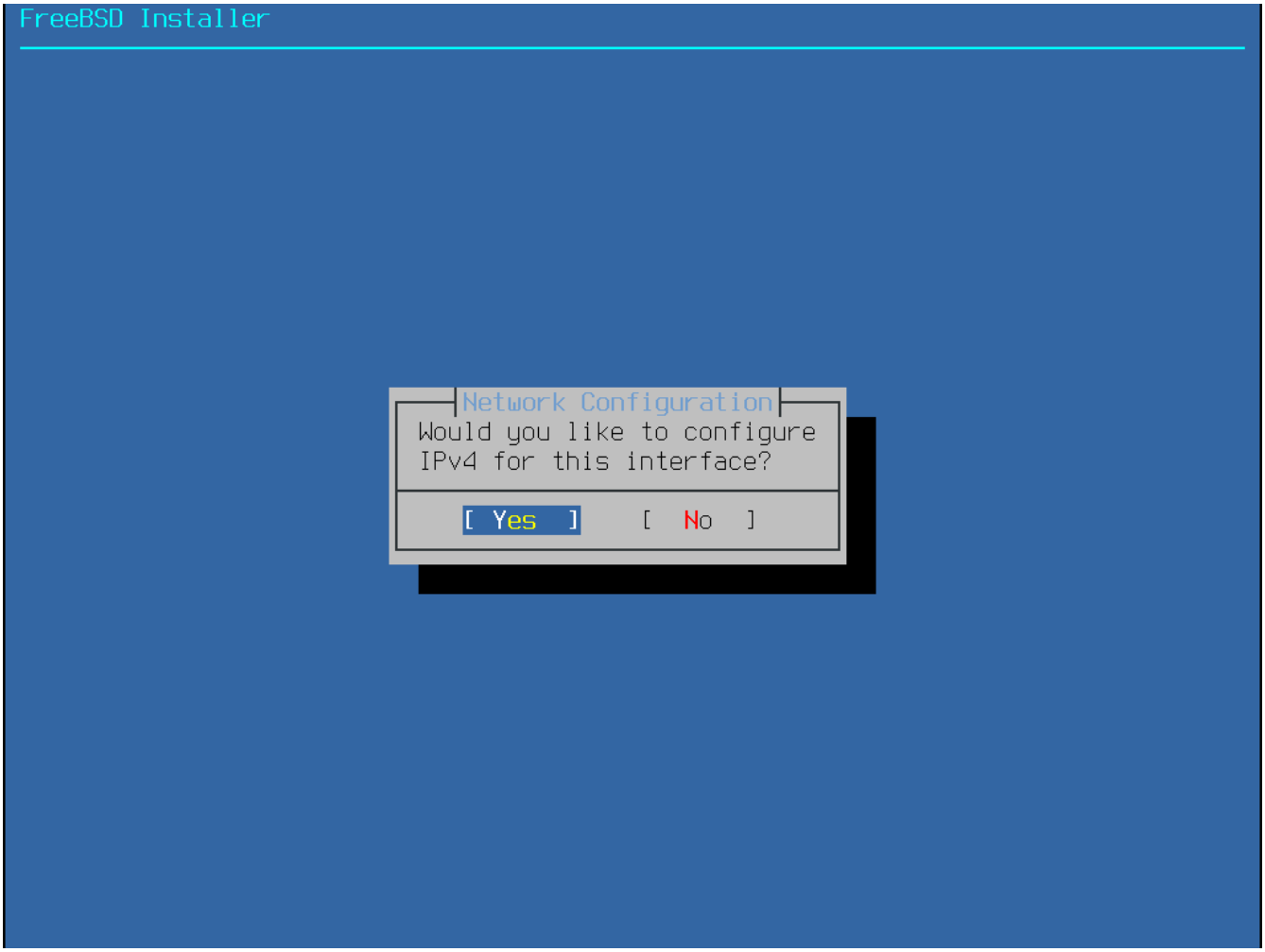


Рисунок 39. Выберите сеть IPv4

Существует два способа настройки IPv4. DHCP автоматически правильно настроит сетевой интерфейс и должен использоваться, если в сети есть DHCP-сервер. В противном случае, информацию об адресации необходимо ввести вручную как статическую конфигурацию.

Не вводите произвольные сетевые настройки, так как это не сработает. Если DHCP-сервер недоступен, получите информацию, указанную в Необходимая информация о сети, у администратора сети или интернет-провайдера.

Если доступен DHCP-сервер, выберите **[Да]** в следующем меню для автоматической настройки сетевого интерфейса. Установщик может показаться зависшим на минуту или около того, пока он находит DHCP-сервер и получает адресную информацию для системы.

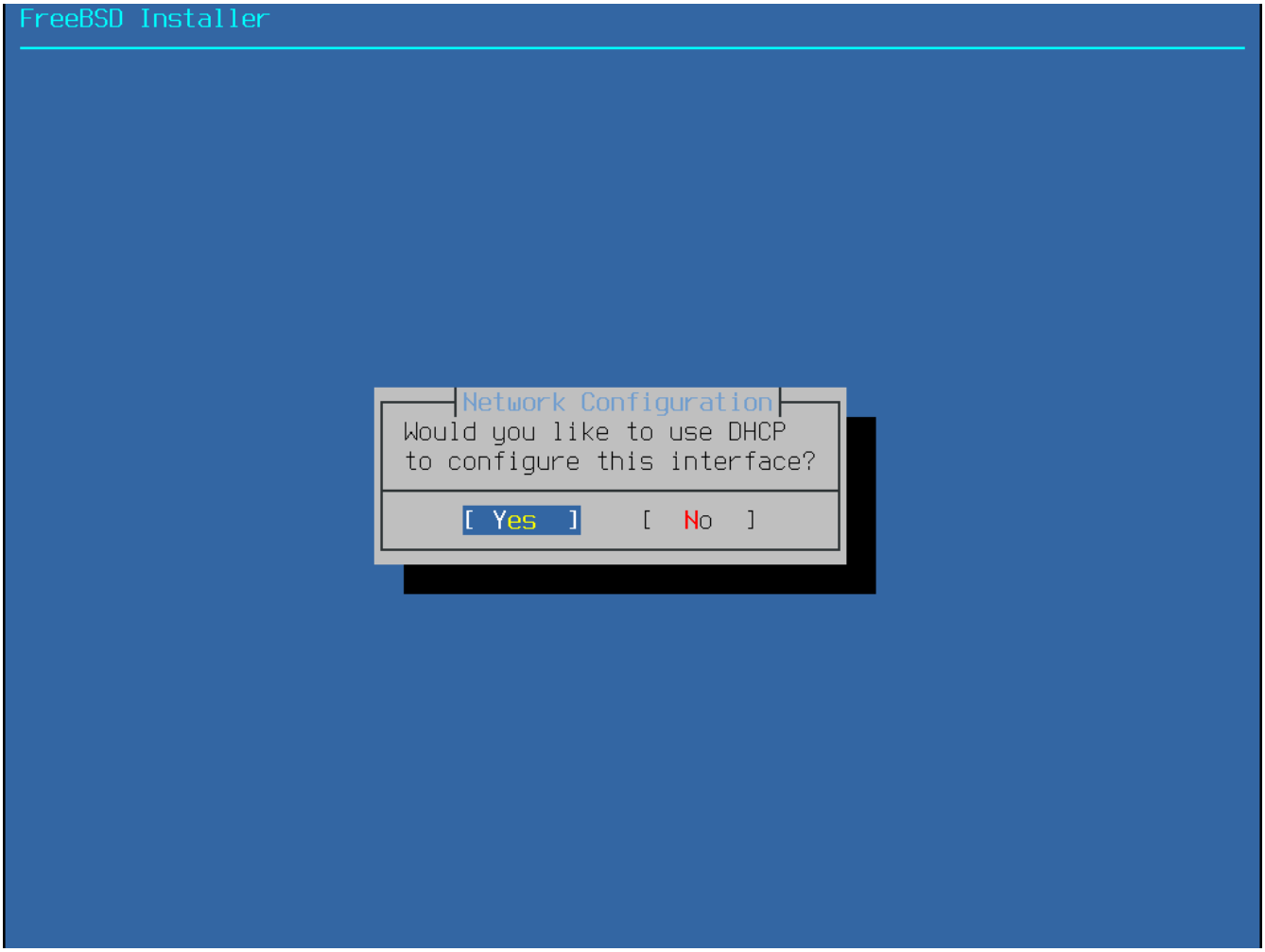


Рисунок 40. Выберите конфигурацию IPv4 DHCP

Если DHCP-сервер недоступен, выберите **[Нет]** и введите следующую информацию об адресации в этом меню:

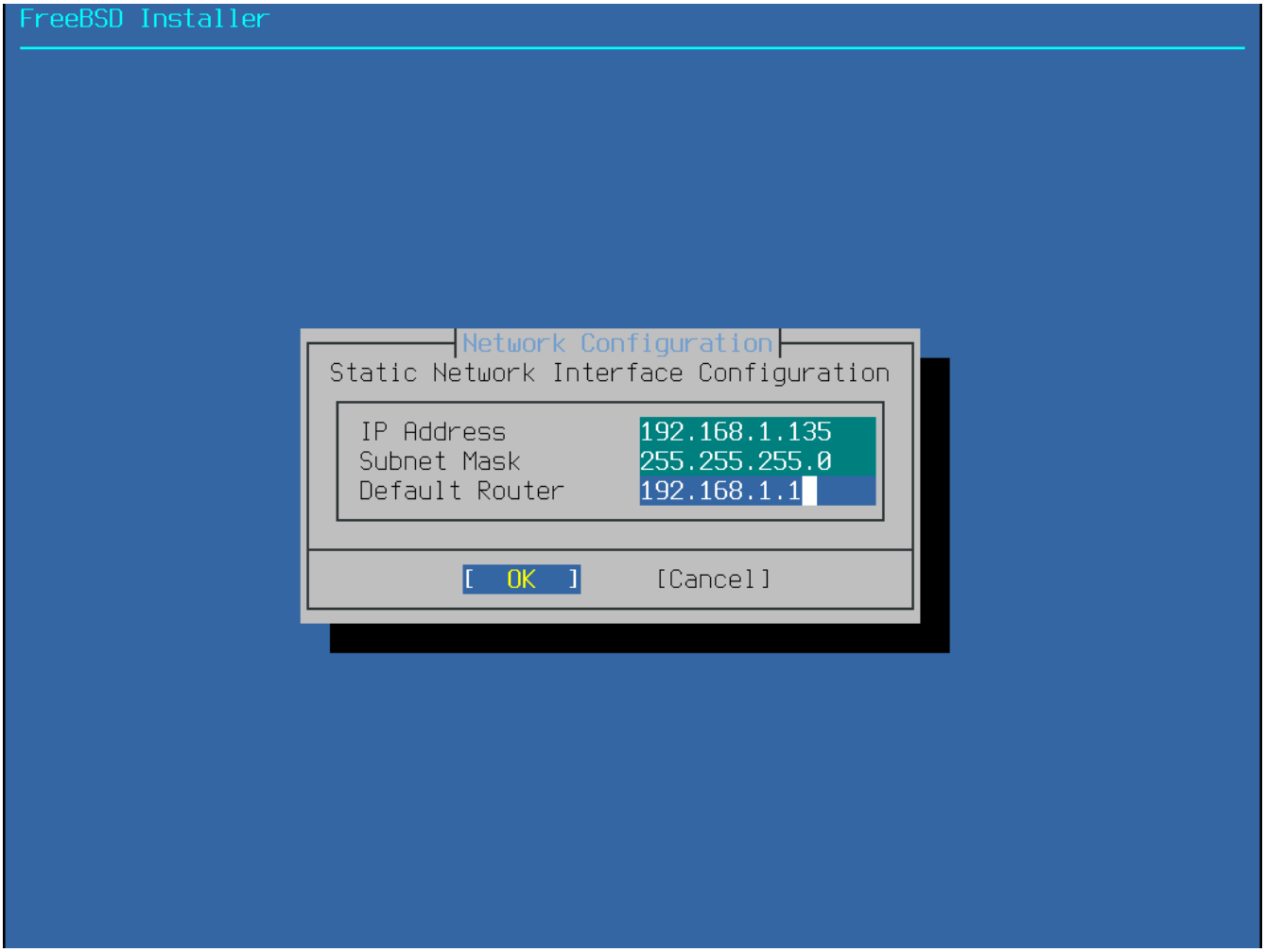


Рисунок 41. Статическая настройка IPv4

- `IP-адрес` - IPv4-адрес, назначенный этому компьютеру. Адрес должен быть уникальным и не должен уже использоваться другим устройством в локальной сети.
- `Маска подсети (Subnet Mask)` - Маска подсети для сети.
- `Шлюз по умолчанию (Default Router)` - IP-адрес сетевого шлюза по умолчанию.

Следующий экран спросит, нужно ли настраивать интерфейс для IPv6. Если IPv6 доступен и нужен, выберите **[Да]**, чтобы включить его.

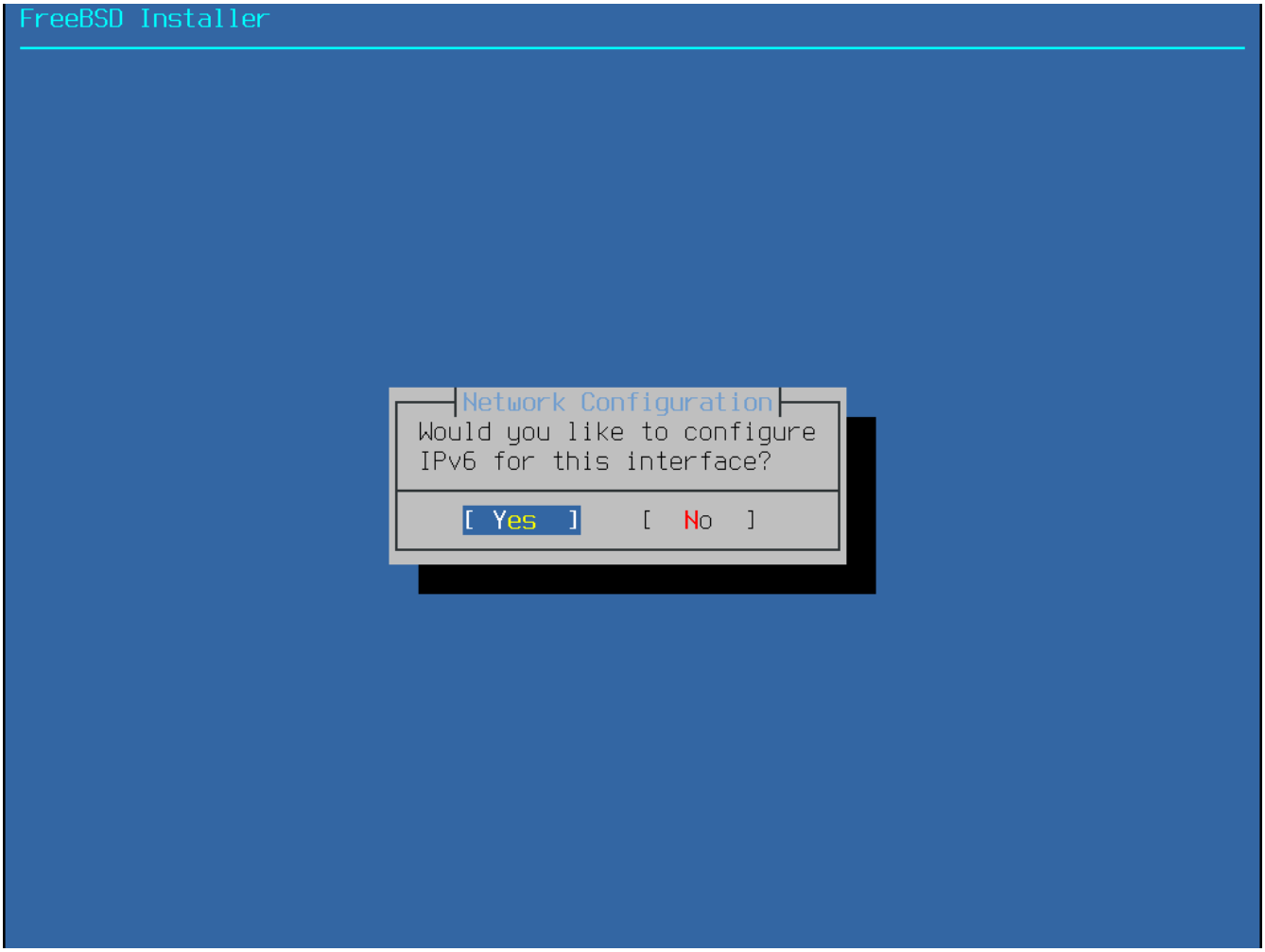


Рисунок 42. Выберите сеть IPv6

В IPv6 также есть два метода настройки. Stateless Address Autoconfiguration (SLAAC) автоматически запрашивает правильную конфигурационную информацию у локального маршрутизатора. Подробнее см. rfc4862. Статическая настройка требует ручного ввода сетевой информации.

Если доступен маршрутизатор IPv6, выберите **[Да]** в следующем меню для автоматической настройки сетевого интерфейса. Установщик может на минуту или около того показаться зависшим, пока он ищет маршрутизатор и получает адресную информацию для системы.

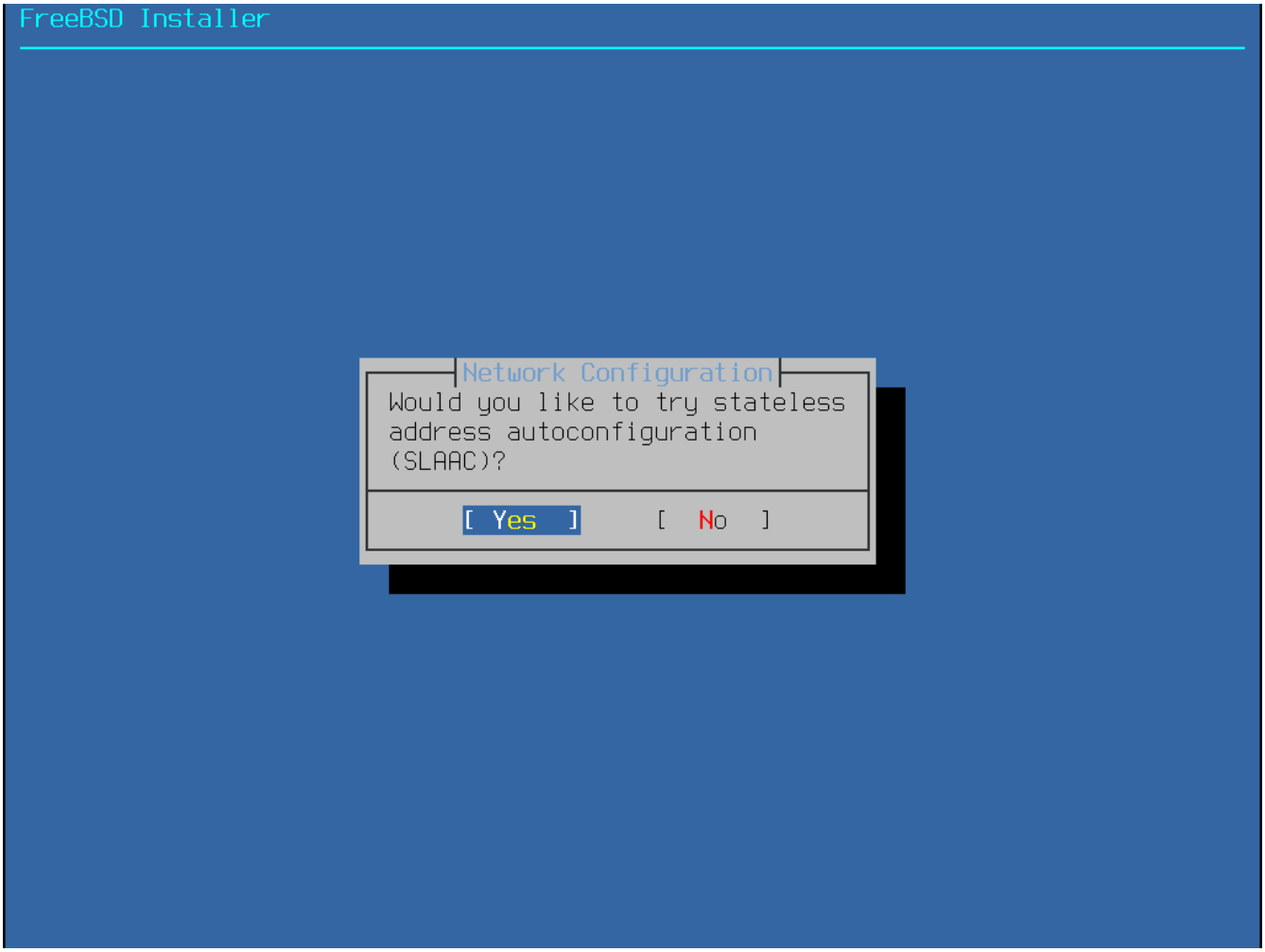


Рисунок 43. Выберите конфигурацию IPv6 SLAAC

Если маршрутизатор IPv6 недоступен, выберите **[Нет]** и введите следующую информацию о настройке адресации в этом меню:

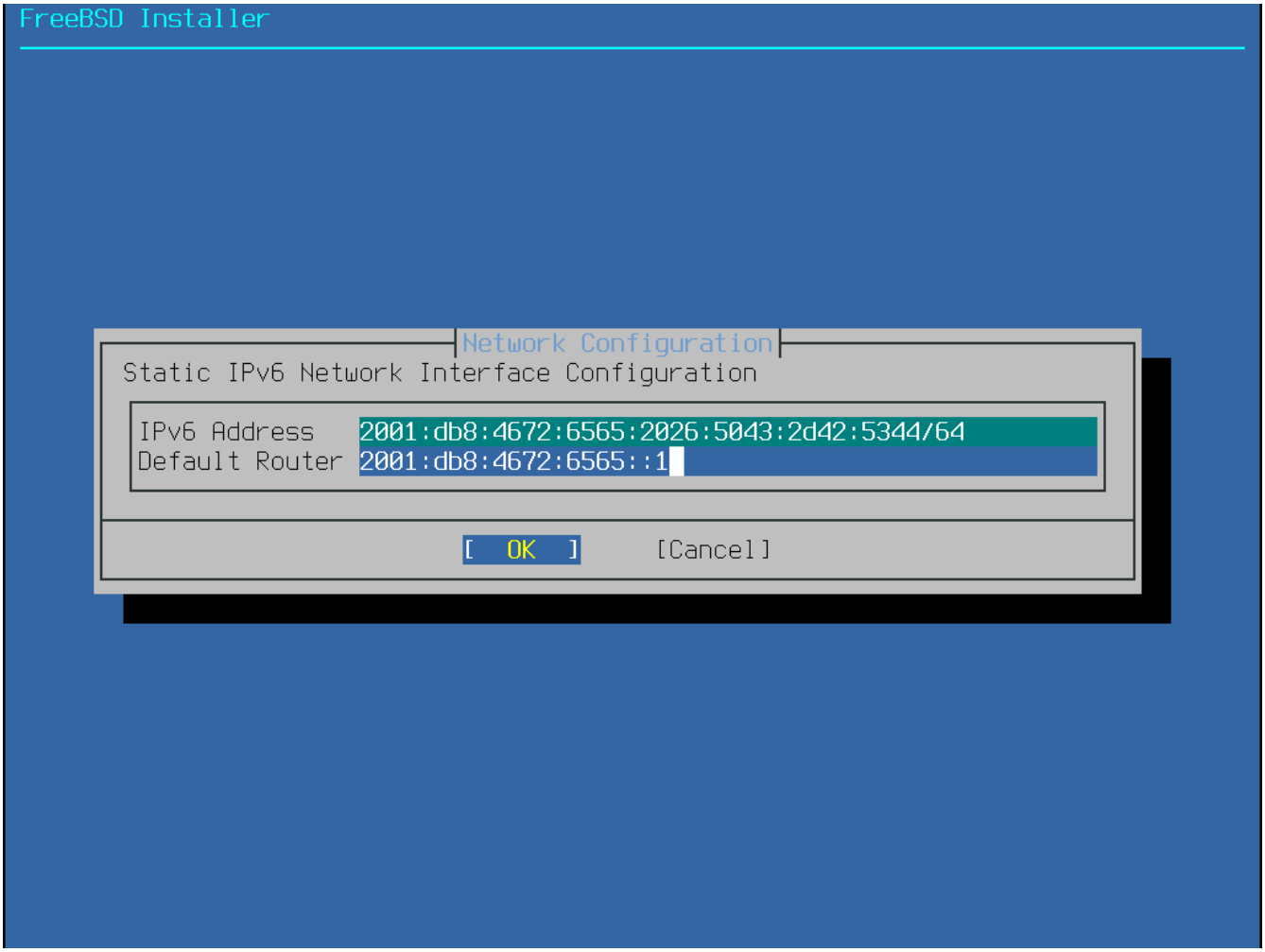


Рисунок 44. Статическая настройка IPv6

- `IPv6-адрес` - IPv6-адрес, назначенный этому компьютеру. Адрес должен быть уникальным и не использоваться другим устройством в локальной сети.
- `Шлюз по умолчанию (Default Router)` - IPv6-адрес шлюза по умолчанию в сети.

Последнее меню настройки сети используется для конфигурации резолвера Domain Name System (DNS), который преобразует имена хостов в сетевые адреса и обратно. Если для автоматической настройки сетевого интерфейса использовались DHCP или SLAAC, значения в `Resolver Configuration` могут быть уже заполнены. В противном случае введите имя домена локальной сети в поле `Search`. `DNS #1` и `DNS #2` — это IPv4 и/или IPv6-адреса DNS-серверов. Требуется указать хотя бы один DNS-сервер.

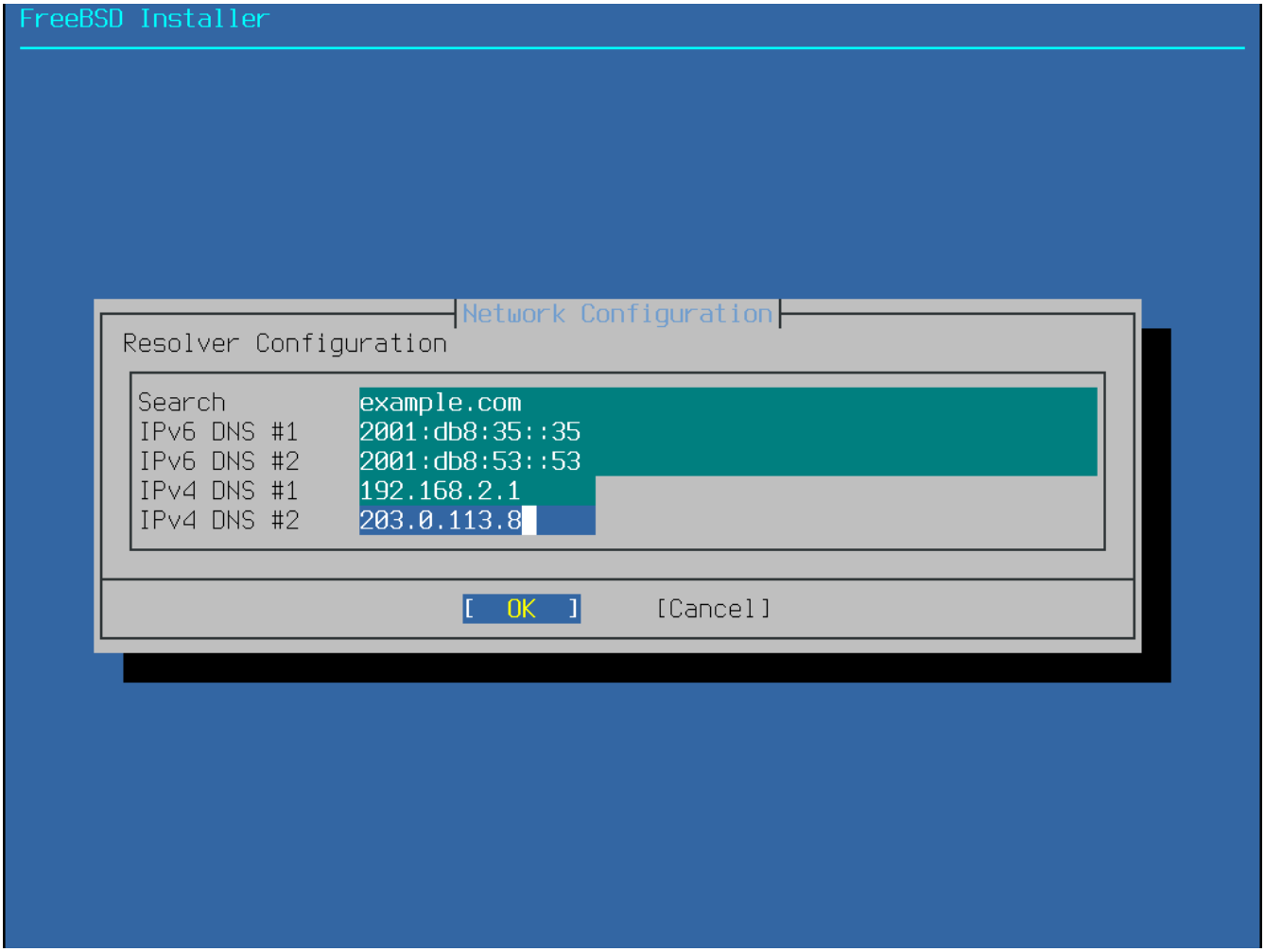


Рисунок 45. Конфигурация DNS

После настройки интерфейса выберите зеркальный сайт, расположенный в том же регионе мира, что и компьютер, на который устанавливается FreeBSD. Файлы можно загрузить быстрее, если зеркало находится ближе к целевому компьютеру, что сокращает время установки.

Выбор `http://download.freebsd.org Основной сайт (GeoDNS, HTTP)` или `ftp://download.freebsd.org Основной сайт (GeoDNS, FTP)` автоматически перенаправит на ближайший зеркальный сервер.

Рисунок 46. Выбор зеркала

2.8.3. Установка часового пояса

Следующая серия меню используется для определения правильного местного времени путем выбора географического региона, страны и часовой зоны. Установка часовой зоны позволяет системе автоматически корректировать региональные изменения времени, такие как переход на летнее время, и правильно выполнять другие функции, связанные с часовыми зонами.

Пример, приведённый здесь, предназначен для машины, находящейся в материковой часовой зоне Испании, Европа. Выбор будет варьироваться в зависимости от географического местоположения.

Рисунок 47. Выберите регион

Соответствующий регион выбирается с помощью клавиш со стрелками, а затем нажатием `Enter`.

Рисунок 48. Выберите страну

Выберите соответствующую страну с помощью клавиш со стрелками и нажмите `Enter`.

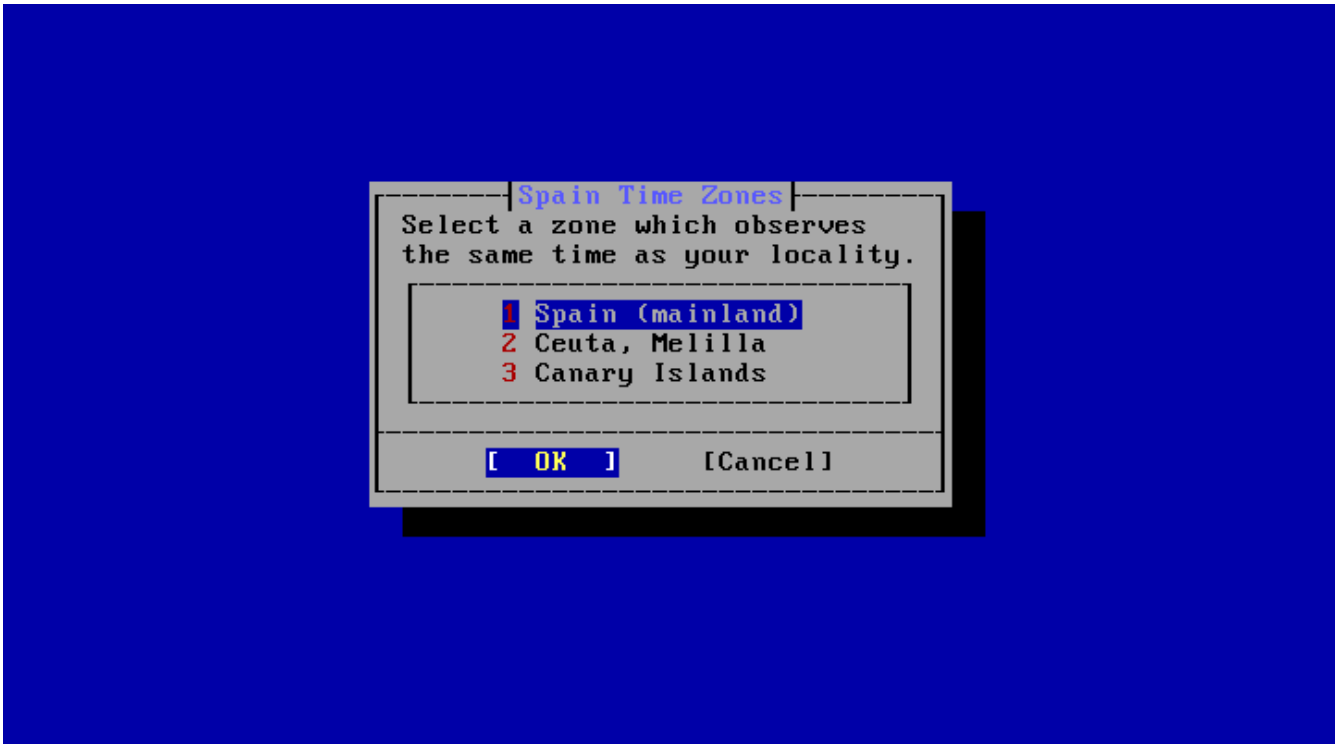


Рисунок 49. Выбор часовой зоны

Используя клавиши со стрелками, выберите подходящую часовую зону и нажмите `Enter`.

Рисунок 50. Подтверждение часовой зоны

Убедитесь, что аббревиатура часовой зоны верна.

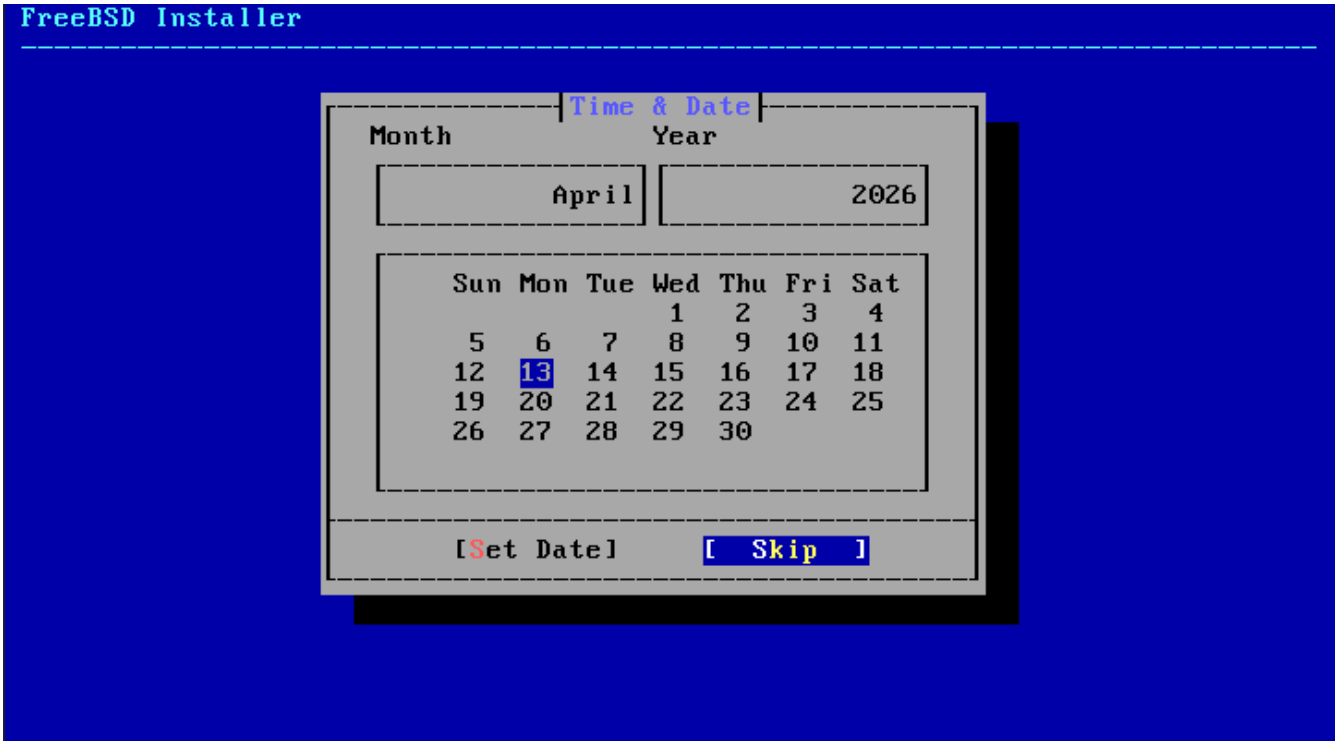


Рисунок 51. Выбор даты

Выбор нужной даты осуществляется с помощью клавиш со стрелками, после чего нажимается **[Установить дату]**. В противном случае можно пропустить выбор даты, нажав **[Пропустить]**.

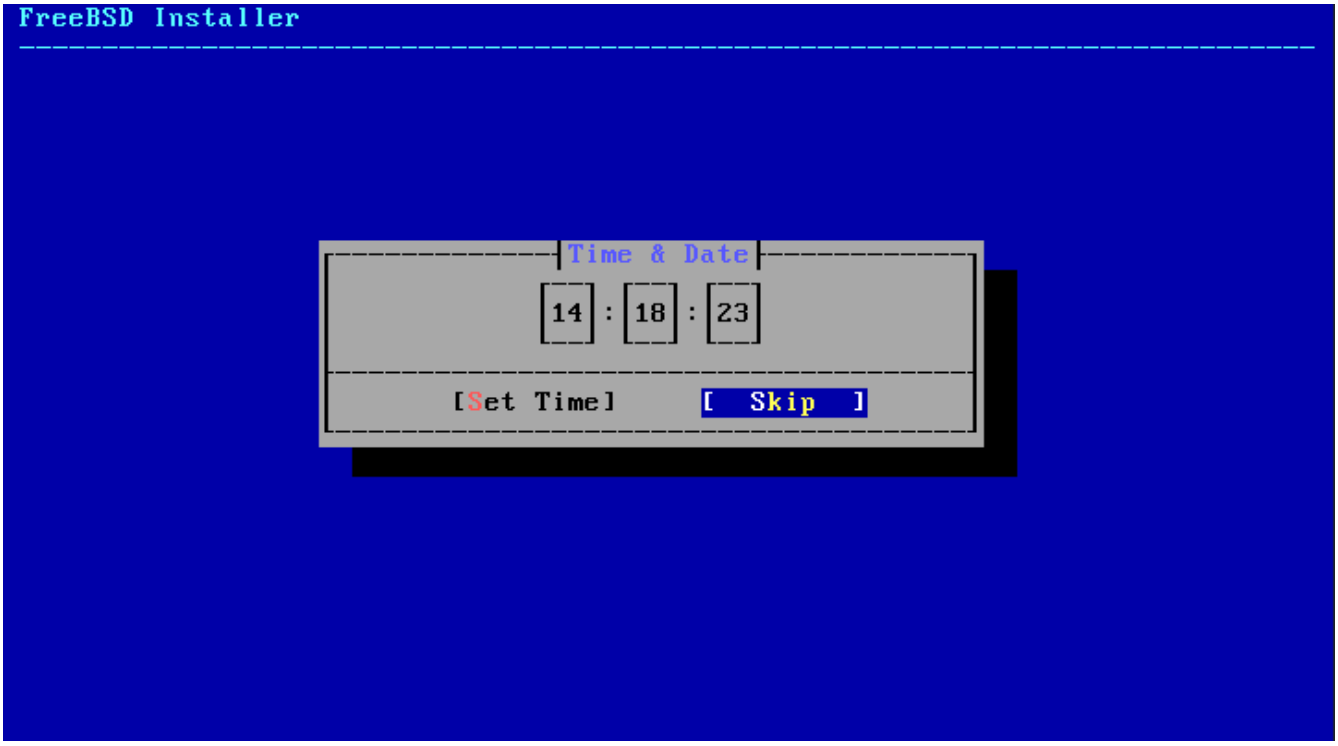


Рисунок 52. Выбор времени

Подходящее время выбирается с помощью клавиш со стрелками, а затем нажатием **[Установить время]**. В противном случае можно пропустить выбор времени, нажав **[Пропустить]**.

2.8.4. Включение сервисов

Следующее меню используется для настройки системных служб, которые будут запускаться при загрузке системы. Все эти службы являются опциональными. Запускайте только те службы, которые необходимы для функционирования системы.

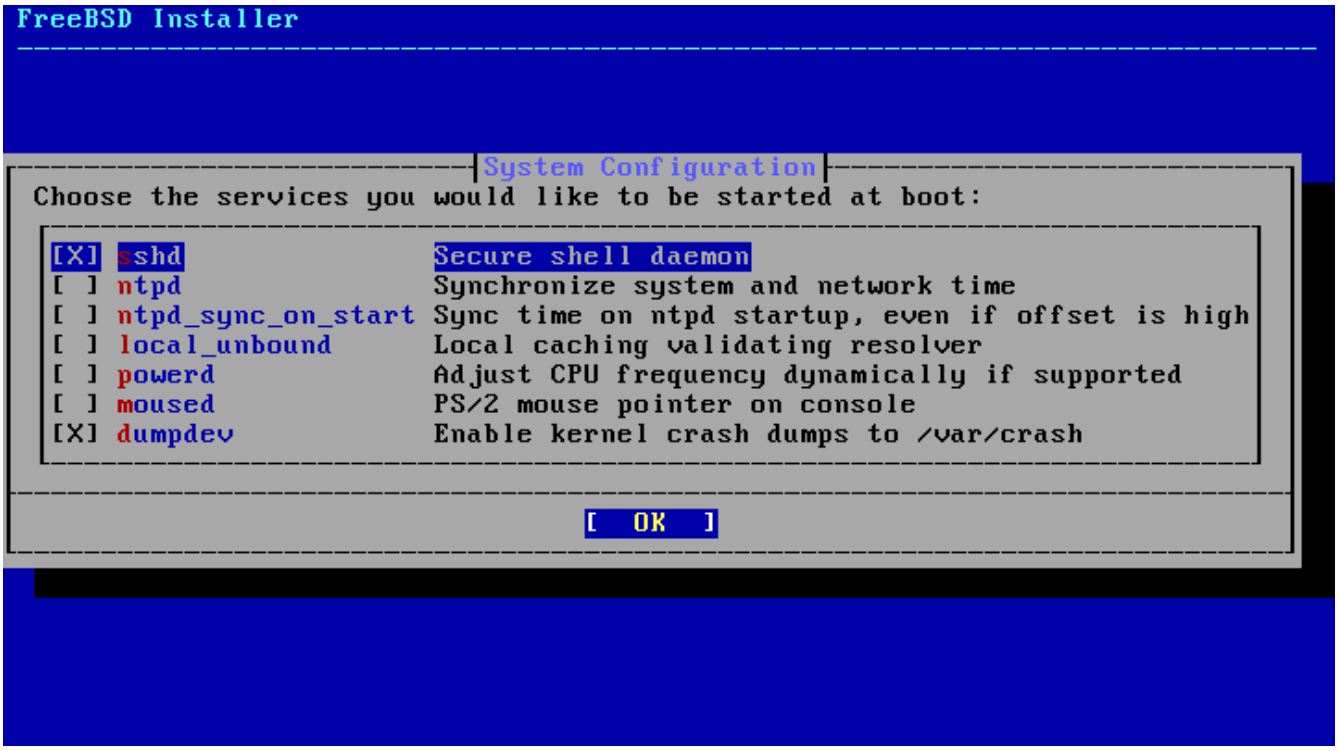


Рисунок 53. Выбор дополнительных служб для включения

Вот перечень служб, которые можно включить в этом меню:

- `sshd` — демон Secure Shell (SSH), используемый для удалённого доступа к системе через зашифрованное соединение. Включайте эту службу только если системе необходимо быть доступной для удалённых входов.
- `ntpd` — демон протокола сетевого времени (NTP) для автоматической синхронизации часов. Включите эту службу, если хотите синхронизировать системные часы с удалённым сервером времени или пулом серверов.
- `ntpd_sync_on_start` — если включено, ntpd(8) запускается с флагом `-g` во время загрузки системы, выполняя немедленную однократную синхронизацию часов при старте.
- `local_unbound` — Включить локальный DNS-резолвер unbound. Необходимо учитывать, что данная конфигурация предназначена только для использования в качестве локального кэширующего пересылающего резолвера. Если цель - настроить резолвер для всей сети, установите пакет dns/unbound.
- `powerd` — Утилита управления питанием системы для контроля питания и энергосбережения.
- `moused` — Включите эту службу, если мышь будет использоваться в командной строке системной консоли.
- `dumpdev` — Дампы памяти полезны при отладке проблем с системой, поэтому пользователям рекомендуется их включить.

2.8.5. Включение параметров усиленной безопасности

Следующее меню используется для настройки параметров безопасности, которые будут включены. Все эти параметры необязательны, но их использование рекомендуется.

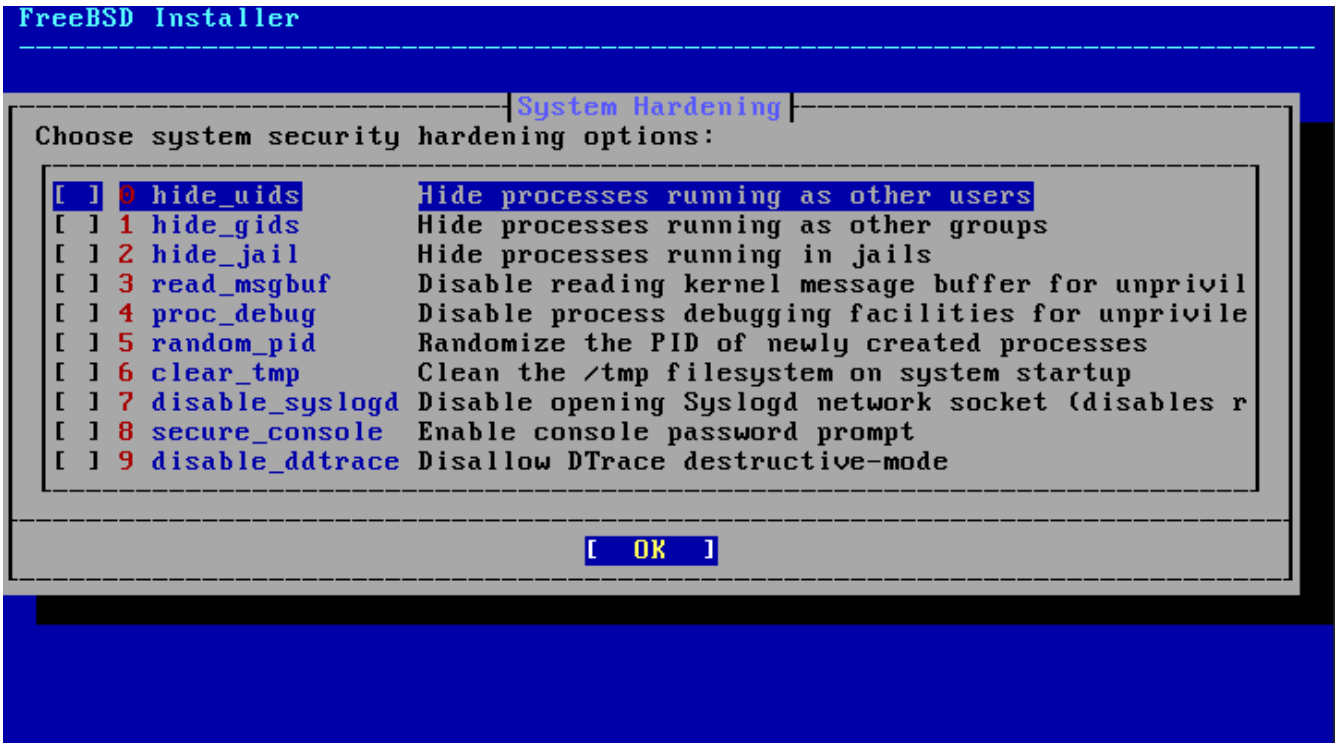


Рисунок 54. Выбор параметров усиленной безопасности

Вот сводка опций, которые можно включить в этом меню:

- `hide_uids` — Скрывать процессы, выполняемые от имени других пользователей (UID). Это предотвращает возможность непривилегированным пользователям видеть запущенные процессы других пользователей.
- `hide_gids` — Скрывать процессы, выполняемые от имени других групп (GID). Это предотвращает возможность непривилегированных пользователей видеть выполняемые процессы других групп.
- `hide_jail` — Скрывать процессы, выполняющиеся в клетке. Это предотвращает возможность непривилегированным пользователям видеть процессы, выполняющиеся внутри клетки.
- `read_msgbuf` — Запретить чтение буфера сообщений ядра непривилегированным пользователям. Предотвращает возможность использования непривилегированными пользователями dmesg(8) для просмотра сообщений из буфера журнала ядра.
- `proc_debug` — Отключает средства отладки процессов для непривилегированных пользователей. Отключает различные сервисы отладки межпроцессного взаимодействия для непривилегированных пользователей, включая некоторую функциональность procfs, `ptrace()` и `ktrace()`. Обратите внимание, что это также заблокирует работу инструментов отладки, таких как lldb(1), truss(1) и procstat(1), а также некоторых встроенных средств отладки в определённых скриптовых языках, например PHP.
- `random_pid` — Рандомизировать PID процессов.

- `clear_tmp` — очистка `/tmp` при запуске системы.
- `disable_syslogd` — Отключить открытие сетевого сокета syslogd. По умолчанию FreeBSD запускает syslogd в безопасном режиме с параметром `-s`, что предотвращает прослушивание входящих UDP-запросов на порту 514. При включении этой опции syslogd будет запущен с параметром `-ss`, что запрещает ему открывать какие-либо порты. Подробнее см. syslogd(8).
- `secure_console` — заставляет командную строку запрашивать пароль `root` при входе в однопользовательский режим.
- `disable_ddtrace` — DTrace может работать в режиме, который влияет на работающее ядро. Деструктивные действия не могут быть использованы, если они явно не включены. Используйте `-w` для включения этой опции при работе с DTrace. Для получения дополнительной информации см. dtrace(1).

2.8.6. Добавление пользователей

Следующее меню предлагает создать хотя бы одну учётную запись пользователя. Рекомендуется входить в систему под учётной записью пользователя, а не как `root`. При входе под `root` практически отсутствуют ограничения или защита от возможных действий. Вход под обычным пользователем безопаснее и надёжнее.

Выберите **[Да]**, чтобы добавить новых пользователей.

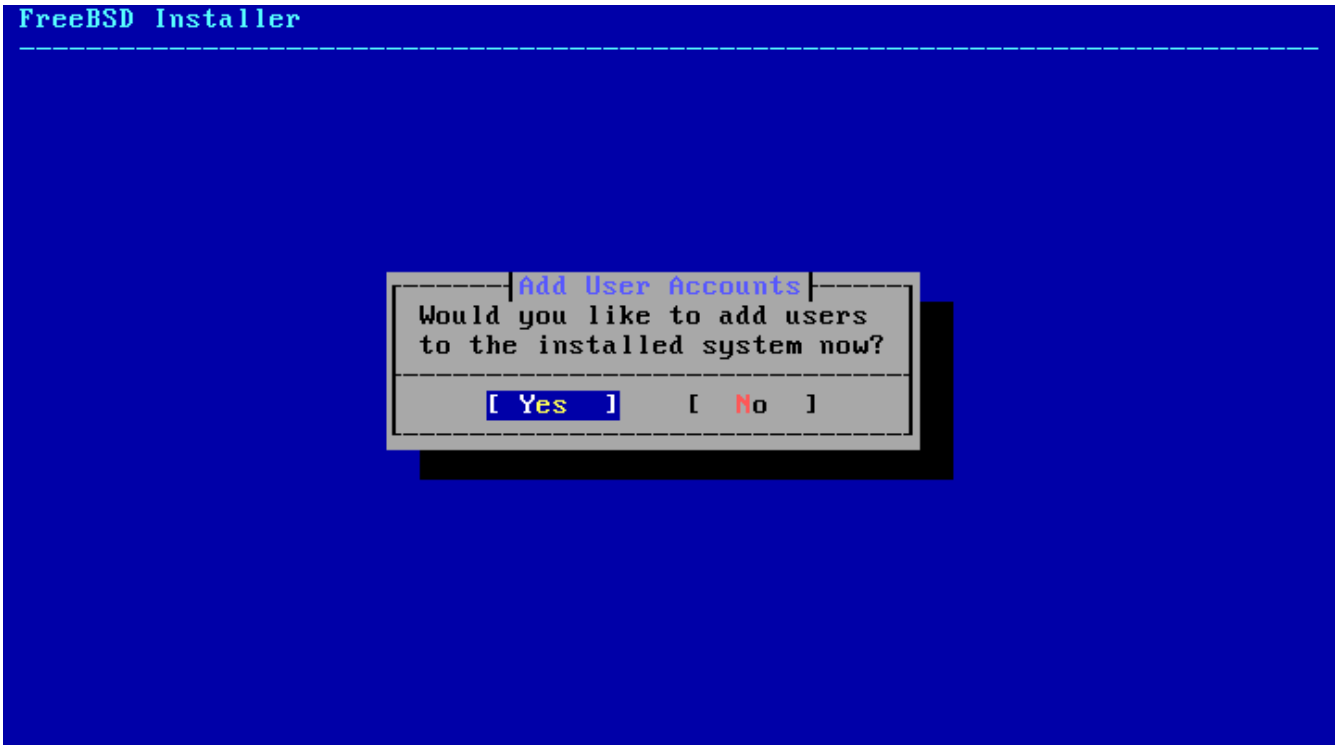


Рисунок 55. Добавить учётные записи пользователей

Следуйте подсказкам и введите запрашиваемую информацию для учётной записи пользователя. Пример, показанный в Ввод информации о пользователе, создаёт учётную запись пользователя `asample`.

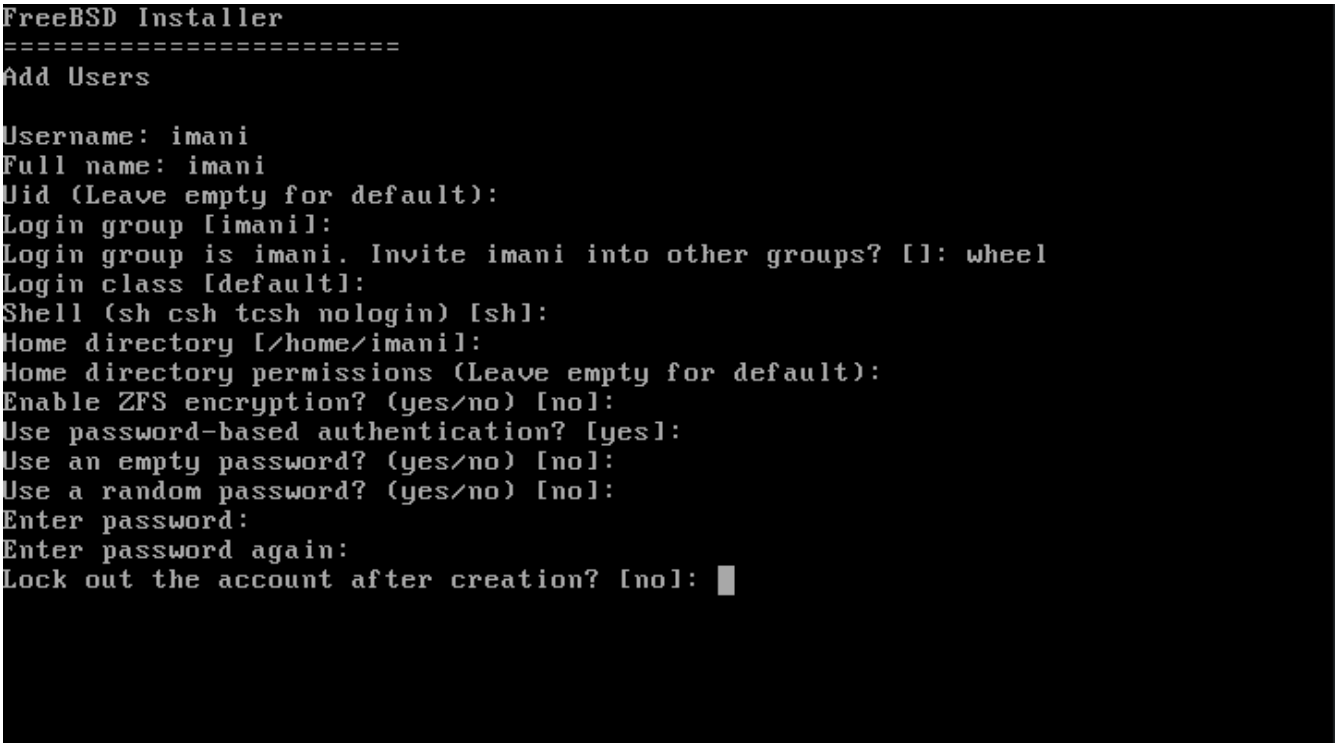


Рисунок 56. Введите информацию о пользователе

Вот сводка информации для ввода:

- `Имя пользователя (Username)` — Имя пользователя, которое будет использоваться для входа. Обычно применяется соглашение, согласно которому имя пользователя формируется из первой буквы имени и фамилии, при условии что каждое имя пользователя уникально в системе. Имя пользователя чувствительно к регистру и не должно содержать пробелов.
- `Полное имя (Full name)` — Полное имя пользователя. Может содержать пробелы и используется в качестве описания учётной записи пользователя.
- `Uid` - Идентификатор пользователя. Обычно оставляется пустым, чтобы система автоматически назначила значение.
- `Группа (Login group)` - Группа пользователя. Обычно оставляется пустым для использования значения по умолчанию.
- `Добавить пользователя в другие группы?` — Дополнительные группы, в которые будет добавлен пользователь. Если пользователю нужны административные права, укажите здесь `wheel`.
- `Класс логина (Login class)` - Обычно оставляется пустым для значений по умолчанию.
- `Оболочка (Shell)` - Введите одно из предложенных значений, чтобы установить интерактивную оболочку для пользователя. Дополнительную информацию об оболочках см. в Оболочки.
- `Домашний каталог (Home directory)` - Домашний каталог пользователя. Обычно значение по умолчанию является правильным.
- `Права доступа к домашнему каталогу (Home directory permissions)` - Права доступа к домашнему каталогу пользователя. Обычно значение по умолчанию является правильным.

- Использовать аутентификацию на основе пароля (Use password-based authentication)? - Обычно yes, чтобы пользователь вводил пароль при входе.
- Использовать пустой пароль (Use an empty password)? - Обычно нет, так как пустые или простые пароли ненадёжны.
- Использовать случайный пароль (Use a random password)? - Обычно нет, чтобы пользователь мог установить свой пароль в следующем запросе.
- Введите пароль (Enter password) - Пароль для этого пользователя. Вводимые символы не будут отображаться на экране.
- Повторите пароль (Enter password again) - Пароль должен быть введён повторно для проверки.
- Заблокировать учётную запись после создания (Lock out the account after creation)? - Обычно нет, чтобы пользователь мог войти в систему.

После ввода всех данных отображается сводка для проверки. Если была допущена ошибка, введите no, чтобы исправить её. Когда всё верно, введите yes для создания нового пользователя.

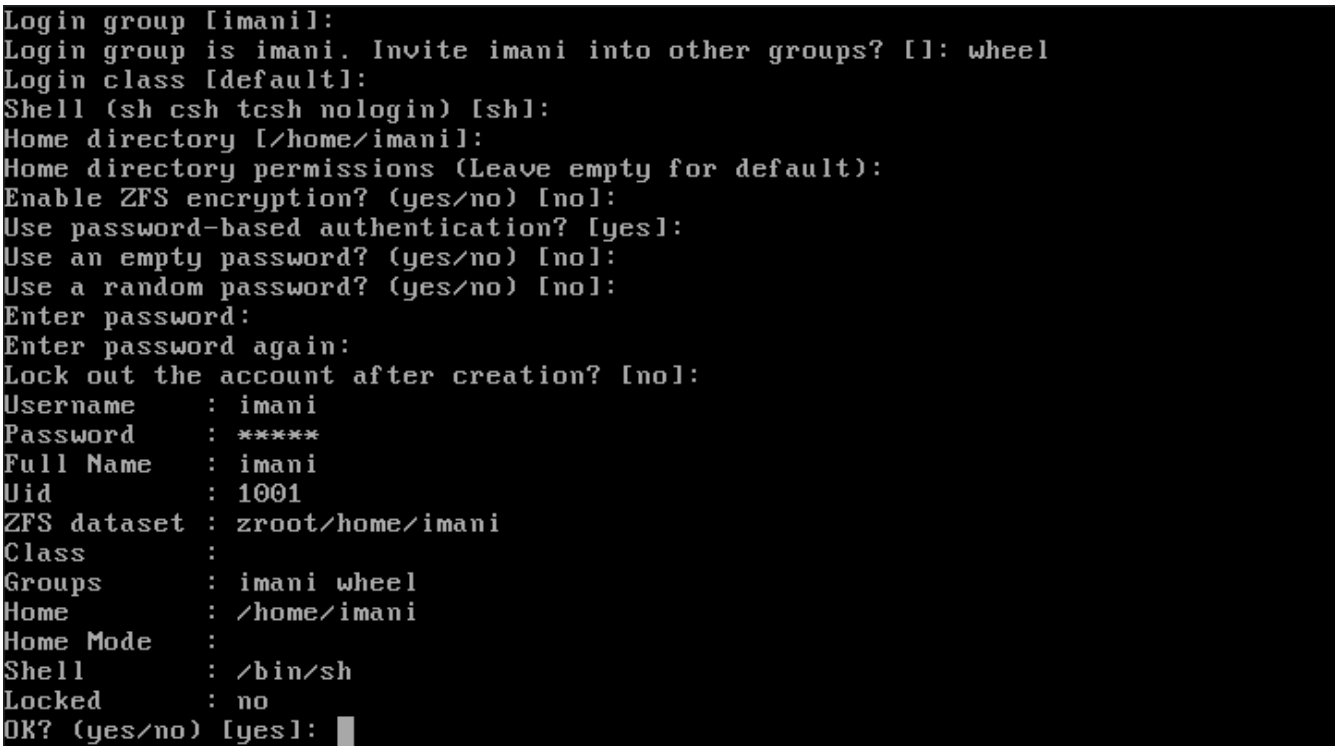


Рисунок 57. Выход из управления пользователями и группами

Если нужно добавить других пользователей, ответьте yes на вопрос Добавить другого пользователя (Add another user)?. Введите no, чтобы завершить добавление пользователей и продолжить установку.

Для получения дополнительной информации о добавлении пользователей и управлении учётными записями см. Пользователи и основы управления учётными записями.

2.8.7. Окончательная конфигурация

После установки и настройки всех компонентов предоставляется последняя возможность

изменить параметры.

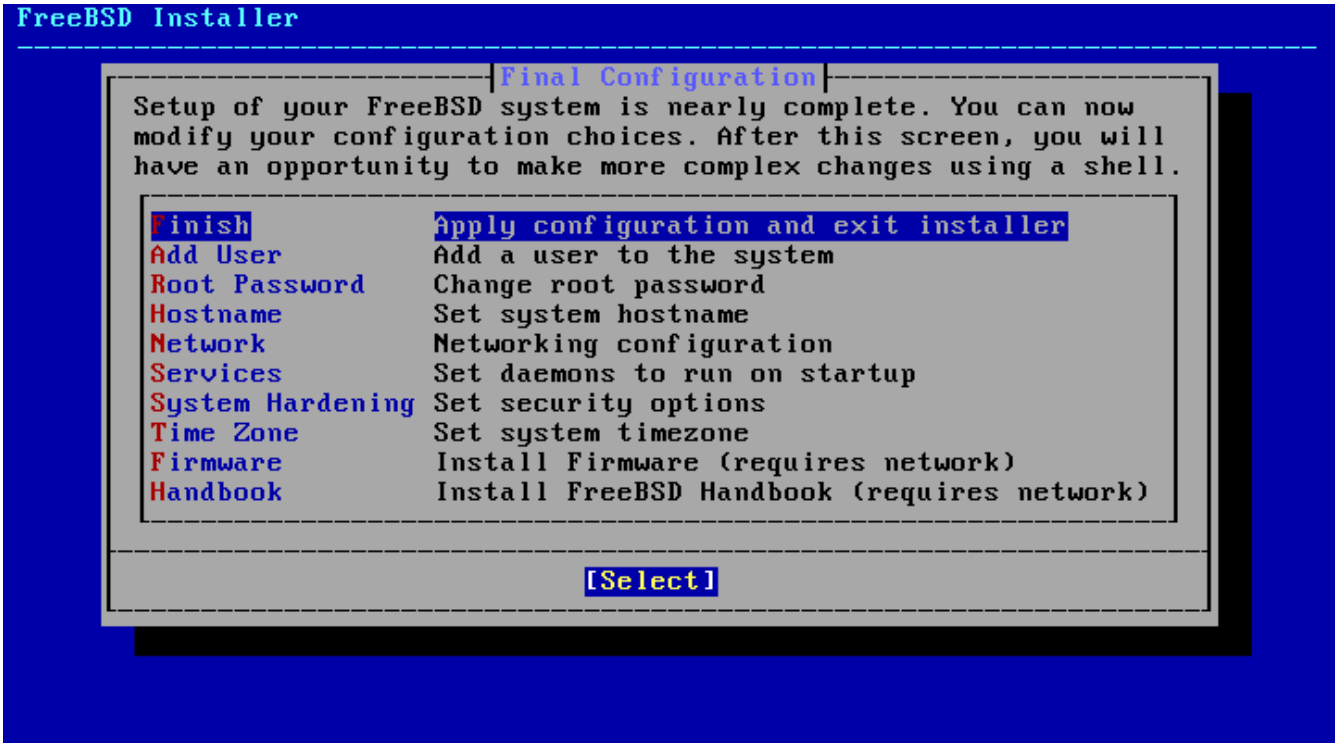


Рисунок 58. Окончательная конфигурация

Используйте это меню для внесения изменений или выполнения дополнительной настройки перед завершением установки.

- `Добавить пользователя (Add User)` - Описано в Добавление пользователей.
- `Пароль root (Root Password)` - Описано в Установка пароля `root`.
- `Имя хоста (Hostname)` - Описано в Установка имени хоста.
- `Сеть (Network)` - Описано в Настройка сетевых интерфейсов.
- `Службы (Services)` - Описано в Включение служб.
- `Усиление защиты системы (System Hardening)` - Описано в Включение опций усиления безопасности.
- `Часовая зона (Time Zone)` - Описано в Настройка часовой зоны.
- `Firmware` (Прошивка) — загружает и устанавливает необходимое микропрограммное обеспечение для системных компонентов, таких как Wi-Fi или графические карты.
- `Руководство (Handbook)` - Загрузить и установить Руководство FreeBSD.

После завершения настройки выберите **[Выход (Exit)]**.

Рисунок 59. Ручная настройка

bsdinstall предложит выполнить любую дополнительную настройку, необходимую перед перезагрузкой в новую систему. Выберите **[Да]**, чтобы выйти в оболочку новой системы, или **[Нет]**, чтобы перейти к последнему шагу установки.

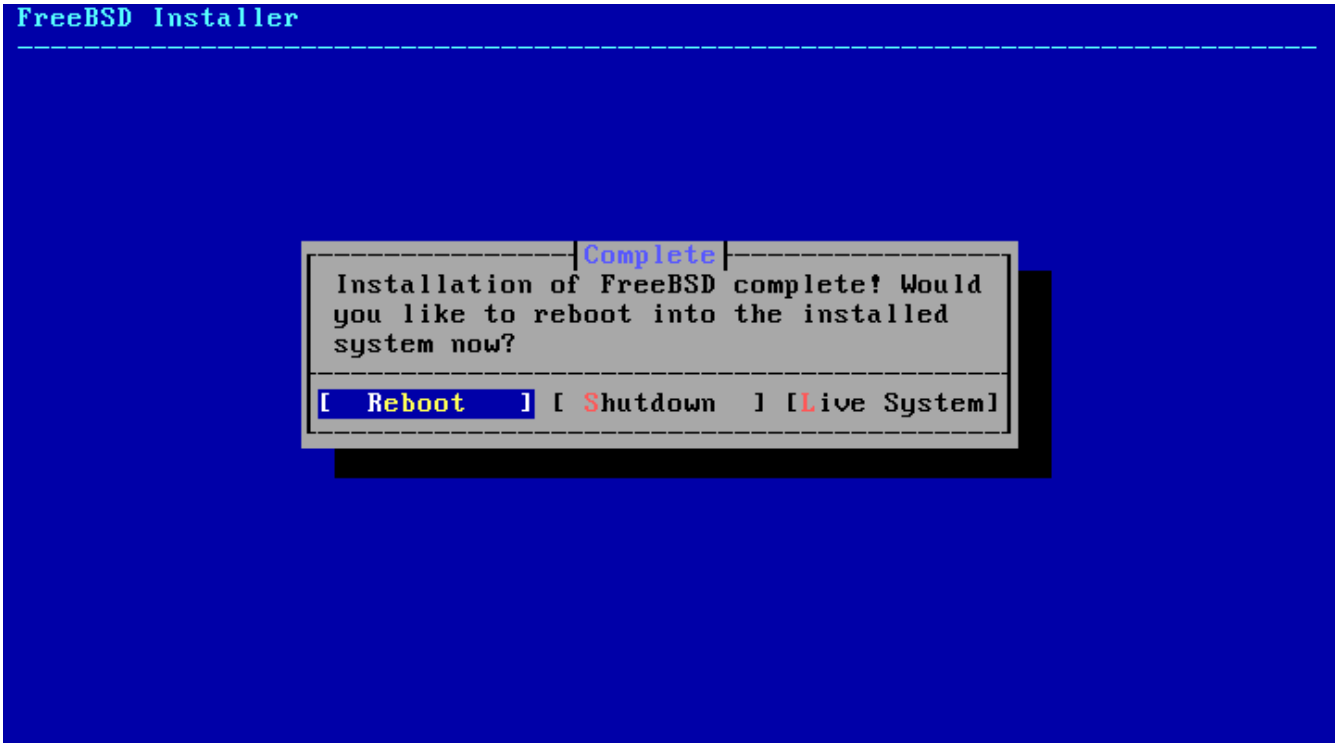


Рисунок 60. Завершение установки

Если требуется дополнительная настройка или специальная установка, выберите **[Live CD]**, чтобы загрузить установочный носитель в режиме Live CD.

Если установка завершена, выберите **[Перезагрузка]**, чтобы перезагрузить компьютер и запустить новую систему FreeBSD. Не забудьте извлечь установочный носитель FreeBSD,

иначе компьютер может снова загрузиться с него.

При загрузке FreeBSD отображаются информационные сообщения. После завершения загрузки системы появляется приглашение для входа. В ответ на приглашение `login:` введите имя пользователя, добавленное во время установки. Избегайте входа в систему как `root`. Инструкции по получению прав суперпользователя, когда требуется административный доступ, приведены в Учётная запись суперпользователя.

Сообщения, появляющиеся во время загрузки, можно просмотреть, нажав `Scroll-Lock`, чтобы включить буфер прокрутки. Для перемещения по сообщениям можно использовать клавиши `PgUp`, `PgDn` и стрелки. По завершении нажмите `Scroll-Lock` снова, чтобы разблокировать экран и вернуться к консоли. Для просмотра этих сообщений после работы системы в течение некоторого времени введите `less /var/run/dmesg.boot` в командной строке. Нажмите `q`, чтобы вернуться в командную строку после просмотра.

Если в Выбор дополнительных служб для включения была включена служба sshd, первая загрузка может быть немного медленнее, так как система генерирует SSH-ключи хоста. Последующие загрузки будут быстрее. Отпечатки ключей отображаются, как в следующем примере:

```
Generating public/private rsa1 key pair.
Your identification has been saved in /etc/ssh/ssh_host_key.
Your public key has been saved in /etc/ssh/ssh_host_key.pub.
The key fingerprint is:
10:a0:f5:af:93:ae:a3:1a:b2:bb:3c:35:d9:5a:b3:f3 root@machine3.example.com
The key's randomart image is:
+--[RSA1 1024]----+
|    o..          |
|   o . .         |
|  .   o          |
|       o         |
|    o   S        |
|   + + o         |
|o . + *          |
|o+ ..+ .         |
|==o..o+E         |
+-----------------+
Generating public/private dsa key pair.
Your identification has been saved in /etc/ssh/ssh_host_dsa_key.
Your public key has been saved in /etc/ssh/ssh_host_dsa_key.pub.
The key fingerprint is:
7e:1c:ce:dc:8a:3a:18:13:5b:34:b5:cf:d9:d1:47:b2 root@machine3.example.com
The key's randomart image is:
+--[ DSA 1024]----+
|       ..     . .|
|      o  .   . + |
|     . ..   . E .|
|    . .  o o . . |
|     +  S = .    |
|    +  . = o     |
```

```
|     +  . * .    |
|    . .  o .     |
|      .o. .      |
+-----------------+
Starting sshd.
```

Обратитесь к разделу OpenSSH для получения дополнительной информации об отпечатках и SSH.

FreeBSD не устанавливает графическое окружение по умолчанию. Дополнительную информацию об установке и настройке графического оконного менеджера можно найти в The X Window System.

Правильное завершение работы компьютера под управлением FreeBSD помогает защитить данные и оборудование от повреждений. *Не отключайте питание до того, как система будет правильно остановлена!* Если пользователь является членом группы `wheel`, необходимо стать суперпользователем, введя `su` в командной строке и указав пароль `root`. Затем введите `shutdown -p now`, и система корректно завершит работу, а если оборудование поддерживает такую возможность, выключится автоматически.

2.9. Устранение неполадок

Этот раздел посвящён устранению основных проблем при установке, включая распространённые ошибки, о которых сообщали пользователи.

Проверьте заметки о совместимости оборудования на странице информации о выпусках FreeBSD для соответствующей версии FreeBSD, чтобы убедиться, что оборудование поддерживается.

Некоторые проблемы при установке можно избежать или уменьшить, обновив микропрограмму различных компонентов оборудования, в первую очередь материнской платы. Микропрограмма материнской платы обычно называется BIOS. У большинства производителей материнских плат и компьютеров есть веб-сайты с обновлениями и информацией о них.

Производители, как правило, не рекомендуют обновлять BIOS материнской платы без веской причины, такой как критическое обновление. Процесс обновления *может* пройти неудачно, что приведёт к повреждению BIOS и неработоспособности компьютера.

Если система зависает при проверке оборудования во время загрузки или ведёт себя странно в процессе установки, причиной может быть ACPI. FreeBSD активно использует системную службу ACPI на платформах i386 и amd64 для помощи в настройке системы, если она обнаружена во время загрузки. К сожалению, в драйвере ACPI, а также в материнских платах и BIOS до сих пор существуют ошибки. ACPI можно отключить, установив подсказку `hint.acpi.0.disabled` в загрузчике третьей стадии:

```
set hint.acpi.0.disabled="1"
```

Этот параметр сбрасывается при каждой загрузке системы, поэтому необходимо добавить `hint.acpi.0.disabled="1"` в файл `/boot/loader.conf`. Дополнительную информацию о загрузчике можно найти в Synopsis.

2.10. Использование Live CD

Меню приветствия `bsdinstall`, показанное в Меню приветствия, предоставляет опцию **[Live CD]**. Это полезно для тех, кто ещё сомневается, подходит ли FreeBSD в качестве операционной системы, и хочет протестировать некоторые функции перед установкой.

Следует отметить следующие моменты перед использованием **[Live CD]**:

- Для доступа к системе требуется аутентификация. Имя пользователя — `root`, пароль пустой.
- Поскольку система работает непосредственно с установочного носителя, производительность будет значительно ниже, чем у системы, установленной на жёсткий диск.
- Эта опция предоставляет только командную строку, а не графический интерфейс.

Глава 3. Основы FreeBSD

3.1. Обзор

В этой главе рассматриваются основные команды и функциональные возможности операционной системы FreeBSD. Большая часть материала применима к любым UNIX®-подобным операционным системам. Новым пользователям FreeBSD рекомендуется внимательно ознакомиться с этой главой.

Прочтите эту главу, чтобы узнать:

- Как использовать и настраивать виртуальные консоли.
- Как создавать пользователей и группы пользователей во FreeBSD и управлять ими.
- Как работают права доступа на файлы в UNIX® и файловые флаги во FreeBSD.
- Иерархию каталогов FreeBSD.
- Организация дисков в FreeBSD.
- Как монтировать и размонтировать файловые системы.
- Что такое процессы, демоны и сигналы.
- Что такое оболочка и как изменить среду входа по умолчанию.
- Как использовать основные текстовые редакторы.
- Что такое устройства и узлы устройств.
- Как читать справочные страницы для получения дополнительной информации.

3.2. Виртуальные консоли и терминалы

Если FreeBSD не настроена для автоматического запуска графической среды во время загрузки, система загрузится в командную строку с приглашением для входа, как показано в этом примере:

```
FreeBSD/amd64 (pc3.example.org) (ttyv0)

login:
```

Первая строка содержит некоторую информацию о системе. `amd64` указывает, что FreeBSD работает на 64-битной x86 системе. Имя хоста — `pc3.example.org`, а `ttyv0` означает, что это «системная консоль». Вторая строка — это приглашение для входа в систему.

Поскольку FreeBSD — это многопользовательская система, требуется способ различать пользователей. Для этого необходимо, чтобы каждый пользователь вошел в систему перед получением доступа к программам. У каждого пользователя есть уникальное «имя пользователя» и личный «пароль».

Для входа в системную консоль введите имя пользователя, заданное при установке

системы, как описано в Добавление пользователей, и нажмите `Enter`. Затем введите пароль, связанный с этим именем пользователя, и нажмите `Enter`. Пароль *не отображается* в целях безопасности.

После ввода правильного пароля будет отображено сообщение дня (MOTD), за которым последует приглашение командной строки. В зависимости от выбранной оболочки при создании пользователя, это приглашение может быть символом `#`, `$` или `%`. Приглашение указывает, что пользователь теперь вошел в консоль системы FreeBSD и готов попробовать доступные команды.

3.2.1. Виртуальные консоли

Хотя системную консоль можно использовать для взаимодействия с системой, пользователь, работающий из командной строки на клавиатуре FreeBSD, обычно вместо этого входит в виртуальную консоль. Это связано с тем, что системные сообщения по умолчанию настроены на отображение на системной консоли. Эти сообщения будут появляться поверх команды или файла, с которыми работает пользователь, что затрудняет концентрацию на текущей задаче.

По умолчанию FreeBSD настроена для предоставления нескольких виртуальных консолей для ввода команд. Каждая виртуальная консоль имеет собственную приглашение для входа и оболочку, и переключение между виртуальными консолями осуществляется легко. Это, по сути, предоставляет эквивалент командной строки для одновременного открытия нескольких окон в графической среде.

Сочетания клавиш `Alt` + `F1` — `Alt` + `F8` зарезервированы в FreeBSD для переключения между виртуальными консолями. Используйте `Alt` + `F1` для перехода к системной консоли (`ttyv0`), `Alt` + `F2` для доступа к первой виртуальной консоли (`ttyv1`), `Alt` + `F3` для доступа ко второй виртуальной консоли (`ttyv2`) и так далее. При использовании Xorg в качестве графической консоли для возврата к текстовой виртуальной консоли используется сочетание `Ctrl` + `Alt` + `F1`.

При переключении с одной консоли на другую FreeBSD управляет выводом на экран. В результате создаётся иллюзия наличия нескольких виртуальных экранов и клавиатур, которые можно использовать для ввода команд на выполнение в FreeBSD. Программы, запущенные в одной виртуальной консоли, продолжают работать при переключении пользователя на другую виртуальную консоль.

Обратитесь к kbdcontrol(1), vidcontrol(1), atkbd(4), syscons(4) и vt(4) для более технического описания консоли FreeBSD и её драйверов клавиатуры.

В FreeBSD количество доступных виртуальных консолей настраивается в этом разделе `/etc/ttys`:

```
# name    getty                           type  status comments
#
ttyv0   "/usr/libexec/getty Pc"         xterm   on  secure
# Virtual terminals
ttyv1   "/usr/libexec/getty Pc"         xterm   on  secure
```

```
ttyv2   "/usr/libexec/getty Pc"         xterm   on  secure
ttyv3   "/usr/libexec/getty Pc"         xterm   on  secure
ttyv4   "/usr/libexec/getty Pc"         xterm   on  secure
ttyv5   "/usr/libexec/getty Pc"         xterm   on  secure
ttyv6   "/usr/libexec/getty Pc"         xterm   on  secure
ttyv7   "/usr/libexec/getty Pc"         xterm   on  secure
ttyv8   "/usr/X11R6/bin/xdm -nodaemon"  xterm   off secure
```

Чтобы отключить виртуальную консоль, поставьте символ комментария (`#`) в начале строки, соответствующей этой виртуальной консоли. Например, чтобы уменьшить количество доступных виртуальных консолей с восьми до четырёх, добавьте `#` перед последними четырьмя строками, представляющими виртуальные консоли `ttyv5` до `ttyv8`. *Не* комментируйте строку системной консоли `ttyv0`. Обратите внимание, что последняя виртуальная консоль (`ttyv8`) используется для доступа к графической среде, если Xorg установлен и настроен, как описано в Система X Window.

Для подробного описания каждой колонки в этом файле и доступных опций для виртуальных консолей обратитесь к ttys(5).

3.2.2. Однопользовательский режим

В меню загрузки FreeBSD есть пункт с названием "Boot Single User". При выборе этого пункта система загрузится в особый режим, известный как "однопользовательский режим". Этот режим обычно используется для восстановления системы, которая не загружается, или для сброса пароля `root`, если он неизвестен. В однопользовательском режиме сеть и другие виртуальные консоли недоступны. Однако предоставляется полный доступ `root` к системе, и по умолчанию пароль `root` не требуется. По этим причинам для загрузки в этом режиме необходим физический доступ к клавиатуре, и определение того, кто имеет такой доступ, является важным аспектом обеспечения безопасности системы FreeBSD.

Настройки, управляющие однопользовательским режимом, находятся в этом разделе файла `/etc/ttys`:

```
# name  getty                           type  status  comments
#
# If console is marked "insecure", then init will ask for the root password
# when going to single-user mode.
console none                            unknown  off  secure
```

По умолчанию статус установлен в `secure`. Это предполагает, что тот, кто имеет физический доступ к клавиатуре, либо не важен, либо контролируется политикой физической безопасности. Если этот параметр изменён на `insecure`, предполагается, что среда сама по себе небезопасна, так как любой может получить доступ к клавиатуре. При изменении этой строки на `insecure` FreeBSD будет запрашивать пароль `root`, когда пользователь выбирает загрузку в однопользовательском режиме.

Будьте осторожны при изменении этой настройки на `insecure`*!* Если пароль `root` забыт, загрузка в однопользовательском режиме всё ещё возможна, но

> может быть затруднительна для тех, кто не знаком с процессом загрузки FreeBSD.

3.2.3. Изменение видеорежимов консоли

Режим видео по умолчанию в консоли FreeBSD может быть изменён на 1024x768, 1280x1024 или любой другой, поддерживаемый графическим чипом и монитором. Для использования другого видео режима загрузите модуль `VESA`:

```
# kldload vesa
```

Для определения видеорежимов, поддерживаемых оборудованием, используйте vidcontrol(1). Чтобы получить список поддерживаемых видеорежимов, выполните следующую команду:

```
# vidcontrol -i mode
```

Результат выполнения этой команды выводит список видеорежимов, поддерживаемых оборудованием. Чтобы выбрать новый видеорежим, укажите режим с помощью vidcontrol(1) от имени пользователя `root`:

```
# vidcontrol MODE_279
```

Если новый видео-режим приемлем, его можно установить постоянным при загрузке, добавив в `/etc/rc.conf`:

```
allscreens_flags="MODE_279"
```

3.3. Пользователи и базовая настройка учётных записей

FreeBSD позволяет нескольким пользователям одновременно работать на компьютере. Хотя только один пользователь может находиться перед экраном и использовать клавиатуру в любой момент времени, любое количество пользователей может войти в систему через сеть. Для использования системы каждый пользователь должен иметь свою собственную учётную запись.

Эта глава описывает:

- Типы учётных записей пользователей в системе FreeBSD.
- Как добавлять, удалять и изменять учётные записи пользователей.
- Как установить ограничения для контроля ресурсов, доступных пользователям и группам.

- Как создавать группы и добавлять пользователей в качестве членов группы.

3.3.1. Типы учётных записей

Поскольку весь доступ к системе FreeBSD осуществляется с использованием учётных записей, а все процессы выполняются пользователями, управление пользователями и учётными записями является важным.

Существует три основных типа учётных записей: системные учётные записи, пользовательские учётные записи и учётная запись суперпользователя.

Системные учётные записи

Системные учётные записи используются для запуска служб, таких как DNS, почтовые и веб-серверы. Причина этого — безопасность: если бы все службы работали от имени суперпользователя, они могли бы действовать без ограничений.

Примеры системных учётных записей: `daemon`, `operator`, `bind`, `news` и `www`.

`nobody` — это общая непривилегированная системная учётная запись. Однако чем больше сервисов используют `nobody`, тем больше файлов и процессов будут связаны с этим пользователем, и, следовательно, тем больше привилегий он получит.

Пользовательские учётные записи

Пользовательские учётные записи назначаются реальным людям и используются для входа в систему и работы с ней. Каждый человек, получающий доступ к системе, должен иметь уникальную учётную запись. Это позволяет администратору отслеживать, кто и что делает, и предотвращает возможность изменения пользователями настроек друг друга.

Каждый пользователь может настроить свою собственную среду для удобной работы с системой, изменив настройки оболочки по умолчанию, редактора, сочетаний клавиш и языковых параметров.

Каждая учётная запись пользователя в системе FreeBSD имеет определённую связанную с ней информацию:

Имя пользователя

Имя пользователя вводится в приглашении `login:`. У каждого пользователя должно быть уникальное имя. Существует ряд правил для создания допустимых имён пользователей, которые описаны в passwd(5). Рекомендуется использовать имена, состоящие из восьми или менее строчных символов, для обеспечения обратной совместимости с приложениями.

Пароль

У каждой учётной записи есть связанный с ней пароль.

Идентификатор пользователя (UID)

Идентификатор пользователя (UID) — это число, используемое для однозначной идентификации пользователя в системе FreeBSD. Команды, которые позволяют указать

имя пользователя, сначала преобразуют его в UID. Рекомендуется использовать UID меньше 65535, так как более высокие значения могут вызвать проблемы совместимости с некоторым программным обеспечением.

Идентификатор группы (GID)

Идентификатор группы (GID) — это число, используемое для однозначной идентификации основной группы, к которой принадлежит пользователь. Группы представляют собой механизм управления доступом к ресурсам на основе GID пользователя, а не его UID. Это может значительно уменьшить размер некоторых конфигурационных файлов и позволяет пользователям быть членами более чем одной группы. Рекомендуется использовать GID 65535 или меньше, так как более высокие GID могут привести к неработоспособности некоторых программ.

Класс входа (Login class)

Классы входа являются расширением механизма групп, предоставляющим дополнительную гибкость при настройке системы для различных пользователей. Подробнее классы входа рассматриваются в Настройка классов входа.

Время изменения пароля

По умолчанию пароли не имеют срока действия. Однако можно включить истечение срока действия пароля для отдельных пользователей, что заставит некоторых или всех пользователей изменить свои пароли по истечении определённого времени.

Время истечения срока действия учётной записи

По умолчанию FreeBSD не ограничивает срок действия учётных записей. При создании учётных записей с ограниченным сроком действия, например, для учащихся в школе, укажите дату истечения срока с помощью pw(8). После истечения указанного времени учётная запись не может быть использована для входа в систему, хотя каталоги и файлы этой учётной записи останутся.

Полное имя пользователя

Имя пользователя уникально идентифицирует учётную запись в FreeBSD, но не обязательно отражает реальное имя пользователя. Как и в случае с комментарием, эта информация может содержать пробелы, заглавные буквы и быть длиннее 8 символов.

Домашний каталог

Домашний каталог — это полный путь к каталогу в системе. Это стартовый каталог пользователя при входе в систему. Общепринятое соглашение — размещать все домашние каталоги пользователей в `/home/имя_пользователя` или `/usr/home/имя_пользователя`. Каждый пользователь хранит свои личные файлы и подкаталоги в собственном домашнем каталоге.

Оболочка пользователя

Оболочка предоставляет пользователю среду по умолчанию для взаимодействия с системой. Существует множество различных видов оболочек, и опытные пользователи могут выбрать предпочтительную, настроив её в параметрах своей учётной записи.

Учётная запись суперпользователя

Учётная запись суперпользователя, обычно называемая `root`, используется для управления системой без ограничений в правах. По этой причине её не следует использовать для повседневных задач, таких как отправка и получение почты, обычное изучение системы или программирование.

Суперпользователь, в отличие от других учётных записей, может работать без ограничений, и неправильное использование учётной записи суперпользователя может привести к серьёзным последствиям. Обычные пользователи не могут случайно разрушить операционную систему, поэтому рекомендуется входить в систему под обычной учётной записью и становиться суперпользователем только тогда, когда команде требуются дополнительные привилегии.

Всегда дважды и трижды проверяйте команды, выполняемые от имени суперпользователя, поскольку лишний пробел или пропущенный символ могут привести к невосполнимой потере данных.

Существует несколько способов получения привилегий суперпользователя. Хотя можно войти в систему как `root`, это крайне не рекомендуется.

Вместо этого используйте su(1), чтобы стать суперпользователем. Если при выполнении этой команды указан параметр `-`, пользователь также унаследует окружение пользователя root. Пользователь, выполняющий эту команду, должен состоять в группе `wheel`, иначе команда завершится ошибкой. Пользователь также должен знать пароль учётной записи `root`.

В этом примере пользователь становится суперпользователем только для выполнения команды `make install`, так как этот шаг требует прав суперпользователя. После завершения команды пользователь вводит `exit`, чтобы выйти из учётной записи суперпользователя и вернуться к правам своей учётной записи.

Пример 2. Установка программы от имени суперпользователя

```
% configure
% make
% su -
Password:
# make install
# exit
%
```

Встроенная система su(1) хорошо подходит для отдельных компьютеров или небольших сетей с одним системным администратором. Альтернативой является установка пакета security/sudo или порта. Это программное обеспечение обеспечивает журналирование действий и позволяет администратору настроить, какие пользователи могут выполнять определённые команды с правами суперпользователя.

3.3.2. Управление учётными записями

FreeBSD предоставляет различные команды для управления учётными записями пользователей. Наиболее распространённые команды перечислены в Утилиты для управления учётными записями пользователей, а также приведены примеры их использования. Для получения дополнительной информации и примеров использования обратитесь к руководству (man) каждой утилиты.

Таблица 2. Утилиты для управления учётными записями пользователей

Команда	Краткое содержание
adduser(8)	Рекомендуемое приложение командной строки для добавления новых пользователей.
rmuser(8)	Рекомендуемое приложение командной строки для удаления пользователей.
chpass(1)	Гибкий инструмент для изменения информации в пользовательской базе данных.
passwd(1)	Инструмент командной строки для изменения паролей пользователей.
pw(8)	Мощный и гибкий инструмент для изменения всех аспектов пользовательских учётных записей.
bsdconfig(8)	Утилита для настройки системы с поддержкой управления учётными записями.

Добавление пользователя

Рекомендуемая программа для добавления новых пользователей — adduser(8). При добавлении нового пользователя эта программа автоматически обновляет `/etc/passwd` и `/etc/group`. Она также создаёт домашний каталог для нового пользователя, копирует в него стандартные конфигурационные файлы из `/usr/share/skel` и может отправить новому пользователю приветственное письмо. Эту утилиту необходимо запускать от имени суперпользователя.

Утилита adduser(8) интерактивна и проводит пользователя через шаги создания новой учётной записи. Как показано в Добавление пользователя в FreeBSD, можно ввести необходимую информацию или нажать `Enter`, чтобы принять значение по умолчанию, указанное в квадратных скобках. В этом примере пользователь был добавлен в группу `wheel`, что позволяет ему стать суперпользователем с помощью su(1). По завершении утилита предложит создать ещё одного пользователя или выйти.

Пример 3. Добавление пользователя в FreeBSD

```
# adduser
```

Вывод должен быть похож на следующий:

```
Username: jru
Full name: J. Random User
Uid (Leave empty for default):
Login group [jru]:
Login group is jru. Invite jru into other groups? []: wheel
Login class [default]:
Shell (sh csh tcsh zsh nologin) [sh]: zsh
Home directory [/home/jru]:
Home directory permissions (Leave empty for default):
Use password-based authentication? [yes]:
Use an empty password? (yes/no) [no]:
Use a random password? (yes/no) [no]:
Enter password:
Enter password again:
Lock out the account after creation? [no]:
Username   : jru
Password   : ****
Full Name  : J. Random User
Uid        : 1001
Class      :
Groups     : jru wheel
Home       : /home/jru
Shell      : /usr/local/bin/zsh
Locked     : no
OK? (yes/no): yes
adduser: INFO: Successfully added (jru) to the user database.
Add another user? (yes/no): no
Goodbye!
```

Поскольку пароль не отображается при вводе, будьте внимательны, чтобы не ошибиться при создании учётной записи пользователя.

Удаление пользователя

Чтобы полностью удалить пользователя из системы, выполните команду rmuser(8) с правами суперпользователя. Эта команда выполняет следующие действия:

1. Удаляет запись пользователя в crontab(1), если она существует.
2. Удаляет все задания at(1), принадлежащие пользователю.
3. Посылает сигнал SIGKILL всем процессам, принадлежащим пользователю.
4. Удаляет пользователя из локального файла паролей системы.
5. Удаляет домашний каталог пользователя (если он принадлежит пользователю), включая обработку символических ссылок в пути к фактическому домашнему каталогу.
6. Удаляет входящие почтовые файлы пользователя из `/var/mail`.

7. Удаляет все файлы, принадлежащие пользователю, из `/tmp`, `/var/tmp` и `/var/tmp/vi.recover`.
8. Удаляет имя пользователя из всех групп, к которым он принадлежит в `/etc/group`. (Если группа становится пустой и имя группы совпадает с именем пользователя, группа удаляется; это дополняет функциональность adduser(8) в части уникальных групп для каждого пользователя.)
9. Удаляет все очереди сообщений, сегменты разделяемой памяти и семафоры, принадлежащие пользователю.

rmuser(8) не может быть использован для удаления учётных записей суперпользователя, так как это почти всегда свидетельствует о катастрофических последствиях.

По умолчанию используется интерактивный режим, как показано в следующем примере.

Пример 4. `rmuser` Интерактивное удаление учётной записи

```
# rmuser jru
```

Вывод должен быть похож на следующий:

```
Matching password entry:
jru:*:1001:1001::0:0:J. Random User:/home/jru:/usr/local/bin/zsh
Is this the entry you wish to remove? y
Remove user's home directory (/home/jru)? y
Removing user (jru): mailspool home passwd.
```

Изменить информацию о пользователе

Любой пользователь может использовать chpass(1) для изменения своей оболочки по умолчанию и персональной информации, связанной с его учётной записью. Суперпользователь может использовать эту утилиту для изменения дополнительных параметров учётной записи любого пользователя.

Без указания опций, за исключением необязательного имени пользователя, chpass(1) отображает редактор с информацией о пользователе. После выхода пользователя из редактора база данных пользователей обновляется новой информацией.

Этота утилита запрашивает пароль пользователя при выходе из редактора, если только она не запущена от имени суперпользователя.

В Использование `chpass` суперпользователем, суперпользователь ввел `chpass jru` и теперь видит поля, которые можно изменить для этого пользователя. Если же команду выполнит сам `jru`, будут отображены и доступны для редактирования только последние шесть полей. Это показано в Использование `chpass` обычным пользователем.

Пример 5. Использование chpass с правами суперпользователя

```
# chpass jru
```

Вывод должен быть похож на следующий:

```
# Changing user database information for jru.
Login: jru
Password: *
Uid [#]: 1001
Gid [# or name]: 1001
Change [month day year]:
Expire [month day year]:
Class:
Home directory: /home/jru
Shell: /usr/local/bin/zsh
Full Name: J. Random User
Office Location:
Office Phone:
Home Phone:
Other information:
```

Пример 6. Использование chpass обычным пользователем

```
#Changing user database information for jru.
Shell: /usr/local/bin/zsh
Full Name: J. Random User
Office Location:
Office Phone:
Home Phone:
Other information:
```

Команды chfn(1) и chsh(1) являются ссылками на chpass(1), как и ypchpass(1), ypchfn(1) и ypchsh(1). Поскольку поддержка NIS автоматическая, указывать yp перед командой не требуется. Настройка NIS описана в Сетевые серверы.

Изменение пароля пользователя

Любой пользователь может легко изменить свой пароль с помощью passwd(1). Для предотвращения случайных или несанкционированных изменений эта команда запросит исходный пароль пользователя перед установкой нового пароля:

Пример 7. Смена собственного пароля

```
% passwd
```

Вывод должен быть похож на следующий:

```
Changing local password for jru.
Old password:
New password:
Retype new password:
passwd: updating the database...
passwd: done
```

Суперпользователь может изменить пароль любого пользователя, указав имя пользователя при запуске passwd(1). При запуске этой утилиты с правами суперпользователя она не запрашивает текущий пароль пользователя. Это позволяет изменить пароль, если пользователь не может вспомнить исходный пароль.

Пример 8. Изменение пароля другого пользователя суперпользователем

```
# passwd jru
```

Вывод должен быть похож на следующий:

```
Changing local password for jru.
New password:
Retype new password:
passwd: updating the database...
passwd: done
```

Как и chpass(1), yppasswd(1) является ссылкой на passwd(1), поэтому NIS работает с любой из этих команд.

Создание, удаление, изменение и отображение пользователей и групп системы

Утилита pw(8) может создавать, удалять, изменять и отображать пользователей и группы. Она работает как интерфейс к системным файлам пользователей и групп. pw(8) обладает мощным набором параметров командной строки, что делает её пригодной для использования в shell-скриптах, но новым пользователям она может показаться сложнее других команд, представленных в этом разделе.

3.3.3. Управление группами

Группа — это список пользователей. Группа идентифицируется по своему названию и GID. В FreeBSD ядро использует UID процесса и список групп, к которым он принадлежит, чтобы определить, что процесс может делать. В большинстве случаев под GID пользователя или процесса обычно подразумевается первая группа в списке.

Соответствие имён групп и GID перечислено в `/etc/group`. Это простой текстовый файл с четырьмя полями, разделёнными двоеточиями. Первое поле — имя группы, второе — зашифрованный пароль, третье — GID, а четвертое — список участников, разделённых запятыми. Полное описание синтаксиса смотрите в group(5).

Суперпользователь может изменить `/etc/group` с помощью текстового редактора, однако предпочтительнее редактировать файл групп с помощью vigr(8), так как это позволяет избежать некоторых распространённых ошибок. Или для добавления и редактирования групп можно использовать pw(8). Например, чтобы добавить группу с именем `teamtwo` и затем убедиться, что она существует:

При использовании группы operator необходимо соблюдать осторожность, так как могут быть предоставлены непредусмотренные привилегии, аналогичные правам суперпользователя, включая, но не ограничиваясь, возможностью выключения системы, перезагрузки и доступа ко всем элементам в `/dev` в рамках группы.

Пример 9. Добавление группы с помощью pw(8)

```
# pw groupadd teamtwo
# pw groupshow teamtwo
```

Вывод должен быть похож на следующий:

```
teamtwo:*:1100:
```

В этом примере `1100` — это GID группы `teamtwo`. На данный момент в `teamtwo` нет участников. Эта команда добавит `jru` в качестве участника `teamtwo`.

Пример 10. Добавление пользовательских учётных записей в новую группу с помощью pw(8)

```
# pw groupmod teamtwo -M jru
# pw groupshow teamtwo
```

Вывод должен быть похож на следующий:

```
teamtwo:*:1100:jru
```

Аргумент `-M` представляет собой разделённый запятыми список пользователей, которые будут добавлены в новую (пустую) группу или заменят членов существующей группы. Для пользователя это членство в группе отличается от (и дополняет) его основной группы, указанной в файле паролей. Это означает, что пользователь не будет отображаться как член группы при использовании `groupshow` с pw(8), но будет виден при запросе информации через id(1) или аналогичный инструмент. Когда pw(8) используется для добавления пользователя в группу, он только изменяет `/etc/group` и не пытается читать дополнительные данные из `/etc/passwd`.

Пример 11. Добавление нового участника в группу с помощью pw(8)

```
# pw groupmod teamtwo -m db
# pw groupshow teamtwo
```

Вывод должен быть похож на следующий:

```
teamtwo:*:1100:jru,db
```

В этом примере аргумент `-m` представляет собой список пользователей, разделённых запятыми, которые будут добавлены в группу. В отличие от предыдущего примера, эти пользователи добавляются к группе и не заменяют уже существующих в ней пользователей.

Пример 12. Использование id(1) для определения членства в группе

```
% id jru
```

Вывод должен быть похож на следующий:

```
uid=1001(jru) gid=1001(jru) groups=1001(jru), 1100(teamtwo)
```

В этом примере `jru` является участником групп `jru` и `teamtwo`.

Для получения дополнительной информации об этой команде и формате файла `/etc/group` обратитесь к pw(8) и group(5).

3.4. Разрешения

В FreeBSD каждому файлу и каталогу сопоставлен набор прав доступа, и существует несколько утилит для просмотра и изменения этих прав. Понимание работы прав доступа необходимо для того, чтобы пользователи могли получать доступ к нужным им файлам, но не имели возможности несанкционированного доступа к файлам операционной системы или файлам других пользователей.

В этом разделе рассматриваются традиционные права доступа UNIX®, используемые в FreeBSD. Для более детального управления доступом к файловой системе обратитесь к разделу Списки контроля доступа.

В UNIX® базовые права доступа назначаются с использованием трёх типов доступа: чтение, запись и выполнение. Эти типы доступа определяют права на файл для владельца, группы и остальных (всех остальных). Права на чтение, запись и выполнение могут быть представлены буквами `r`, `w` и `x`. Они также могут быть представлены в виде двоичных чисел, так как каждое право либо включено, либо выключено (`0`). При представлении в виде чисел порядок всегда читается как `rwx`, где `r` имеет значение `4`, `w` — `2`, а `x` — `1`.

Таблица 4.1 обобщает возможные числовые и буквенные варианты. В столбце "Список файлов каталога" символ `-` используется для обозначения отключённого разрешения.

Таблица 3. Права доступа UNIX®

Значение	Разрешение	Список файлов каталога
0	Нет чтения, нет записи, нет выполнения	`---`
1	Нет чтения, нет записи, выполнение	`--x`
2	Нет чтения, записи, выполнения	`-w-`
3	Нет прав на чтение, запись, выполнение	`-wx`
4	Чтение, нет записи, нет выполнения	`r--`
5	Чтение, нет записи, выполнение	`r-x`
6	Чтение, запись, без выполнения	`rw-`
7	Чтение, запись, выполнение	`rwx`

Используйте аргумент `-l` с ls(1) для просмотра длинного списка каталога, который включает колонку с информацией о правах доступа к файлу для владельца, группы и всех остальных. Например, `ls -l` в произвольном каталоге может вывести:

```
% ls -l
```

Вывод должен быть похож на следующий:

```
total 530
-rw-r--r--  1 root  wheel     512 Sep  5 12:31 myfile
-rw-r--r--  1 root  wheel     512 Sep  5 12:31 otherfile
```

```
-rw-r--r--  1 root  wheel    7680 Sep  5 12:31 email.txt
```

Рассматривая строку для файла `myfile`, первый `(крайний слева)` символ указывает, является ли этот файл обычным файлом, каталогом, специальным символьным устройством, сокетом или любым другим специальным псевдофайловым устройством. В данном примере символ `-` обозначает обычный файл. Следующие три символа, в этом примере `rw-`, указывают права для владельца файла. Следующие три символа, `r--`, указывают права для группы, к которой принадлежит файл. Последние три символа, `r--`, указывают права для всех остальных. Дефис означает, что право отключено. В этом примере права настроены так, что владелец может читать и записывать файл, группа может читать файл, а все остальные могут только читать файл. Согласно приведённой выше таблице, права для этого файла будут `644`, где каждая цифра представляет одну из трёх частей прав файла.

Как система управляет правами доступа к устройствам? FreeBSD рассматривает большинство аппаратных устройств как файлы, которые программы могут открывать, читать и записывать в них данные. Эти специальные файлы устройств хранятся в `/dev/`.

Каталоги также рассматриваются как файлы. У них есть права на чтение, запись и выполнение. Бит выполнения для каталога имеет несколько иное значение, чем для файлов. Если каталог помечен как исполняемый, это означает, что в него можно перейти с помощью cd(1). Это также означает, что можно получить доступ к файлам внутри этого каталога, в зависимости от прав на сами файлы.

Для выполнения списка файлов в каталоге необходимо установить право на чтение для этого каталога. Чтобы удалить файл, имя которого известно, требуются права на запись *и* выполнение для каталога, содержащего этот файл.

Существуют и другие биты разрешений, но они в основном используются в особых случаях, таких как setuid-бинарники и sticky-каталоги. Для получения дополнительной информации о правах доступа к файлам и их настройке обратитесь к chmod(1).

3.4.1. Символьные права доступа

Символьные права доступа используют символы вместо восьмеричных значений для назначения прав доступа файлам или каталогам. Синтаксис символьных прав доступа имеет вид (кто) (действие) (права доступа), где доступны следующие значения:

Опция	**Буква**	**Представляет**
(кто)	u	Пользователь
(кто)	g	Владелец группы
(кто)	o	Другие
(кто)	a	Все (all, "world")
(действие)	+	Добавление разрешений
(действие)	-	Удаление прав доступа
(действие)	=	Явно заданные разрешения

Опция	Буква	Представляет
(права доступа)	r	Чтение
(права доступа)	w	Запись
(права доступа)	x	Выполнение
(права доступа)	t	Бит закрепления (sticky)
(права доступа)	s	Установка UID или GID

Эти значения используются с chmod(1), но с буквами вместо чисел. Например, следующая команда запретит доступ к *FILE* как членам группы, связанной с *FILE*, так и всем остальным пользователям:

```
% chmod go= FILE
```

Список, разделённый запятыми, может быть указан, если к файлу необходимо применить более одного изменения. Например, следующая команда удаляет разрешение на запись для группы и всех остальных пользователей у *FILE*, а также добавляет разрешение на выполнение для всех:

```
% chmod go-w,a+x FILE
```

3.4.2. Флаги файлов FreeBSD

В дополнение к правам доступа к файлам FreeBSD поддерживает использование "флагов файлов". Эти флаги добавляют дополнительный уровень безопасности и контроля над файлами, но не над каталогами. С помощью флагов файлов даже пользователь `root` может быть лишён возможности удалять или изменять файлы.

Флаги файлов изменяются с помощью chflags(1). Например, чтобы установить системный флаг «неудаляемый» для файла `file1`, выполните следующую команду:

```
# chflags sunlink file1
```

Чтобы отключить системный флаг "неудаляемый", добавьте "no" перед `sunlink`:

```
# chflags nosunlink file1
```

Чтобы просмотреть флаги файла, используйте `-lo` с ls(1):

```
# ls -lo file1
```

```
-rw-r--r--  1 trhodes  trhodes  sunlnk 0 Mar  1 05:54 file1
```

Некоторые флаги файлов могут быть добавлены или удалены только пользователем `root`. В остальных случаях владелец файла может устанавливать его флаги. Дополнительную информацию можно найти в chflags(1) и chflags(2).

3.4.3. Установленные биты setuid, setgid и sticky

Помимо уже рассмотренных прав доступа, существуют три специальных параметра, которые должны знать все администраторы. Это права `setuid`, `setgid` и `sticky`.

Эти настройки важны для некоторых операций UNIX®, так как предоставляют функциональность, обычно недоступную обычным пользователям. Чтобы понять их, необходимо отметить разницу между реальным идентификатором пользователя (real user ID) и эффективным идентификатором пользователя (effective user ID).

Реальный идентификатор пользователя (UID) — это UID, который владеет или запускает процесс. Эффективный идентификатор пользователя — это UID, от имени которого выполняется процесс. Например, passwd(1) запускается с реальным UID, когда пользователь меняет свой пароль. Однако для обновления базы данных паролей команда выполняется с эффективным UID пользователя `root`. Это позволяет пользователям изменять свои пароли без ошибки `Permission Denied`.

Разрешение setuid может быть добавлено символически путем добавления права `s` для пользователя, как в следующем примере:

```
# chmod u+s suidexample.sh
```

Права setuid также могут быть установлены путем добавления числа четыре (4) перед набором прав, как показано в следующем примере:

```
# chmod 4755 suidexample.sh
```

Разрешения для файла `suidexample.sh` теперь выглядят следующим образом:

```
-rwsr-xr-x   1 trhodes  trhodes    63 Aug 29 06:36 suidexample.sh
```

Обратите внимание, что `s` теперь является частью набора прав для владельца файла, заменив бит исполнения. Это позволяет создавать программы, требующие повышенных прав, таких как passwd(1).

Опция `nosuid` в mount(8) приведёт к тихому отказу таких бинарных файлов без уведомления пользователя. Эта опция не полностью надёжна, так как обёртка `nosuid` может её обойти.

Чтобы наблюдать это в реальном времени, откройте два терминала. В одном введите `passwd` как обычный пользователь. Пока команда ожидает ввода нового пароля, проверьте таблицу процессов и посмотрите информацию о пользователе для passwd(1):

В терминале A:

```
Changing local password for trhodes
Old Password:
```

В терминале B:

```
# ps aux | grep passwd
```

```
trhodes  5232  0.0  0.2  3420  1608   0  R+    2:10AM   0:00.00 grep passwd
root     5211  0.0  0.2  3620  1724   2  I+    2:09AM   0:00.01 passwd
```

Хотя passwd(1) запускается от имени обычного пользователя, он использует эффективный UID `root`.

Разрешение `setgid` выполняет ту же функцию, что и разрешение `setuid`, за исключением того, что оно изменяет настройки группы. Когда приложение или утилита запускается с этой настройкой, оно получает разрешения в соответствии с группой, которой принадлежит файл, а не пользователем, запустившим процесс.

Чтобы установить право `setgid` на файл в символьном виде, добавьте право `s` для группы с помощью chmod(1):

```
# chmod g+s sgidexample.sh
```

В качестве альтернативы укажите в chmod(1) ведущую двойку (2):

```
# chmod 2755 sgidexample.sh
```

В следующем листинге обратите внимание, что символ `s` теперь находится в поле, предназначенном для настроек прав группы:

```
-rwxr-sr-x   1 trhodes  trhodes    44 Aug 31 01:49 sgidexample.sh
```

В этих примерах, даже если рассматриваемый shell-скрипт является исполняемым файлом, он не будет запускаться с другим EUID или эффективным идентификатором пользователя. Это происходит потому, что shell-скрипты не могут использовать системные вызовы setuid(2).

`setuid` и `setgid` биты прав могут снизить безопасность системы, предоставляя повышенные привилегии. Третий специальный бит прав, `sticky bit`, напротив, может повысить безопасность системы.

Когда `sticky bit` установлен на каталоге, он разрешает удаление файлов только владельцу файла. Это полезно для предотвращения удаления файлов в общедоступных каталогах, таких как `/tmp`, пользователями, не являющимися владельцами файла. Чтобы использовать этот режим доступа, добавьте режим `t` к файлу:

```
# chmod +t /tmp
```

В качестве альтернативы можно добавить единицу (1) перед набором прав доступа:

```
# chmod 1777 /tmp
```

Разрешение `sticky bit` отображается как `t` в самом конце набора разрешений:

```
# ls -al / | grep tmp
```

```
drwxrwxrwt  10 root  wheel         512 Aug 31 01:49 tmp
```

3.5. Структура каталогов

Иерархия каталогов FreeBSD является основой для общего понимания системы. Наиболее важным каталогом является корневой или "/". Этот каталог монтируется первым при загрузке и содержит базовую систему, необходимую для подготовки операционной системы к многопользовательскому режиму. Корневой каталог также содержит точки монтирования для других файловых систем, которые монтируются во время перехода в многопользовательский режим.

Точка монтирования — это каталог, в который можно подключить дополнительные файловые системы к родительской файловой системе (обычно корневой файловой системе). Это подробно описано в разделе Организация дисков. Стандартные точки монтирования включают `/usr/`, `/var/`, `/tmp/`, `/mnt/` и `/media/`. Эти каталоги обычно связаны с записями в файле `/etc/fstab`. Этот файл представляет собой таблицу различных файловых систем и точек монтирования, которая читается системой. Большинство файловых систем в `/etc/fstab` автоматически монтируются при загрузке с помощью скрипта rc(8), если их запись не содержит параметра `noauto`. Подробности можно найти в разделе Файл fstab.

Полное описание иерархии файловой системы доступно в hier(7). В следующей таблице представлен краткий обзор наиболее распространённых каталогов.

Каталог	Описание
`/`	Корневой каталог файловой системы.

`/bin/`	Пользовательские утилиты, основные как для однопользовательской, так и для многопользовательской среды.
`/boot/`	Программы и конфигурационные файлы, используемые при загрузке операционной системы.
`/boot/defaults/`	Файлы конфигурации загрузки по умолчанию. Подробности смотрите в loader.conf(5).
`/dev/`	Специальные файлы устройств, управляемые devfs(5)
`/etc/`	Системные конфигурационные файлы и скрипты.
`/etc/defaults/`	Файлы конфигурации системы по умолчанию. Подробности смотрите в rc(8).
`/etc/periodic/`	Скрипты, выполняемые ежедневно, еженедельно и ежемесячно, через cron(8). Подробности смотрите в periodic(8).
`/lib/`	Критические системные библиотеки, необходимые для бинарных файлов в `/bin` и `/sbin`
`/libexec/`	Критические системные файлы
`/media/`	Содержит подкаталоги, предназначенные для использования в качестве точек монтирования съемных носителей, таких как компакт-диски, USB-накопители и дискеты
`/mnt/`	Пустой каталог, обычно используемый системными администраторами в качестве временной точки монтирования.
`/net/`	Автомонтируемые NFS-ресурсы; см. auto_master(5)
`/proc/`	Файловая система процессов. Подробности смотрите в procfs(5).
`/rescue/`	Статически связанные программы для аварийного восстановления, как описано в rescue(8).
`/root/`	Домашний каталог для учётной записи `root`.
`/sbin/`	Системные программы и административные утилиты, необходимые как для однопользовательского, так и для многопользовательского окружения.
`/tmp/`	Временные файлы, которые обычно *не* сохраняются после перезагрузки системы. Часто в `/tmp` монтируется файловая система в памяти. Это можно автоматизировать, используя переменные, связанные с tmpmfs, из rc.conf(5), или добавив запись в `/etc/fstab`; подробности смотрите в mdmfs(8).
`/usr/`	Большинство пользовательских утилит и приложений.
`/usr/bin/`	Общие утилиты, инструменты программирования и приложения.
`/usr/include/`	Стандартные включаемые файлы языка C.
`/usr/lib/`	Библиотеки архивов.
`/usr/libdata/`	Различные вспомогательные файлы данных.

/usr/libexec/	Системные демоны и системные утилиты, выполняемые другими программами.
/usr/local/	Локальные исполняемые файлы и библиотеки. Также используется как путь по умолчанию для установки портов FreeBSD. Внутри /usr/local должна использоваться общая структура каталогов, описанная в hier(7) для /usr. Исключениями являются каталог man, который находится непосредственно в /usr/local, а не в /usr/local/share, а документация портов располагается в share/doc/port.
/usr/ports/	Коллекция портов FreeBSD (опционально).
/usr/sbin/	Системные демоны и системные утилиты, выполняемые пользователями.
/usr/share/	Архитектурно-независимые файлы.
/usr/src/	BSD и/или локальные исходные файлы.
/var/	Многоцелевые файлы журналов, временные, транзиентные и файлы очередей.
/var/log/	Различные системные журналы.
/var/tmp/	Временные файлы, которые обычно сохраняются после перезагрузки системы.

3.6. Организация диска

Наименьшей единицей организации, которую FreeBSD использует для поиска файлов, является имя файла. Имена файлов чувствительны к регистру, что означает, что readme.txt и README.TXT — это два разных файла. FreeBSD не использует расширение файла для определения, является ли файл программой, документом или каким-либо другим видом данных.

Файлы хранятся в каталогах. Каталог может не содержать файлов или содержать сотни файлов. Каталог также может содержать другие каталоги, что позволяет создавать иерархию вложенных каталогов для организации данных.

Файлы и каталоги указываются путём перечисления имени файла или каталога, за которым следует косая черта /, а затем — при необходимости — другие имена каталогов. Например, если каталог foo содержит каталог bar, в котором находится файл readme.txt, то полное имя или *путь* к файлу будет foo/bar/readme.txt. Обратите внимание, что это отличается от Windows®, где для разделения имён файлов и каталогов используется \. В FreeBSD не используются буквы дисков или другие обозначения накопителей в пути. Например, в FreeBSD не указывают путь вида c:\foo\bar\readme.txt.

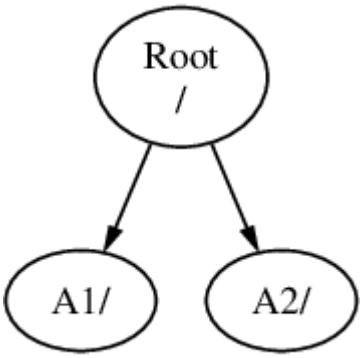
3.6.1. Файловые системы

Каталоги и файлы хранятся в файловой системе. Каждая файловая система содержит ровно один каталог на самом верхнем уровне, называемый *корневым каталогом* для этой

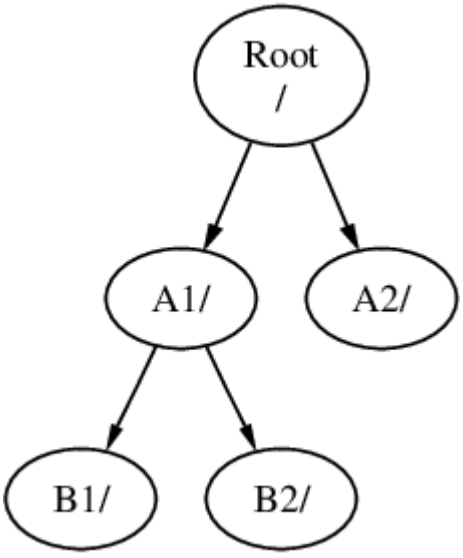
файловой системы. Этот корневой каталог может содержать другие каталоги. Одна файловая система назначается *корневой файловой системой* или `/`. Все остальные файловые системы *монтируются* в корневую файловую систему. Независимо от количества дисков в системе FreeBSD, каждый каталог выглядит как часть одного и того же диска.

Рассмотрим три файловые системы: `A`, `B` и `C`. Каждая файловая система имеет один корневой каталог, который содержит два других каталога с именами `A1`, `A2` (аналогично `B1`, `B2` и `C1`, `C2`).

Назовем корневую файловую систему `A`. Если использовать ls(1) для просмотра содержимого этого каталога, будут видны два подкаталога, `A1` и `A2`. Дерево каталогов выглядит следующим образом:

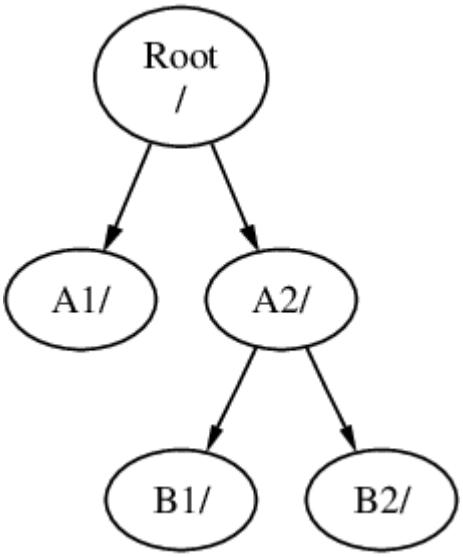


Файловая система должна быть смонтирована в каталог другой файловой системы. При монтировании файловой системы `B` в каталог `A1` корневой каталог `B` заменяет `A1`, а каталоги в `B` отображаются соответствующим образом:



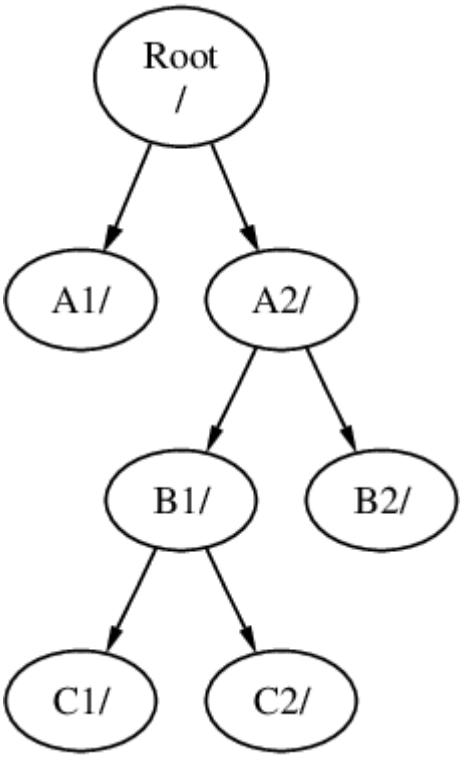
Любые файлы в каталогах `B1` или `B2` доступны по пути `/A1/B1` или `/A1/B2` соответственно. Файлы, которые находились в `/A1`, временно скрыты. Они снова станут доступны, если `B` будет *отмонтирован* из `A`.

Если бы `B` был смонтирован на `A2`, то диаграмма выглядела бы так:

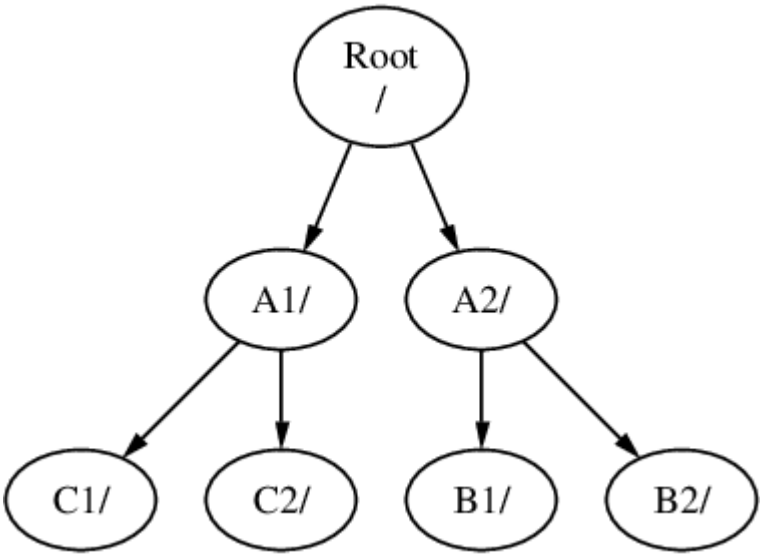


и пути будут `/A2/B1` и `/A2/B2` соответственно.

Файловые системы могут монтироваться одна поверх другой. Продолжая последний пример, файловая система `C` может быть смонтирована поверх каталога `B1` в файловой системе `B`, что приведёт к следующей структуре:



Или `C` может быть непосредственно смонтирована в файловую систему `A`, в каталог `A1`:



Вполне возможно иметь одну большую корневую файловую систему и не создавать других. У такого подхода есть несколько недостатков и одно преимущество.

Преимущества нескольких файловых систем

- Различные файловые системы могут иметь разные *параметры монтирования*. Например, корневая файловая система может быть смонтирована в режиме только для чтения, что предотвращает случайное удаление или редактирование важных файлов пользователями. Разделение файловых систем, доступных для записи пользователями, таких как `/home`, от остальных позволяет монтировать их с опцией *nosuid*. Эта опция предотвращает действие битов *suid*/*guid* у исполняемых файлов, хранящихся в данной файловой системе, что может повысить безопасность.
- FreeBSD автоматически оптимизирует расположение файлов в файловой системе в зависимости от того, как она используется. Таким образом, файловая система, содержащая множество часто записываемых небольших файлов, будет оптимизирована иначе, чем система с меньшим количеством более крупных файлов. При использовании одной большой файловой системы такая оптимизация нарушается.
- Файловые системы FreeBSD устойчивы к потере питания. Однако отключение питания в критический момент всё же может повредить структуру файловой системы. Разделение данных между несколькими файловыми системами увеличивает вероятность успешной загрузки системы, упрощая восстановление из резервной копии при необходимости.

Преимущество единой файловой системы

- Файловые системы имеют фиксированный размер. Если файловая система создаётся при установке FreeBSD и ей задаётся определённый размер, позже может оказаться, что раздел нужно увеличить. Это нелегко сделать без резервного копирования, пересоздания файловой системы с новым размером и последующего восстановления данных из резервной копии.

FreeBSD предоставляет команду growfs(8), которая позволяет увеличивать размер файловой системы на лету, устраняя это ограничение. Файловая система может быть расширена только в свободное пространство раздела, в котором она находится. Если после

раздела есть место, сам раздел можно расширить с помощью gpart(8). Если раздел является последним на виртуальном диске и диск расширен, то раздел также можно расширить.

3.6.2. Разделы диска

Файловые системы содержатся в *разделах*. Диски разделяются на разделы с использованием одной из нескольких схем разметки; см. Ручная разметка разделов. Более новая схема — GPT; старые компьютеры на базе BIOS используют MBR. GPT поддерживает разделение диска на разделы с указанием размера, смещения и типа. Она поддерживает большое количество разделов и их типов, и рекомендуется к использованию везде, где это возможно. Разделы GPT используют имя диска с суффиксом, где суффикс `p1` соответствует первому разделу, `p2` — второму и так далее. MBR, однако, поддерживает только небольшое количество разделов. Разделы MBR в FreeBSD называются *слайсами*. Слайсы могут использоваться для разных операционных систем. Слайсы FreeBSD дополнительно разбиваются на разделы с использованием меток BSD (см. bsdlabel(8)).

Номера слайсов указываются после имени устройства с префиксом `s` и начинаются с 1. Таким образом, "da0*s1*" — это первый слайс на первом SCSI-диске. На диске может быть только четыре физических слайса, но внутри физических слайсов соответствующего типа могут находиться логические слайсы. Эти расширенные слайсы нумеруются, начиная с 5, поэтому "ada0*s5*" — это первый расширенный слайс на первом SATA-диске. Эти устройства используются файловыми системами, которые предназначены для размещения в слайсе.

Каждый раздел GPT или BSD может содержать только одну файловую систему, что означает, что файловые системы часто описываются либо по их стандартной точке монтирования в иерархии файловых систем, либо по имени раздела, в котором они находятся.

FreeBSD также использует место на диске для *раздела подкачки* (*swap space*), чтобы обеспечить работу *виртуальной памяти*. Это позволяет компьютеру вести себя так, как будто у него больше памяти, чем есть на самом деле. Когда FreeBSD исчерпывает доступную память, она перемещает часть данных, которые в данный момент не используются, в раздел подкачки, а затем возвращает их обратно (перемещая что-то другое), когда они нужны. Этот процесс называется *подкачкой* (*paging*).

Некоторые разделы BSD имеют определённые соглашения, связанные с ними.

Раздел	Соглашение
`a`	Обычно содержит корневую файловую систему.
`b`	Обычно содержит раздел подкачки.
`c`	Обычно имеет тот же размер, что и окружающий слайс. Это позволяет утилитам, которым необходимо работать со всем слайсом, например, сканеру плохих блоков, работать с разделом `c`. Файловая система на этом разделе обычно не создаётся.
`d`	Раздел `d` ранее имел особое значение, но сейчас это ушло в прошлое, и `d` может использоваться как обычный раздел.

Слайсы и «опасно выделенные» (dangerously dedicated) физические диски содержат разделы BSD, которые обозначаются буквами от `a` до `h`. Эта буква добавляется к имени устройства, поэтому «da0*a*» — это раздел `a` на первом диске `da`, который является «опасно выделенным». «ada1s3*e*» — это пятый раздел в третьем срезе второго диска SATA.

Наконец, каждый диск в системе идентифицируется. Имя диска начинается с кода, указывающего тип диска, за которым следует номер, обозначающий конкретный диск. В отличие от разделов и слайсов, нумерация дисков начинается с 0. Распространённые коды перечислены в Имена устройств дисков.

При указании раздела в слайсе укажите имя диска, `s`, номер слайса, а затем букву раздела. Примеры приведены в разделе Примеры имён дисков, слайсов и разделов. Разделы GPT включают имя диска, `p`, а затем номер раздела.

Концептуальная модель диска показывает концептуальную модель разметки диска с использованием разделов MBR.

При установке FreeBSD настройте слайсы диска, если используется MBR, и создайте разделы внутри слайса, который будет использоваться для FreeBSD. Если используется GPT, настройте разделы для каждой файловой системы. В обоих случаях создайте файловую систему или область подкачки в каждом разделе и определите, где будет монтироваться каждая файловая система. Подробности о работе с разделами см. в gpart(8).

Таблица 4. Имена устройств дисков

Тип накопителя	Имя устройства накопителя
SATA и IDE жёсткие диски	`ada`
SCSI жёсткие диски и USB устройства хранения данных	`da`
Хранилище NVMe	`nvd` или `nda`
SATA и IDE приводы CD-ROM	`cd`
SCSI CD-ROM приводы	`cd`
Накопители на гибких дисках	`fd`
SCSI-ленточные накопители	`sa`
RAID-диски	Примеры включают `aacd` для Adaptec® AdvancedRAID, `mlxd` и `mlyd` для Mylex®, `amrd` для AMI MegaRAID®, `idad` для Compaq Smart RAID, `twed` для 3ware® RAID.

Таблица 5. Примеры названий дисков, слайсов и разделов

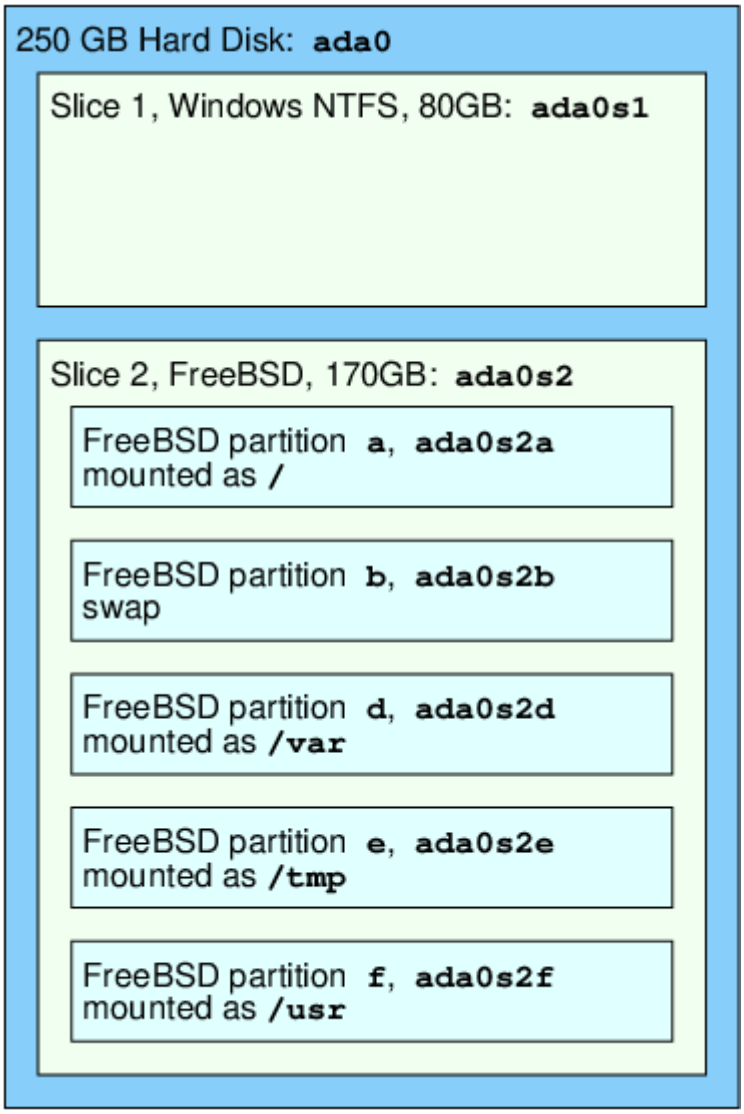
Имя	Значение
`ada0s1a`	Первый раздел (`a`) на первом слайсе (`s1`) на первом SATA-диске (`ada0`).

Имя	Значение
da1s2e	Пятый раздел (e) на втором срезе (s2) второго SCSI-диска (da1).

Пример 13. Концептуальная модель диска

На этой диаграмме показано, как FreeBSD видит первый SATA-диск, подключённый к системе. Предположим, что размер диска составляет 250 ГБ, и он содержит раздел на 80 ГБ и раздел на 170 ГБ (разделы MS-DOS®). Первый раздел содержит файловую систему Windows® NTFS, `C:`, а второй раздел содержит установленную систему FreeBSD. В данном примере установки FreeBSD присутствуют четыре раздела с данными и один раздел подкачки.

Четыре раздела содержат файловые системы. Раздел `a` используется для корневой файловой системы, `d` — для `/var/`, `e` — для `/tmp/`, а `f` — для `/usr/`. Буква раздела `c` относится ко всему слайсу и поэтому не используется для обычных разделов.



3.7. Монтирование и демонтирование файловых систем

Файловую систему удобно визуализировать как дерево, корнем которого является `/`. Каталоги `/dev`, `/usr` и другие в корневом каталоге представляют собой ветви, которые, в свою очередь, могут иметь собственные ветви, например `/usr/local`, и так далее.

Существуют различные причины для размещения некоторых из этих каталогов на отдельных файловых системах. `/var` содержит каталоги `log/`, `spool/` и различные типы временных файлов, поэтому может заполняться. Заполнение корневой файловой системы нежелательно, поэтому часто предпочтительно отделить `/var` от `/`.

Еще одна распространённая причина размещать определённые каталоги на других файловых системах — это необходимость их размещения на отдельных физических дисках или виртуальных дисках, таких как сетевые файловые системы (NFS), описанные в “Сетевая файловая система (NFS)”, или приводы CDROM.

3.7.1. Файл fstab

В процессе загрузки (Процесс загрузки FreeBSD), файловые системы, перечисленные в `/etc/fstab`, автоматически монтируются, за исключением записей, содержащих `noauto`. Этот файл содержит записи в следующем формате:

```
device       /mount-point fstype     options      dumpfreq     passno
```

`device`

Существующее имя устройства, как описано в Имена устройств дисков.

`mount-point`

Существующий каталог, на который монтируется файловая система.

`fstype`

Тип файловой системы, передаваемый в mount(8). Файловая система по умолчанию в FreeBSD — `ufs`.

`options`

`rw` для файловых систем с доступом на чтение и запись или `ro` для файловых систем только для чтения, за которыми могут следовать другие необходимые опции. Часто используется опция `noauto` для файловых систем, которые обычно не монтируются при загрузке. Другие опции перечислены в mount(8).

`dumpfreq`

Используется dump(8) для определения, какие файловые системы требуют дампинга. Если поле отсутствует, предполагается значение ноль.

`passno`

Определяет порядок проверки файловых систем UFS с помощью fsck(8) после

перезагрузки. Файловые системы, которые следует пропускать, должны иметь значение `passno`, равное нулю. Корневая файловая система должна проверяться первой и иметь значение `passno`, равное единице. Остальные файловые системы должны иметь значения больше единицы. Если несколько файловых систем имеют одинаковое значение `passno`, fsck(8) попытается проверять их параллельно, если это возможно.

См. fstab(5) для получения дополнительной информации о формате `/etc/fstab` и его параметрах.

3.7.2. Использование mount(8)

Файловые системы монтируются с помощью mount(8). Базовая синтаксическая конструкция выглядит следующим образом:

```
# mount device mountpoint
```

Файловая система, указанная в `/etc/fstab`, также может быть смонтирована, если указать только точку монтирования.

Эта команда предоставляет множество опций, которые описаны в mount(8). Наиболее часто используемые опции включают:

Параметры монтирования

`-a`

Смонтировать все файловые системы, перечисленные в `/etc/fstab`, за исключением тех, которые помечены как "noauto", исключены флагом `-t` или уже смонтированы.

`-d`

Выполнить все действия, кроме самого системного вызова mount. Эта опция полезна в сочетании с флагом `-v` для определения того, что на самом деле пытается сделать mount(8).

`-f`

Принудительно смонтировать поврежденную файловую систему (опасно) или отозвать права на запись при понижении статуса монтирования файловой системы с чтения-записи на только чтение.

`-r`

Смонтировать файловую систему в режиме только для чтения. Это эквивалентно использованию `-o ro`.

`-t тип_фс`

Смонтировать указанный тип файловой системы или смонтировать только файловые системы данного типа, если включен параметр `-a`. Тип файловой системы "ufs" используется по умолчанию.

`-u`

Обновить параметры монтирования файловой системы.

`-v`

Выдавать более подробную информацию.

`-w`

Смонтировать файловую систему в режиме чтения-записи.

Следующие параметры могут быть переданы в `-o` в виде списка, разделённого запятыми:

`nosuid`

Не интерпретировать флаги setuid или setgid на файловой системе. Это также полезная опция безопасности.

3.7.3. Использование umount(8)

Для размонтирования файловой системы используйте umount(8). Эта команда принимает один параметр, которым может быть точка монтирования, имя устройства, `-a` или `-A`.

Все формы команды принимают параметр `-f` для принудительного размонтирования и `-v` для вывода подробной информации. Учтите, что использование `-f` обычно не рекомендуется, так как это может привести к аварийному завершению работы компьютера или повреждению данных в файловой системе.

Для размонтирования всех смонтированных файловых систем или только файловых систем указанных после `-t` типов используйте `-a` или `-A`. Обратите внимание, что `-A` не пытается размонтировать корневую файловую систему.

3.8. Процессы и демоны

FreeBSD - это многозадачная операционная система. Каждая программа, выполняемая в любой момент времени, называется *процессом*. Каждая запущенная команда создаёт как минимум один новый процесс, и в системе FreeBSD выполняется ряд системных процессов.

Каждый процесс однозначно идентифицируется числом, называемым *идентификатором процесса* (PID). Подобно файлам, каждый процесс имеет владельца и группу, а права владельца и группы используются для определения того, какие файлы и устройства процесс может открыть. Большинство процессов также имеют родительский процесс, который их запустил. Например, оболочка — это процесс, и любая команда, запущенная в оболочке, является процессом, для которого оболочка выступает родительским процессом. Исключением является специальный процесс init(8), который всегда запускается первым при загрузке и всегда имеет PID 1.

Некоторые программы не предназначены для постоянного взаимодействия с пользователем и отключаются от терминала при первой возможности. Например, веб-сервер отвечает на веб-запросы, а не на действия пользователя. Почтовые серверы — ещё один пример таких приложений. Эти программы называются *демонами*. Термин «демон» происходит из греческой мифологии и обозначает сущность, которая не является ни доброй,

ни злой и незаметно выполняет полезные задачи. Именно поэтому талисманом BSD стал улыбающийся демон в кроссовках и с вилами.

Существует соглашение называть программы, которые обычно работают как демоны, с добавлением буквы "d" в конце. Например, BIND — это Berkeley Internet Name Domain, но фактическая программа, которая выполняется, называется `named`. Веб-сервер Apache называется `httpd`, а демон очереди печати — `lpd`. Это всего лишь соглашение об именовании. Например, основной почтовый демон для приложения Sendmail называется `sendmail`, а не `maild`.

3.8.1. Просмотр процессов

Чтобы просмотреть процессы, выполняемые в системе, используйте ps(1) или top(1). Для отображения статичного списка текущих процессов, их PID, объёма используемой памяти и команд, которыми они были запущены, используйте ps(1). Для отображения всех выполняемых процессов с периодическим обновлением списка каждые несколько секунд, чтобы интерактивно наблюдать за работой компьютера, используйте top(1).

По умолчанию ps(1) отображает только команды, выполняемые и принадлежащие текущему пользователю. Например:

```
% ps
```

Вывод должен быть похож на следующий:

```
  PID TT  STAT    TIME COMMAND
 8203  0  Ss   0:00.59 /bin/csh
 8895  0  R+   0:00.00 ps
```

Вывод команды ps(1) организован в несколько столбцов. В столбце `PID` отображается идентификатор процесса. PID начинаются с 1, достигают 99999, затем снова начинаются с начала. Однако, если PID уже используется, он не будет повторно назначен. Столбец `TT` показывает tty, на котором выполняется программа, а `STAT` отображает состояние программы. `TIME` — это время, в течение которого программа выполнялась на CPU. Обычно это не общее время с момента запуска программы, так как большинство программ проводят много времени в ожидании событий, прежде чем им потребуется время на CPU. Наконец, `COMMAND` — это команда, которая использовалась для запуска программы.

Доступно несколько различных опций для изменения отображаемой информации. Один из наиболее полезных наборов — `auxww`, где `a` показывает информацию обо всех запущенных процессах всех пользователей, `u` отображает имя пользователя и использование памяти владельцем процесса, `x` показывает информацию о процессах демонов, а `ww` заставляет ps(1) выводить полную командную строку для каждого процесса вместо её обрезки, когда она становится слишком длинной для экрана.

Вывод команды top(1) выглядит аналогично:

```
% top
```

Вывод должен быть похож на следующий:

```
last pid:  9609;  load averages:  0.56,  0.45,  0.36              up 0+00:20:03
10:21:46
107 processes: 2 running, 104 sleeping, 1 zombie
CPU:  6.2% user,  0.1% nice,  8.2% system,  0.4% interrupt, 85.1% idle
Mem: 541M Active, 450M Inact, 1333M Wired, 4064K Cache, 1498M Free
ARC: 992M Total, 377M MFU, 589M MRU, 250K Anon, 5280K Header, 21M Other
Swap: 2048M Total, 2048M Free

  PID USERNAME    THR PRI NICE   SIZE    RES STATE   C   TIME   WCPU COMMAND
  557 root          1 -21  r31   136M 42296K select  0   2:20  9.96% Xorg
 8198 dru           2  52    0   449M 82736K select  3   0:08  5.96% kdeinit4
 8311 dru          27  30    0  1150M   187M uwait   1   1:37  0.98% firefox
  431 root          1  20    0 14268K  1728K select  0   0:06  0.98% moused
 9551 dru           1  21    0 16600K  2660K CPU3    3   0:01  0.98% top
 2357 dru           4  37    0   718M   141M select  0   0:21  0.00% kdeinit4
 8705 dru           4  35    0   480M    98M select  2   0:20  0.00% kdeinit4
 8076 dru           6  20    0   552M   113M uwait   0   0:12  0.00% soffice.bin
 2623 root          1  30   10 12088K  1636K select  3   0:09  0.00% powerd
 2338 dru           1  20    0   440M 84532K select  1   0:06  0.00% kwin
 1427 dru           5  22    0   605M 86412K select  1   0:05  0.00% kdeinit4
```

Вывод разделён на две части. Заголовок (первые пять или шесть строк) показывает PID последнего запущенного процесса, среднюю загрузку системы (которая отражает, насколько система занята), время работы системы (время с последней перезагрузки) и текущее время. Остальные данные в заголовке относятся к количеству запущенных процессов, объёму используемой оперативной памяти и файла подкачки (свопа), а также времени, которое система проводит в различных состояниях процессора. Если загружен модуль файловой системы ZFS, строка `ARC` указывает, сколько данных было прочитано из кэша памяти, а не с диска.

Ниже заголовка расположен ряд столбцов с информацией, аналогичной выводу команды ps(1), такой как PID, имя пользователя, объём CPU времени и команда, запустившая процесс. По умолчанию top(1) также отображает объём памяти, занимаемый процессом. Эта информация разделена на два столбца: один для общего размера и один для резидентного размера. Общий размер — это объём памяти, который потребовался приложению, а резидентный размер — это объём, который оно фактически использует в данный момент.

top(1) автоматически обновляет отображение каждые две секунды. Другой интервал можно указать с помощью `-s`.

3.8.2. Прекращение процессов

Один из способов взаимодействия с любым запущенным процессом или демоном —

отправить *сигнал* с помощью kill(1). Существует множество различных сигналов; некоторые имеют определённое значение, в то время как другие описаны в документации приложения. Пользователь может отправлять сигналы только своим процессам, и попытка отправить сигнал чужому процессу приведёт к ошибке отказа в доступе. Исключением является пользователь `root`, который может отправлять сигналы любым процессам.

Операционная система также может отправлять сигналы процессу. Если приложение написано с ошибками и пытается получить доступ к памяти, к которой оно не должно обращаться, FreeBSD отправит процессу сигнал "Нарушение сегментации" (`SIGSEGV`). Если приложение было написано с использованием системного вызова alarm(3) для оповещения по истечении определённого времени, ему будет отправлен сигнал "Будильник" (`SIGALRM`).

Для остановки процесса могут использоваться два сигнала: `SIGTERM` и `SIGKILL`. `SIGTERM` — это вежливый способ завершить процесс, так как процесс может прочитать сигнал, закрыть все открытые файлы журналов и попытаться завершить текущие операции перед остановкой. В некоторых случаях процесс может игнорировать `SIGTERM`, если он находится в середине задачи, которую нельзя прервать.

`SIGKILL` не может быть проигнорирован процессом. Отправка `SIGKILL` процессу обычно немедленно останавливает этот процесс. [1].

Другие часто используемые сигналы — это `SIGHUP`, `SIGUSR1` и `SIGUSR2`. Поскольку они предназначены для общего применения, разные программы могут реагировать на них по-разному.

Например, после изменения конфигурационного файла веб-сервера необходимо указать веб-серверу перечитать его конфигурацию. Перезапуск `httpd` приведёт к кратковременному простою веб-сервера. Вместо этого отправьте демону сигнал `SIGHUP`. Учтите, что разные демоны могут вести себя по-разному, поэтому обратитесь к документации демона, чтобы определить, приведёт ли `SIGHUP` к желаемому результату.

Убить случайный процесс в системе — плохая идея. В частности, init(8), PID 1, является особым. Запуск `/bin/kill -s KILL 1` — это быстрый, но не рекомендуемый способ выключить систему. *Всегда* перепроверяйте аргументы для kill(1) *перед* нажатием `Return`.

3.9. Оболочки

Оболочка предоставляет интерфейс командной строки для взаимодействия с операционной системой. Оболочка получает команды из входного канала и выполняет их. Многие оболочки предоставляют встроенные функции для помощи в повседневных задачах, таких как управление файлами, подстановка имён файлов (file globbing), редактирование командной строки, макросы команд и переменные окружения. FreeBSD поставляется с несколькими оболочками, включая расширенную оболочку POSIX® shell (sh(1)) и расширенную оболочку C shell (tcsh(1)). Другие оболочки доступны в коллекции портов FreeBSD, например `zsh` и `bash`.

Используемая оболочка — это действительно вопрос предпочтений. Программисту на C

может быть удобнее работать с C-подобной оболочкой, такой как tcsh(1). Пользователь Linux® может предпочесть `bash`. Каждая оболочка обладает уникальными свойствами, которые могут подходить или не подходить под предпочитаемую пользователем рабочую среду, поэтому существует выбор, какую оболочку использовать.

Одной из распространённых возможностей оболочки является завершение имён файлов. Когда пользователь вводит первые несколько букв команды или имени файла и нажимает `Tab`, оболочка автоматически дописывает оставшуюся часть команды или имени файла. Рассмотрим два файла с именами `foobar` и `football`. Чтобы удалить `foobar`, пользователь может ввести `rm foo` и нажать `Tab` для завершения имени файла.

Но оболочка показывает только `rm foo`. Она не смогла завершить имя файла, потому что и `foobar`, и `football` начинаются с `foo`. Некоторые оболочки издают звуковой сигнал или показывают все варианты, если совпадает более одного имени. Пользователь должен затем ввести дополнительные символы, чтобы указать нужное имя файла. Ввод `t` и повторное нажатие `Tab` достаточно, чтобы оболочка определила, какое имя файла требуется, и заполнила остальное.

Еще одна особенность оболочки — использование переменных окружения. Переменные окружения представляют собой пару переменная/ключ, хранящуюся в окружении оболочки. Это окружение может быть прочитано любой программой, запущенной оболочкой, и поэтому содержит множество настроек программ. В Распространенные переменные окружения приведён список распространённых переменных окружения и их значений. Обратите внимание, что имена переменных окружения всегда записываются в верхнем регистре.

Таблица 6. Распространенные переменные окружения

Переменная	Описание
`USER`	Имя текущего вошедшего пользователя.
`PATH`	Двоеточием разделённый список каталогов для поиска бинарных файлов.
`DISPLAY`	Сетевое имя дисплея Xorg для подключения, если доступно.
`SHELL`	Текущая оболочка.
`TERM`	Имя типа терминала пользователя. Используется для определения возможностей терминала.
`TERMCAP`	Запись в базе данных escape-кодов терминала для выполнения различных функций терминала.
`OSTYPE`	Тип операционной системы.
`MACHTYPE`	Архитектура процессора системы.
`EDITOR`	Предпочитаемый текстовый редактор пользователя.
`PAGER`	Предпочитаемая пользователем утилита для просмотра текста постранично.
`MANPATH`	Двоеточием разделённый список каталогов для поиска страниц руководства.

Как установить переменную окружения, зависит от используемой оболочки. В tcsh(1) и csh(1) используйте setenv для установки переменных окружения. В sh(1) и bash используйте export для установки текущих переменных окружения. В этом примере устанавливается значение EDITOR по умолчанию в /usr/local/bin/emacs для оболочки tcsh(1):

```
% setenv EDITOR /usr/local/bin/emacs
```

Эквивалентная команда для bash будет:

```
% export EDITOR="/usr/local/bin/emacs"
```

Чтобы раскрыть переменную окружения и увидеть её текущее значение, введите символ $ перед её именем в командной строке. Например, echo $TERM выведет текущее значение переменной $TERM.

Оболочки интерпретируют специальные символы, известные как метасимволы, как особые представления данных. Наиболее распространённый метасимвол — *, который обозначает любое количество символов в имени файла. Метасимволы могут использоваться для подстановки имён файлов (globbing). Например, команда echo * эквивалентна ls, поскольку оболочка выбирает все файлы, соответствующие шаблону *, а echo выводит их в командной строке.

Чтобы предотвратить интерпретацию специального символа оболочкой, экранируйте его с помощью обратной косой черты (\). Например, echo $TERM выводит настройки терминала, тогда как echo \$TERM выводит строку $TERM буквально.

3.9.1. Изменение оболочки

Самый простой способ навсегда изменить оболочку по умолчанию — использовать chsh. Запуск этой команды откроет редактор, настроенный в переменной окружения EDITOR, которая по умолчанию установлена в vi(1). Измените строку Shell: на полный путь к новой оболочке.

Или используйте chsh -s, который установит указанную оболочку без открытия редактора. Например, чтобы изменить оболочку на bash:

```
% chsh -s /usr/local/bin/bash
```

Введите пароль в командной строке и нажмите Return, чтобы изменить оболочку. Выйдите из системы и войдите снова, чтобы начать использовать новую оболочку.

Новая оболочка **обязательно** должна присутствовать в /etc/shells. Если оболочка была установлена из Коллекции портов FreeBSD, как описано в Установка приложений: Пакеты и порты, она должна быть автоматически добавлена в этот файл. Если её там нет, добавьте её с помощью следующей команды, заменив путь на путь к вашей оболочке:

```
# echo /usr/local/bin/bash >> /etc/shells
```

Затем снова запустите chsh(1).

3.9.2. Продвинутые методы работы с оболочкой

UNIX® оболочка — это не просто командный интерпретатор, а мощный инструмент, который позволяет пользователям выполнять команды, перенаправлять их вывод и ввод, а также объединять команды в цепочки для улучшения конечного результата. В сочетании со встроенными командами это предоставляет пользователю среду, способную максимизировать эффективность работы.

Перенаправление в оболочке — это действие отправки вывода или ввода команды в другую команду или файл. Например, чтобы сохранить вывод команды ls(1) в файл, перенаправьте вывод:

```
% ls > directory_listing.txt
```

Содержимое каталога теперь будет отображено в файле `directory_listing.txt`. Некоторые команды, например sort(1), могут использоваться для чтения ввода. Чтобы отсортировать этот список, перенаправьте ввод:

```
% sort < directory_listing.txt
```

Входные данные будут отсортированы и выведены на экран. Чтобы перенаправить эти данные в другой файл, можно перенаправить вывод `sort(1)`, изменив направление:

```
% sort < directory_listing.txt > sorted.txt
```

Во всех предыдущих примерах команды выполняют перенаправление с использованием файловых дескрипторов. Каждая UNIX®-система имеет файловые дескрипторы, включая стандартный ввод (stdin), стандартный вывод (stdout) и стандартный вывод ошибок (stderr). Каждый из них имеет своё назначение: ввод может осуществляться с клавиатуры или мыши — устройств, предоставляющих входные данные. Вывод может направляться на экран или бумагу в принтере. А ошибки — это всё, что используется для диагностических или сообщений об ошибках. Все три типа считаются файловыми дескрипторами, основанными на вводе-выводе, и иногда называются потоками.

С помощью этих дескрипторов оболочка позволяет передавать вывод и ввод между различными командами и перенаправлять их в файл или из файла. Другой метод перенаправления — оператор конвейера.

Оператор UNIX® pipe, `|`, позволяет передавать вывод одной команды напрямую или направлять его другой программе. По сути, pipe позволяет передать стандартный вывод одной команды в качестве стандартного ввода другой команды, например:

```
% cat directory_listing.txt | sort | less
```

В этом примере содержимое файла `directory_listing.txt` будет отсортировано, а вывод передан в less(1). Это позволяет пользователю прокручивать вывод в удобном темпе и предотвращает его исчезновение с экрана.

3.10. Текстовые редакторы

Большая часть настройки FreeBSD выполняется путём редактирования текстовых файлов, поэтому рекомендуется освоить текстовый редактор. В базовую систему FreeBSD входит несколько таких редакторов, а ещё больше доступно в Коллекции портов.

Простой редактор для освоения — это ee(1), что означает «легкий редактор». Чтобы запустить его, введите `ee имя_файла`, где *имя_файла* — это имя редактируемого файла. Внутри редактора все команды для управления его функциями перечислены в верхней части экрана. Символ каретки (`^`) обозначает `Ctrl`, поэтому `^e` означает `Ctrl` + `e`. Для выхода из ee(1) нажмите `Esc` и выберите пункт «покинуть редактор» в главном меню. Если файл был изменен, редактор предложит сохранить изменения.

FreeBSD также включает более мощные текстовые редакторы, такие как vi(1), в составе базовой системы. Другие редакторы, например editors/emacs и editors/vim, доступны в коллекции портов FreeBSD. Эти редакторы предоставляют больше возможностей, но их освоение сложнее. Изучение более мощного редактора, такого как vim или Emacs, может сэкономить время в долгосрочной перспективе.

Многие приложения, которые изменяют файлы или требуют ввода текста, автоматически открывают текстовый редактор. Чтобы изменить редактор по умолчанию, установите переменную окружения `EDITOR`, как описано в Оболочки.

3.11. Устройства и узлы устройств

Устройство — это термин, в основном используемый для описания аппаратных компонентов системы, таких как диски, принтеры, видеокарты и клавиатуры. При загрузке FreeBSD большинство сообщений относятся к обнаружению устройств. Копия этих сообщений сохраняется в файле `/var/run/dmesg.boot`.

У каждого устройства есть имя и номер. Например, `ada0` — это первый жёсткий диск SATA, а `kbd0` обозначает клавиатуру.

Большинство устройств в FreeBSD должны быть доступны через специальные файлы, называемые узлами устройств, которые расположены в `/dev`.

3.12. Справочник

Наиболее полная документация по FreeBSD представлена в виде Справочника (manual pages). Почти каждая программа в системе имеет краткое справочное руководство, объясняющее основы работы и доступные аргументы. Эти руководства можно

просматривать с помощью команды `man`:

```
% man command
```

где *command* — это название команды, о которой нужно узнать больше. Например, чтобы узнать больше о ls(1), введите:

```
% man ls
```

Страницы Справочника разделены на разделы, которые обозначают тип темы. В FreeBSD доступны следующие разделы:

1. Пользовательские команды.
2. Системные вызовы и коды ошибок.
3. Функции в библиотеках C.
4. Драйверы устройств.
5. Форматы файлов.
6. Развлечения и другие игры.
7. Различная информация.
8. Системные команды обслуживания и эксплуатации.
9. Интерфейсы ядра системы.

В некоторых случаях одна и та же тема может встречаться в нескольких разделах онлайн-руководства. Например, существует пользовательская команда `chmod` и системный вызов `chmod()`. Чтобы указать man(1), какой раздел отображать, укажите номер раздела:

```
% man 1 chmod
```

Это отобразит справочную страницу для пользовательской команды chmod(1). Ссылки на определённый раздел онлайн-руководства традиционно заключаются в скобки в письменной документации, поэтому chmod(1) относится к пользовательской команде, а chmod(2) — к системному вызову.

Если название страницы руководства неизвестно, используйте `man -k` для поиска ключевых слов в описаниях страниц руководства:

```
% man -k mail
```

Эта команда выводит список команд, содержащих ключевое слово "mail" в их описаниях. Это эквивалентно использованию apropos(1).

Чтобы прочитать описания всех команд в `/usr/sbin`, введите:

```
% cd /usr/sbin
% man -f * | more
```

или

```
% cd /usr/sbin
% whatis * |more
```

3.12.1. Файлы GNU Info

FreeBSD включает несколько приложений и утилит, созданных Free Software Foundation (FSF). Помимо man-страниц, эти программы могут содержать гипертекстовые документы, называемые файлами `info`. Их можно просматривать с помощью info(1) или, если установлен editors/emacs, в режиме info редактора emacs.

Чтобы использовать info(1), введите:

```
% info
```

Для краткого введения введите `h`. Для быстрой справки по командам введите `?`.

[1] Существует несколько задач, которые не могут быть прерваны. Например, если процесс пытается прочитать файл, находящийся на другом компьютере в сети, и этот компьютер недоступен, процесс считается непрерываемым. В конечном итоге процесс завершится по таймауту, обычно через две минуты. Как только это произойдет, процесс будет убит.

Глава 4. Установка приложений: пакеты и порты

4.1. Обзор

FreeBSD поставляется с богатым набором системных инструментов в составе базовой системы. Кроме того, FreeBSD предоставляет две дополнительные технологии для установки стороннего программного обеспечения: коллекцию портов FreeBSD для установки из исходных кодов и пакеты для установки из предварительно собранных бинарных файлов. Оба метода могут быть использованы для установки программного обеспечения как с локальных носителей, так и из сети.

Прочтите эту главу, чтобы узнать:

- Разница между бинарными пакетами и портами.
- Как найти стороннее программное обеспечение, портированное на FreeBSD.
- Как управлять бинарными пакетами с помощью pkg.
- Как собрать стороннее программное обеспечение из исходных кодов с использованием Коллекции портов.
- Как найти файлы, установленные с приложением, для пост-установочной настройки.
- Что делать, если установка программного обеспечения не удалась.

4.2. Обзор установки программного обеспечения

Порт FreeBSD — это набор файлов, предназначенных для автоматизации процесса компиляции приложения из исходного кода. Файлы, из которых состоит порт, содержат всю необходимую информацию для автоматической загрузки, распаковки, наложения исправлений, компиляции и установки приложения.

Если программное обеспечение ещё не адаптировано и не протестировано на FreeBSD, может потребоваться редактирование исходного кода для его правильной установки и работы.

Однако более 37000 сторонних приложений уже портированы на FreeBSD. По возможности эти приложения доступны для загрузки в виде предварительно скомпилированных *пакетов*.

Пакетами можно управлять с помощью команд управления пакетами FreeBSD.

Как пакеты, так и порты учитывают зависимости. Если приложение устанавливается с помощью пакета или порта, а необходимая библиотека ещё не установлена, эта библиотека будет автоматически установлена первой.

Пакет FreeBSD содержит предварительно скомпилированные версии всех команд приложения, а также любые конфигурационные файлы и документацию. Пакетом можно

управлять с помощью команд pkg(8), таких как `pkg install`.

Хотя эти две технологии схожи, пакеты и порты имеют свои сильные стороны. Выберите технологию, которая соответствует вашим потребностям, для установки конкретного приложения.

Преимущества пакетов

- Сжатый tar-архив пакета обычно меньше, чем сжатый tar-архив с исходным кодом приложения.
- Пакеты не требуют времени на компиляцию. Для больших приложений, таких как Firefox, KDE Plasma или GNOME, это может быть важно на медленной системе.
- Пакеты не требуют понимания процесса компиляции программного обеспечения в FreeBSD.

Преимущества портов

- Пакеты обычно компилируются с консервативными параметрами, так как они должны работать на максимальном количестве систем. При компиляции из портов можно изменить параметры компиляции.
- Некоторые приложения имеют параметры на этапе компиляции, определяющие, какие функции будут установлены. Например, NGINX® можно настроить с широким набором различных встроенных опций.

  В некоторых случаях для одного приложения могут существовать несколько пакетов с разными настройками. Например, NGINX® доступен в виде пакетов `nginx` и `nginx-lite`. Первый имеет гораздо больше включённых опций, но это, в свою очередь, требует установки множества зависимостей для его работы, что увеличивает занимаемое место и поверхность для атак.

  Транзитивные зависимости могут стать довольно большими, например, полный пакет `nginx` потянет за собой несколько X-библиотек, что может оказаться неожиданным. Сборка из портов позволяет выбрать только нужные опции, избегая подхода «всё включено». В некоторых случаях для одного приложения могут существовать несколько пакетов с разными настройками.

- Условия лицензирования некоторых программных продуктов запрещают распространение в бинарном виде. Такое программное обеспечение должно распространяться в виде исходного кода, который конечный пользователь должен самостоятельно скомпилировать.
- Некоторые люди не доверяют бинарным дистрибутивам или предпочитают изучать исходный код, чтобы выявить потенциальные проблемы.
- Исходный код необходим для применения пользовательских исправлений.

Для отслеживания обновлений портов подпишитесь на рассылки Список рассылки, посвящённый Портам FreeBSD и Список рассылки, посвящённый ошибкам в портах FreeBSD.

Перед установкой приложения проверьте https://vuxml.freebsd.org/ на

> наличие связанных проблем с безопасностью.
>
> Для проверки установленных пакетов на наличие известных уязвимостей выполните команду `pkg audit -F`.

Оставшаяся часть этой главы объясняет, как использовать пакеты и порты для установки и управления сторонним программным обеспечением в FreeBSD.

4.3. Поиск программного обеспечения

Список доступных приложений для FreeBSD постоянно растет. Существует несколько способов найти программное обеспечение для установки:

- Веб-сайт FreeBSD содержит актуальный поисковый список всех доступных приложений на странице Портала портов. Порты можно искать по названию приложения или по категории программного обеспечения.
- Дэн Лангилл поддерживает FreshPorts, который предоставляет удобный поиск и отслеживает изменения в приложениях из коллекции портов. Зарегистрированные пользователи могут создать индивидуальный список наблюдения, чтобы получать автоматические уведомления по электронной почте при обновлении отслеживаемых портов.
- Если найти конкретное приложение становится сложно, попробуйте поискать на таких сайтах, как SourceForge или GitHub, а затем проверьте ссылку Ports Portal, чтобы узнать, было ли это приложение портировано.
- Поиск репозитория бинарных пакетов для приложения с помощью команды pkg(8)

4.4. Использование pkg для управления бинарными пакетами

pkg(8) предоставляет интерфейс для управления пакетами: регистрации, добавления, удаления и обновления пакетов.

Для сайтов, которые хотят использовать только предварительно собранные бинарные пакеты из зеркал FreeBSD, управления пакетами с помощью pkg(8) может быть достаточно.

Однако для сайтов, собирающих из исходного кода, потребуется отдельный инструмент управления портами.

Поскольку pkg(8) работает только с бинарными пакетами, он не является заменой таким инструментам. Эти инструменты можно использовать для установки программного обеспечения как из бинарных пакетов, так и из коллекции портов, тогда как pkg(8) устанавливает только бинарные пакеты.

4.4.1. Начало работы с pkg

Все поддерживаемые версии FreeBSD теперь содержат `/usr/sbin/pkg`, также известный как pkg(7). Это небольшая заглушка, которая обладает лишь минимальной

функциональностью, необходимой для установки настоящего pkg(8).

Для успешного завершения процесса начальной загрузки требуется рабочее подключение к Интернету.

Выполните команду pkg(8) из командной строки:

```
# pkg
```

Вывод должен быть похож на следующий:

```
The package management tool is not yet installed on your system.
Do you want to fetch and install it now? [y/N]
```

pkg(7) перехватит команду, и если вы подтвердите своё намерение, загрузит tarball pkg(8), установит pkg(8) из него, инициализирует локальную базу данных пакетов, а затем выполнит изначально запрошенную команду.

Более новые версии pkg(7) поддерживают `pkg -N` для проверки, установлен ли pkg(8), без запуска процесса установки, и, наоборот, `pkg bootstrap[-f]` для установки pkg(8) (или принудительной переустановки) без выполнения других действий.

Информация по использованию pkg доступна на pkg(8) или при запуске `pkg` без дополнительных аргументов. Дополнительные параметры настройки pkg описаны в pkg.conf(5).

Каждый аргумент команды pkg документирован в соответствующем руководстве, специфичном для команды.

Чтобы прочитать справочную страницу для `pkg install`, например, выполните следующую команду:

```
# pkg help install
```

Оставшаяся часть этого раздела демонстрирует распространённые задачи управления бинарными пакетами, которые можно выполнять с помощью pkg(8). Каждая из представленных команд предоставляет множество ключей для настройки их использования. Подробности и дополнительные примеры смотрите в справке (help) или man-странице соответствующей команды.

4.4.2. Квартальные и последние ветви портов

Ветка `Quarterly` предоставляет пользователям более предсказуемую и стабильную среду для установки и обновления портов и пакетов. Это достигается за счёт того, что в неё вносятся только обновления, не связанные с добавлением новых функций. Ветки Quarterly получают исправления безопасности (которые могут включать обновления версий или обратные

порты коммитов), исправления ошибок, а также изменения, связанные с соответствием портов или изменения в их инфраструктуре. Ветка Quarterly создаётся из HEAD в начале каждого квартала (года) в январе, апреле, июле и октябре. Ветки именуются в соответствии с годом (YYYY) и кварталом (Q1-4), в котором они созданы. Например, ветка quarterly, созданная в январе 2023 года, называется 2023Q1. А ветка `Latest` предоставляет пользователям самые последние версии пакетов.

Чтобы переключить pkg(8) с Quarterly на Latest, выполните следующие команды:

```
# mkdir -p /usr/local/etc/pkg/repos
# echo 'FreeBSD-ports: { url: "pkg+https://pkg.FreeBSD.org/${ABI}/latest" }' >
/usr/local/etc/pkg/repos/FreeBSD.conf
```

Затем выполните эту команду, чтобы обновить каталоги локальных репозиториев пакетов для ветки Latest:

```
# pkg update -f
```

4.4.3. Репозитории модулей ядра

Репозитории модулей ядра позволяют пользователям устанавливать готовые к использованию модули, такие как драйверы графики и поддержка специфичного оборудования. Начиная с FreeBSD 14.3, проект FreeBSD предоставляет собранные модули ядра для каждой поддерживаемой версии. Чтобы создать конфигурацию такого репозитория (если она отсутствует), добавьте следующее в /usr/local/etc/pkg/repos/kmods.conf:

```
FreeBSD-kmods: {
        url: "pkg+https://pkg.FreeBSD.org/${ABI}/KMODSFLAVOR",
        mirror_type: "srv",
        signature_type: "fingerprints",
        fingerprints: "/usr/share/keys/pkg",
        enabled: yes
}
```

Переменная `KMODSFLAVOR` использует следующий шаблон именования: `kmods_PORTBRANCH_MINORRELEASE`.

Например:

Таблица 7. Kmodsflavor

Релиз FreeBSD	**главные порты**	**квартальные порты**
FreeBSD 14.2-RELEASE	kmods_latest_2	kmods_quarterly_2
FreeBSD 14.3-RELEASE	kmods_latest_3	kmods_quarterly_3
FreeBSD 14.3-STABLE	kmods_latest	kmods_quarterly

Релиз FreeBSD	главные порты	квартальные порты
FreeBSD 15.0-CURRENT	kmods_latest	

4.4.4. Настройка pkg

pkg.conf(5) — это общесистемный конфигурационный файл, используемый утилитами pkg(8). Стандартное расположение этого файла — `/usr/local/etc/pkg.conf`.

FreeBSD не требует наличия файла `pkg.conf`. Многие установки будут работать без `pkg.conf` вообще или с пустым `pkg.conf` (за исключением строк с комментариями).

Строки в файле, начинающиеся с символа "#", являются комментариями и игнорируются.

Файл имеет формат UCL. Для получения дополнительной информации о синтаксисе libucl(3) посетите официальный сайт UCL.

Распознаются следующие типы опций - логические, строковые и списковые.

Логическая опция считается включённой, если в файле конфигурации указано одно из следующих значений: YES, TRUE или ON.

4.4.5. Поиск пакетов

Для поиска пакета можно использовать pkg-search(8):

```
# pkg search nginx
```

Вывод должен быть похож на следующий:

```
modsecurity3-nginx-1.0.3       Instruction detection and prevention engine / nginx
Wrapper
nginx-1.22.1_2,3               Robust and small WWW server
nginx-devel-1.23.2_4           Robust and small WWW server
nginx-full-1.22.1_1,3          Robust and small WWW server (full package)
nginx-lite-1.22.1,3            Robust and small WWW server (lite package)
nginx-naxsi-1.22.1,3           Robust and small WWW server (plus NAXSI)
nginx-prometheus-exporter-0.10.0_7 Prometheus exporter for NGINX and NGINX Plus stats
nginx-ultimate-bad-bot-blocker-4.2020.03.2005_1 Nginx bad bot and other things blocker
nginx-vts-exporter-0.10.7_7    Server that scraps NGINX vts stats and export them via
HTTP
p5-Nginx-ReadBody-0.07_1       Nginx embeded perl module to read and evaluate a
request body
p5-Nginx-Simple-0.07_1         Perl 5 module for easy to use interface for Nginx Perl
Module
p5-Test-Nginx-0.30             Testing modules for Nginx C module development
py39-certbot-nginx-2.0.0       NGINX plugin for Certbot
```

```
rubygem-passenger-nginx-6.0.15 Modules for running Ruby on Rails and Rack applications
```

4.4.6. Установка и загрузка пакетов

Для установки бинарного пакета можно использовать pkg-install(8). Эта команда использует данные репозитория для определения версии программного обеспечения, которую нужно установить, а также наличия неустановленных зависимостей. Например, для установки curl:

```
# pkg install curl
```

Вывод должен быть похож на следующий:

```
Updating FreeBSD repository catalogue...
FreeBSD repository is up to date.
All repositories are up to date.
The following 9 package(s) will be affected (of 0 checked):

New packages to be INSTALLED:
        ca_root_nss: 3.83
        curl: 7.86.0
        gettext-runtime: 0.21
        indexinfo: 0.3.1
        libidn2: 2.3.3
        libnghttp2: 1.48.0
        libpsl: 0.21.1_4
        libssh2: 1.10.0.3
        libunistring: 1.0

Number of packages to be installed: 9

The process will require 11 MiB more space.
3 MiB to be downloaded

Proceed with this action? [y/N]
```

Новый пакет и любые дополнительные пакеты, установленные в качестве зависимостей, можно увидеть в списке установленных пакетов:

```
# pkg info
```

Вывод должен быть похож на следующий:

```
ca_root_nss-3.83              Root certificate bundle from the Mozilla Project
curl-7.86.0                   Command line tool and library for transferring data
with URLs
```

```
gettext-runtime-0.21.1          GNU gettext runtime libraries and programs
indexinfo-0.3.1                 Utility to regenerate the GNU info page index
libidn2-2.3.3                   Implementation of IDNA2008 internationalized domain
names
libnghttp2-1.48.0               HTTP/2.0 C Library
libpsl-0.21.1_6                 C library to handle the Public Suffix List
libssh2-1.10.0.3                Library implementing the SSH2 protocol
libunistring-1.0                Unicode string library
pkg-1.18.4                      Package manager
```

Для загрузки пакета с последующей установкой позже или в другом месте используйте pkg-fetch(8). Например, чтобы скачать `nginx-lite`:

```
# pkg fetch -d -o /usr/home/user/packages/ nginx-lite
```

- `-d`: используется для получения всех зависимостей
- `-o`: используется для указания каталога загрузки

Вывод должен быть похож на следующий:

```
Updating FreeBSD repository catalogue...
FreeBSD repository is up to date.
All repositories are up to date.
The following packages will be fetched:

New packages to be FETCHED:
        nginx-lite: 1.22.1,3 (342 KiB: 22.20% of the 2 MiB to download)
        pcre: 8.45_3 (1 MiB: 77.80% of the 2 MiB to download)

Number of packages to be fetched: 2

The process will require 2 MiB more space.
2 MiB to be downloaded.

Proceed with fetching packages? [y/N]:
```

Для установки загруженных пакетов можно использовать pkg-install(8) следующим образом:

```
# cd /usr/home/user/packages/
```

```
# pkg install nginx-lite-1.22.1,3.pkg
```

4.4.7. Получение информации об установленных пакетах

Информацию об установленных в системе пакетах можно просмотреть с помощью

команды pkg-info(8), которая при запуске без каких-либо параметров выводит версию пакета для всех установленных пакетов или указанного пакета.

Например, чтобы узнать, какая версия pkg установлена, выполните:

```
# pkg info pkg
```

Вывод должен быть похож на следующий:

```
pkg-1.19.0
Name           : pkg
Version        : 1.19.0
Installed on   : Sat Dec 17 11:05:28 2022 CET
Origin         : ports-mgmt/pkg
Architecture   : FreeBSD:13:amd64
Prefix         : /usr/local
Categories     : ports-mgmt
Licenses       : BSD2CLAUSE
Maintainer     : pkg@FreeBSD.org
WWW            : https://github.com/freebsd/pkg
Comment        : Package manager
Options        :
        DOCS           : on
Shared Libs provided:
        libpkg.so.4
Annotations    :
        FreeBSD_version: 1301000
        repo_type      : binary
        repository     : FreeBSD
Flat size      : 33.2MiB
Description    :
Package management tool

WWW: https://github.com/freebsd/pkg
```

4.4.8. Обновление установленных пакетов

Установленные пакеты можно обновить до их последних версий с помощью pkg-upgrade(8):

```
# pkg upgrade
```

Эта команда сравнит установленные версии с доступными в каталоге репозитория и обновит их из репозитория.

4.4.9. Проверка установленных пакетов

Уязвимости в программном обеспечении регулярно обнаруживаются в сторонних

приложениях. Для решения этой проблемы pkg включает встроенный механизм аудита. Чтобы определить, есть ли известные уязвимости для программного обеспечения, установленного в системе, используйте pkg-audit(8):

```
# pkg audit -F
```

Вывод должен быть похож на следующий:

```
Fetching vuln.xml.xz: 100%  976 KiB 499.5kB/s    00:02
chromium-108.0.5359.98 is vulnerable:
  chromium -- multiple vulnerabilities
  CVE: CVE-2022-4440
  CVE: CVE-2022-4439
  CVE: CVE-2022-4438
  CVE: CVE-2022-4437
  CVE: CVE-2022-4436
  WWW: https://vuxml.FreeBSD.org/freebsd/83eb9374-7b97-11ed-be8f-3065ec8fd3ec.html
```

4.4.10. Удаление пакетов

Пакеты, которые больше не нужны, можно удалить с помощью pkg-delete(8).

Для примера:

```
# pkg delete curl
```

Вывод должен быть похож на следующий:

```
Checking integrity... done (0 conflicting)
Deinstallation has been requested for the following 1 packages (of 0 packages in the
universe):

Installed packages to be REMOVED:
        curl :7.86.0

Number of packages to be removed: 1

The operation will free 4 MiB.

Proceed with deinstallation packages? [y/N]: y
[1/1] Deinstalling curl-7.86.0...
[1/1] Deleting files for curl-7.86.0: 100%
```

4.4.11. Автоматическое удаление неиспользуемых пакетов

Удаление пакета может оставить зависимости, которые больше не требуются. Ненужные пакеты, установленные как зависимости (листовые пакеты), могут быть автоматически обнаружены и удалены с помощью pkg-autoremove(8):

```
# pkg autoremove
```

Вывод должен быть похож на следующий:

```
Checking integrity... done (0 conflicting)
Deinstallation has been requested for the following 1 packages:

Installed packages to be REMOVED:
        ca_root_nss-3.83

Number of packages to be removed: 1

The operation will free 723 KiB.

Proceed with deinstalling packages? [y/N]:
```

Установленные в качестве зависимостей пакеты называются *автоматическими* пакетами. Неавтоматические пакеты, то есть пакеты, которые были явно установлены не в качестве зависимости для другого пакета, можно вывести с помощью:

```
# pkg prime-list
```

Вывод должен быть похож на следующий:

```
nginx
openvpn
sudo
```

`pkg prime-list` — это псевдоним команды, объявленный в `/usr/local/etc/pkg.conf`. Существует множество других команд, которые можно использовать для запросов к базе данных пакетов системы. Например, команда `pkg prime-origins` позволяет получить каталог портов происхождения для упомянутого выше списка:

```
# pkg prime-origins
```

Вывод должен быть похож на следующий:

```
www/nginx
```

```
security/openvpn
security/sudo
```

Этот список можно использовать для пересборки всех пакетов, установленных в системе, с помощью инструментов сборки, таких как ports-mgmt/poudriere или ports-mgmt/synth.

Пометить установленный пакет как автоматический можно с помощью:

```
# pkg set -A 1 devel/cmake
```

Как только пакет становится листовым и помечается как автоматический, он выбирается командой `pkg autoremove`.

Пометить установленный пакет как *не* автоматический можно с помощью:

```
# pkg set -A 0 devel/cmake
```

4.4.12. Удаление устаревших пакетов

По умолчанию pkg хранит бинарные пакеты в кэш-каталоге, определённом параметром `PKG_CACHEDIR` в pkg.conf(5). Сохраняются только копии последних установленных пакетов. В более старых версиях pkg сохранялись все предыдущие пакеты. Чтобы удалить устаревшие бинарные пакеты, выполните:

```
# pkg clean
```

Весь кэш может быть очищен выполнением команды:

```
# pkg clean -a
```

4.4.13. Блокировка и разблокировка пакетов

pkg-lock(8) используется для блокировки пакетов от переустановки, изменения или удаления. pkg-unlock(8) разблокирует указанные пакеты. Оба варианта влияют только на уже установленные пакеты. Следовательно, невозможно предотвратить установку нового пакета с помощью этого механизма, за исключением случаев, когда такая установка подразумевает обновление заблокированного пакета.

Например, чтобы заблокировать `nginx-lite`:

```
# pkg lock nginx-lite
```

И чтобы разблокировать `nginx-lite`:

```
# pkg unlock nginx-lite
```

4.4.14. Изменение метаданных пакета

Программное обеспечение в коллекции портов FreeBSD может подвергаться изменениям основных номеров версий. Для решения этой проблемы в pkg есть встроенная команда для обновления происхождения пакетов. Это может быть полезно, например, если lang/python3 переименован в lang/python311, чтобы lang/python3 теперь мог представлять версию `3.11`.

Чтобы изменить источник пакета для приведённого выше примера, выполните:

```
# pkg set -o lang/python3:lang/python311
```

В качестве другого примера, чтобы обновить lang/ruby31 до lang/ruby32, выполните:

```
# pkg set -o lang/ruby31:lang/ruby32
```

При изменении происхождения пакетов важно переустановить пакеты, зависящие от пакета с изменённым происхождением. Для принудительной переустановки зависимых пакетов выполните:

```
# pkg install -Rf lang/ruby32
```

4.5. Использование коллекции портов

Коллекция портов — это набор `Makefile`-файлов, патчей и описаний. Каждый такой набор файлов используется для сборки и установки отдельного приложения в FreeBSD и называется *портом*.

По умолчанию Коллекция портов хранится в подкаталоге `/usr/ports`.

Прежде чем устанавливать и использовать Коллекцию портов, учтите, что обычно не рекомендуется использовать Коллекцию портов вместе с бинарными пакетами, предоставляемыми через pkg, для установки программного обеспечения. По умолчанию pkg отслеживает квартальные ветки-релизы дерева портов, а не HEAD. Зависимости для порта в HEAD могут отличаться от его аналога в квартальном релизе ветки, что может привести к конфликтам между зависимостями, установленными pkg, и теми, что из Коллекции портов. Если необходимо использовать Коллекцию портов и pkg вместе, убедитесь, что ваша Коллекция портов и pkg находятся на одной ветке релиза дерева портов.

Коллекция портов содержит каталоги для категорий программного обеспечения. В каждой

категории находятся подкаталоги для отдельных приложений. Каждый подкаталог приложения содержит набор файлов, которые сообщают FreeBSD, как компилировать и устанавливать эту программу, называемый *каркасом порта* (ports skeleton). Каждый каркас порта включает следующие файлы и каталоги:

- **Makefile**: содержит инструкции, определяющие порядок компиляции приложения и расположение его компонентов при установке.
- **distinfo**: содержит имена и контрольные суммы файлов, которые необходимо загрузить для сборки порта.
- **files/**: в этом каталоге содержатся все необходимые патчи для компиляции и установки программы в FreeBSD. Также в этом каталоге могут находиться другие файлы, используемые для сборки порта.
- **pkg-descr**: содержит более подробное описание программы.
- **pkg-plist**: список всех файлов, которые будут установлены портом. Также он указывает системе портов, какие файлы следует удалить при деинсталляции.

Некоторые порты включают файлы `pkg-message` или другие для обработки особых ситуаций. Для получения более подробной информации об этих файлах и о портах в целом обратитесь к Руководству по созданию портов FreeBSD.

Порт не включает в себя исходный код, также известный как `distfile`. Этап извлечения при сборке порта автоматически сохраняет загруженные исходные файлы в `/usr/ports/distfiles`.

4.5.1. Установка коллекции портов

Прежде чем приложение можно будет скомпилировать с использованием порта, необходимо установить Коллекцию портов. Если она не была установлена во время установки FreeBSD, используйте следующий метод для её установки:

Процедура: Метод Git

Если требуется больше контроля над деревом портов или необходимо поддерживать локальные изменения, либо если используется FreeBSD-CURRENT, для получения коллекции портов можно использовать Git. Подробное описание Git см. в руководстве по Git.

Добавляем `--depth 1` в командную строку `git`, чтобы клонировать дерево без получения истории коммитов, что экономит время и приемлемо для большинства пользователей. Если у вас есть собственные изменения в дереве портов или вам нужна история по какой-либо причине, опустите аргумент `--depth 1` ниже.

1. Git должен быть установлен перед тем, как его можно будет использовать для получения дерева портов. Если копия дерева портов уже существует, установите Git следующим образом:

```
# cd /usr/ports/devel/git
```

```
# make install clean
```

Если дерево портов недоступно или для управления пакетами используется pkg, Git можно установить как пакет:

```
# pkg install git
```

2. Извлеките копию ветки HEAD дерева портов:

```
# git clone --depth 1 https://git.FreeBSD.org/ports.git /usr/ports
```

3. Или склонируйте копию квартальной ветки:

```
# git clone --depth 1 https://git.FreeBSD.org/ports.git -b 2023Q1 /usr/ports
```

4. По мере необходимости обновите `/usr/ports` после первоначального получения из Git:

```
# git -C /usr/ports pull
```

5. По мере необходимости переключите `/usr/ports` на другую квартальную ветку:

```
# git -C /usr/ports fetch --depth 1 origin 2023Q2:2023Q2
# git -C /usr/ports switch 2023Q2
```

4.5.2. Установка портов

Этот раздел содержит основные инструкции по использованию коллекции портов для установки или удаления программного обеспечения. Подробное описание доступных целей `make` и переменных окружения доступно в ports(7).

Прежде чем компилировать любой порт, обязательно обновите коллекцию портов, как описано в предыдущем разделе. Поскольку установка любого стороннего программного обеспечения может привести к уязвимостям в безопасности, рекомендуется сначала проверить https://vuxml.freebsd.org/ на наличие известных проблем безопасности, связанных с портом. Или выполните `pkg audit -F` перед установкой нового порта. Эта команда может быть настроена для автоматической проверки безопасности и обновления базы данных уязвимостей в ходе ежедневной проверки безопасности системы. Для получения дополнительной информации обратитесь к pkg-audit(8) и periodic(8).

Использование коллекции портов предполагает наличие работающего подключения к Интернету. Также требуются права суперпользователя.

Для компиляции и установки порта перейдите в каталог порта, который нужно установить, затем введите `make install` в командной строке. Сообщения будут отображать прогресс:

```
# cd /usr/ports/sysutils/lsof
# make install
>> lsof_4.88D.freebsd.tar.gz doesn't seem to exist in /usr/ports/distfiles/.
>> Attempting to fetch from ftp://lsof.itap.purdue.edu/pub/tools/unix/lsof/.
===>  Extracting for lsof-4.88
...
[extraction output snipped]
...
>> Checksum OK for lsof_4.88D.freebsd.tar.gz.
===>  Patching for lsof-4.88.d,8
===>  Applying FreeBSD patches for lsof-4.88.d,8
===>  Configuring for lsof-4.88.d,8
...
[configure output snipped]
...
===>  Building for lsof-4.88.d,8
...
[compilation output snipped]
...

===>  Installing for lsof-4.88.d,8
...
[installation output snipped]
...
===>   Generating temporary packing list
===>   Compressing manual pages for lsof-4.88.d,8
===>   Registering installation for lsof-4.88.d,8
===>  SECURITY NOTE:
      This port has installed the following binaries which execute with
      increased privileges.
/usr/local/sbin/lsof
#
```

Поскольку `lsof` — это программа, работающая с повышенными привилегиями, при её установке отображается предупреждение системы безопасности. После завершения установки будет возвращена командная строка.

Некоторые оболочки сохраняют кэш команд, доступных в каталогах, перечисленных в переменной окружения `PATH`, чтобы ускорить поиск исполняемых файлов этих команд. Пользователи оболочки `tcsh` должны ввести `rehash`, чтобы новоустановленная команда могла использоваться без указания полного пути. Для оболочки `sh` используйте `hash -r`. Дополнительную информацию можно найти в документации по используемой оболочке.

Во время установки создаётся рабочий подкаталог, содержащий все временные файлы, используемые при компиляции. Удаление этого каталога позволяет сэкономить место на диске и снижает вероятность возникновения проблем в дальнейшем при обновлении до более новой версии порта:

```
# make clean
===>  Cleaning for lsof-88.d,8
#
```

Чтобы избежать этого дополнительного шага, используйте `make install clean` при компиляции порта.

Настройка установки портов

Некоторые порты предоставляют опции сборки, которые можно использовать для включения или отключения компонентов приложения, обеспечения параметров безопасности или других настроек. Примеры включают www/firefox и security/gpgme. Если порт зависит от других портов с настраиваемыми опциями, процесс может несколько раз приостанавливаться для взаимодействия с пользователем, так как по умолчанию предлагается выбрать опции из меню. Чтобы избежать этого и выполнить всю настройку одной командой, выполните `make config-recursive` в каталоге порта. Затем выполните `make install [clean]` для компиляции и установки порта.

При использовании `config-recursive` список портов для настройки собирается с помощью цели `all-depends-list`. Рекомендуется выполнять `make config-recursive` до тех пор, пока не будут определены все параметры зависимых портов и экраны выбора опций портов больше не появляются, чтобы убедиться, что все параметры зависимостей настроены.

Существует несколько способов вернуться к меню настроек сборки порта, чтобы добавить, удалить или изменить параметры после того, как порт уже был собран. Один из методов — перейти в каталог с портом с помощью `cd` и ввести `make config`. Другой вариант — использовать `make showconfig`. Также можно выполнить `make rmconfig`, что удалит все выбранные параметры и позволит начать заново. Все эти и другие варианты подробно описаны в ports(7).

Система портов использует fetch(1) для загрузки исходных файлов, которая поддерживает различные переменные окружения. Переменные `FTP_PASSIVE_MODE`, `FTP_PROXY` и `FTP_PASSWORD` может потребоваться установить, если система FreeBSD находится за межсетевым экраном или FTP/HTTP прокси. Полный список поддерживаемых переменных смотрите в fetch(3).

Для пользователей, которые не могут быть постоянно подключены к интернету, команда `make fetch` может быть выполнена в `/usr/ports`, чтобы загрузить все distfiles, или в категории, например `/usr/ports/net`, или в конкретном скелете порта. Обратите внимание, что если порт имеет зависимости, выполнение этой команды в категории или скелете порта *не* загрузит distfiles портов из другой категории. Вместо этого используйте `make fetch-recursive`, чтобы также загрузить distfiles для всех зависимостей порта.

В редких случаях, например, когда у организации есть локальный репозиторий distfiles, переменная `MASTER_SITES` может быть использована для переопределения мест загрузки, указанных в `Makefile`. При использовании укажите альтернативное расположение:

```
# cd /usr/ports/directory
# make MASTER_SITE_OVERRIDE= \
ftp://ftp.organization.org/pub/FreeBSD/ports/distfiles/ fetch
```

Переменные `WRKDIRPREFIX` и `PREFIX` позволяют переопределить рабочий и целевой каталоги по умолчанию. Например:

```
# make WRKDIRPREFIX=/usr/home/example/ports install
```

соберет порт в `/usr/home/example/ports` и установит все в `/usr/local`.

```
# make PREFIX=/usr/home/example/local install
```

скомпилирует порт в `/usr/ports` и установит его в `/usr/home/example/local`. И:

```
# make WRKDIRPREFIX=../ports PREFIX=../local install
```

объединит эти два.

Эти параметры также могут быть установлены как переменные среды. Обратитесь к руководству вашей оболочки для получения инструкций по установке переменных среды.

4.5.3. Удаление установленных портов

Установленные порты можно удалить с помощью `pkg delete`. Примеры использования этой команды приведены на pkg-delete(8).

Или в каталоге порта можно выполнить `make deinstall`:

```
# cd /usr/ports/sysutils/lsof
# make deinstall
===>  Deinstalling for sysutils/lsof
===>   Deinstalling
Deinstallation has been requested for the following 1 packages:

    lsof-4.88.d,8

The deinstallation will free 229 kB
[1/1] Deleting lsof-4.88.d,8... done
```

Рекомендуется прочитать сообщения во время удаления порта. Если у порта есть

приложения, которые от него зависят, эта информация будет отображена, но удаление продолжится. В таких случаях может быть лучше переустановить приложение, чтобы избежать нарушенных зависимостей.

4.5.4. Обновление портов

Со временем в Коллекции портов становятся доступны новые версии программного обеспечения. В этом разделе описано, как определить, какие программы можно обновить, и как выполнить обновление.

Чтобы определить, доступны ли более новые версии установленных портов, убедитесь, что у вас установлена последняя версия дерева портов, используя команду обновления, описанную в Git Method. Следующая команда выведет список устаревших установленных портов:

```
# pkg version -l "<"
```

Перед попыткой обновления прочитайте файл `/usr/ports/UPDATING` с начала до даты, наиболее близкой к последнему обновлению портов или установке системы. Этот файл описывает различные проблемы и дополнительные шаги, с которыми могут столкнуться пользователи при обновлении портов, включая такие вещи, как изменения форматов файлов, перемещение конфигурационных файлов или несовместимость с предыдущими версиями. Отметьте все инструкции, относящиеся к портам, которые требуют обновления, и следуйте этим инструкциям при выполнении обновления.

Инструменты для обновления и управления портами

Коллекция портов содержит несколько утилит для выполнения обновления. У каждой есть свои сильные и слабые стороны.

Исторически большинство установок использовали либо Portmaster, либо Portupgrade. Synth — это более современная альтернатива.

Выбор наилучшего инструмента для конкретной системы остаётся за системным администратором. Рекомендуется создать резервную копию данных перед использованием любого из этих инструментов.

Обновление портов с помощью Portmaster

ports-mgmt/portmaster — это очень небольшая утилита для обновления установленных портов. Она предназначена для использования инструментов, установленных в базовой системе FreeBSD, без зависимости от других портов или баз данных. Чтобы установить эту утилиту как порт:

```
# cd /usr/ports/ports-mgmt/portmaster
```

```
# make install clean
```

Portmaster определяет четыре категории портов:

- Корневой порт (root port): не имеет зависимостей и сам не является зависимостью для других портов.
- Ствольный порт (trunk port): не имеет зависимостей, но другие порты зависят от него.
- Веточный порт (branch port): имеет зависимости и другие порты зависят от него.
- Листовой порт (leaf port): имеет зависимости, но другие порты от него не зависят.

Чтобы перечислить эти категории и найти обновления:

```
# portmaster -L
===>>> Root ports (No dependencies, not depended on)
===>>> ispell-3.2.06_18
===>>> screen-4.0.3
        ===>>> New version available: screen-4.0.3_1
===>>> tcpflow-0.21_1
===>>> 7 root ports
...
===>>> Branch ports (Have dependencies, are depended on)
===>>> apache22-2.2.3
        ===>>> New version available: apache22-2.2.8
...
===>>> Leaf ports (Have dependencies, not depended on)
===>>> automake-1.9.6_2
===>>> bash-3.1.17
        ===>>> New version available: bash-3.2.33
...
===>>> 32 leaf ports

===>>> 137 total installed ports
        ===>>> 83 have new versions available
```

Эта команда используется для обновления всех устаревших портов:

```
# portmaster -a
```

По умолчанию Portmaster создаёт резервную копию пакета перед удалением существующего порта. Если установка новой версии проходит успешно, Portmaster удаляет резервную копию. Использование опции `-b` указывает Portmaster не удалять резервную копию автоматически. Добавление опции `-i` запускает Portmaster в интерактивном режиме, запрашивая подтверждение перед обновлением каждого порта. Доступно множество других опций. Подробности об их использовании можно узнать на man-странице portmaster(8).

Если в процессе обновления возникают ошибки, добавьте `-f` для обновления и пересборки всех портов:

```
# portmaster -af
```

Portmaster также может использоваться для установки новых портов в системе, обновляя все зависимости перед сборкой и установкой нового порта. Чтобы использовать эту функцию, укажите расположение порта в коллекции портов:

```
# portmaster shells/bash
```

Дополнительную информацию о ports-mgmt/portmaster можно найти в его `pkg-descr`.

Обновление портов с помощью Portupgrade

Portupgrade устарел и будет удален в ближайшем будущем.

ports-mgmt/portupgrade — это ещё одна утилита, которую можно использовать для обновления портов. Она устанавливает набор приложений для управления портами. Однако она зависит от Ruby. Для установки порта выполните:

```
# cd /usr/ports/ports-mgmt/portupgrade
# make install clean
```

Прежде чем выполнять обновление с помощью этой утилиты, рекомендуется проверить список установленных портов с помощью `pkgdb -F` и исправить все обнаруженные несоответствия.

Для обновления всех устаревших портов, установленных в системе, используйте `portupgrade -a`. Или добавьте `-i` для запроса подтверждения каждого обновления:

```
# portupgrade -ai
```

Для обновления только указанного приложения вместо всех доступных портов используйте `portupgrade pkgname`. Очень важно включить опцию `-R`, чтобы сначала обновить все порты, необходимые для данного приложения:

```
# portupgrade -R firefox
```

Если указан `-P`, Portupgrade ищет доступные пакеты в локальных каталогах, перечисленных в `PKG_PATH`. Если локально пакеты не найдены, он загружает их с удалённого сайта. Если пакеты не могут быть найдены локально или загружены удалённо, Portupgrade использует порты. Чтобы полностью избежать использования портов, укажите `-PP`. Последний набор опций предписывает Portupgrade прервать работу, если пакеты недоступны:

```
# portupgrade -PP gnome3
```

Чтобы только загрузить distfiles портов или пакеты (если указан `-P`), без сборки или установки, используйте `-F`. Для получения дополнительной информации о всех доступных опциях обратитесь к руководству `portupgrade`.

Дополнительную информацию о ports-mgmt/portupgrade можно найти в его `pkg-descr`.

4.5.5. Порты и дисковое пространство

Использование коллекции портов со временем приводит к расходу дискового пространства. После сборки и установки порта выполнение команды `make clean` в скелете порта очистит временный каталог `work`. При использовании Portmaster для установки порта этот каталог будет удалён автоматически, если не указан параметр `-K`. Если установлен Portupgrade, следующая команда удалит все каталоги `work` в локальной копии коллекции портов:

```
# portsclean -C
```

Кроме того, устаревшие файлы исходных дистрибутивов со временем накапливаются в `/usr/ports/distfiles`. Чтобы использовать Portupgrade для удаления всех distfiles, на которые больше нет ссылок из портов:

```
# portsclean -D
```

Portupgrade может удалить все distfiles, на которые нет ссылок из портов, установленных в системе:

```
# portsclean -DD
```

Если установлен Portmaster, используйте:

```
# portmaster --clean-distfiles
```

По умолчанию эта команда интерактивна и запрашивает подтверждение пользователя на удаление distfile.

В дополнение к этим командам, ports-mgmt/pkg_cutleaves автоматизирует задачу удаления установленных портов, которые больше не нужны.

4.6. Сборка пакетов с poudriere

poudriere — это утилита с лицензией `BSD` для создания и тестирования пакетов FreeBSD. Она использует механизм jail в FreeBSD для настройки изолированных сред сборки. Эти jail

могут использоваться для сборки пакетов для версий FreeBSD, отличных от версии системы, на которой установлена poudriere, а также для сборки пакетов под i386 на хосте с архитектурой amd64. После сборки пакеты располагаются в структуре, идентичной официальным зеркалам. Эти пакеты могут использоваться pkg(8) и другими инструментами управления пакетами.

poudriere устанавливается с помощью пакета ports-mgmt/poudriere или порта. Установка включает пример файла конфигурации `/usr/local/etc/poudriere.conf.sample`. Скопируйте этот файл в `/usr/local/etc/poudriere.conf`. Отредактируйте скопированный файл в соответствии с локальной конфигурацией.

Хотя `ZFS` не является обязательным для системы, на которой запущен poudriere, его использование даёт преимущества. При использовании `ZFS` необходимо указать `ZPOOL` в `/usr/local/etc/poudriere.conf`, а `FREEBSD_HOST` следует установить на ближайшее зеркало. Определение `CCACHE_DIR` позволяет использовать devel/ccache для кэширования компиляции и сокращения времени сборки часто компилируемого кода. Может быть удобно разместить наборы данных poudriere в изолированном дереве, смонтированном в `/poudriere`. Значения по умолчанию для остальных параметров конфигурации являются приемлемыми.

Количество обнаруженных ядер процессора определяет, сколько сборок будет выполняться параллельно. Обеспечьте достаточный объём виртуальной памяти, используя `RAM` или файл подкачки. Если виртуальная память закончится, компиляционные окружения остановятся и будут уничтожены, что приведёт к странным сообщениям об ошибках.

4.6.1. Инициализация Jail и дерева портов

После настройки инициализируйте poudriere, чтобы он установил jail с требуемым деревом FreeBSD и деревом портов. Укажите имя для jail с помощью `-j`, а версию FreeBSD — с помощью `-v`. На системах под управлением FreeBSD/amd64 архитектуру можно задать с помощью `-a`, указав `i386` или `amd64`. По умолчанию используется архитектура, отображаемая командой `uname`.

```
# poudriere jail -c -j 13amd64 -v 13.1-RELEASE
[00:00:00] Creating 13amd64 fs at /poudriere/jails/13amd64... done
[00:00:00] Using pre-distributed MANIFEST for FreeBSD 13.1-RELEASE amd64
[00:00:00] Fetching base for FreeBSD 13.1-RELEASE amd64
/poudriere/jails/13amd64/fromftp/base.txz              125 MB 4110 kBps    31s
[00:00:33] Extracting base... done
[00:00:54] Fetching src for FreeBSD 13.1-RELEASE amd64
/poudriere/jails/13amd64/fromftp/src.txz               154 MB 4178 kBps    38s
[00:01:33] Extracting src... done
[00:02:31] Fetching lib32 for FreeBSD 13.1-RELEASE amd64
/poudriere/jails/13amd64/fromftp/lib32.txz              24 MB 3969 kBps    06s
[00:02:38] Extracting lib32... done
[00:02:42] Cleaning up... done
[00:02:42] Recording filesystem state for clean... done
[00:02:42] Upgrading using ftp
/etc/resolv.conf -> /poudriere/jails/13amd64/etc/resolv.conf
Looking up update.FreeBSD.org mirrors... 3 mirrors found.
```

```
Fetching public key from update4.freebsd.org... done.
Fetching metadata signature for 13.1-RELEASE from update4.freebsd.org... done.
Fetching metadata index... done.
Fetching 2 metadata files... done.
Inspecting system... done.
Preparing to download files... done.
Fetching 124
patches.....10....20....30....40....50....60....70....80....90....100....110....120..
done.
Applying patches... done.
Fetching 6 files... done.
The following files will be added as part of updating to
13.1-RELEASE-p1:
/usr/src/contrib/unbound/.github
/usr/src/contrib/unbound/.github/FUNDING.yml
/usr/src/contrib/unbound/contrib/drop2rpz
/usr/src/contrib/unbound/contrib/unbound_portable.service.in
/usr/src/contrib/unbound/services/rpz.c
/usr/src/contrib/unbound/services/rpz.h
/usr/src/lib/libc/tests/gen/spawnp_enoexec.sh
The following files will be updated as part of updating to
13.1-RELEASE-p1:
[···]
Installing updates...Scanning //usr/share/certs/blacklisted for certificates...
Scanning //usr/share/certs/trusted for certificates...
 done.
13.1-RELEASE-p1
[00:04:06] Recording filesystem state for clean... done
[00:04:07] Jail 13amd64 13.1-RELEASE-p1 amd64 is ready to be used
```

```
# poudriere ports -c -p local -m git+https
[00:00:00] Creating local fs at /poudriere/ports/local... done
[00:00:00] Checking out the ports tree... done
```

На одном компьютере poudriere может собирать порты с различными конфигурациями, в нескольких jail и из разных деревьев портов. Пользовательские конфигурации для таких комбинаций называются *наборами*. Подробности смотрите в разделе CUSTOMIZATION руководства poudriere(8) после установки ports-mgmt/poudriere или ports-mgmt/poudriere-devel.

Базовая конфигурация, представленная здесь, размещает отдельные файлы `make.conf` для каждого jail, порта и набора в `/usr/local/etc/poudriere.d`. Имя файла в этом примере формируется путем объединения имени jail, имени порта и имени набора: `13amd64-local-workstation-make.conf`. Системный `make.conf` и этот новый файл объединяются во время сборки, чтобы создать `make.conf`, используемый build jail.

Пакеты для сборки указываются в файле `13amd64-local-workstation-pkglist` (для портов с FLAVORS можно указать @FLAVOR):

```
editors/emacs
devel/git
devel/php-composer2@php82
ports-mgmt/pkg
...
```

Настраиваются параметры и зависимости для указанных портов:

```
# poudriere options -j 13amd64 -p local -z workstation -f 13amd64-local-workstation-
pkglist
```

Наконец, пакеты собираются и создаётся репозиторий пакетов:

```
# poudriere bulk -j 13amd64 -p local -z workstation -f 13amd64-local-workstation-
pkglist
```

Во время работы нажатие `Ctrl` + `t` отображает текущее состояние сборки. poudriere также создаёт файлы в `/poudriere/logs/bulk/jailname`, которые можно использовать с веб-сервером для отображения информации о сборке.

По завершении новые пакеты становятся доступными для установки из репозитория poudriere.

Для получения дополнительной информации об использовании poudriere см. poudriere(8) и основной веб-сайт: https://github.com/freebsd/poudriere/wiki.

4.6.2. Настройка клиентов pkg для использования репозитория poudriere

Хотя можно использовать как пользовательский репозиторий вместе с официальным, иногда полезно отключить официальный репозиторий. Это делается путём создания конфигурационного файла, который переопределяет и отключает официальный конфигурационный файл. Создайте `/usr/local/etc/pkg/repos/FreeBSD.conf` со следующим содержимым:

```
FreeBSD: {
    enabled: no
}
```

Обычно проще всего предоставлять доступ к репозиторию poudriere клиентским машинам через HTTP. Настройте веб-сервер для обслуживания каталога пакетов, например: `/usr/local/poudriere/data/packages/13amd64`, где `13amd64` — это название сборки.

Если URL репозитория пакетов: `https://pkg.example.com/13amd64`, то файл конфигурации репозитория в `/usr/local/etc/pkg/repos/custom.conf` будет выглядеть так:

```
custom: {
    url: "https://pkg.example.com/13amd64",
    enabled: yes,
}
```

Если нежелательно предоставлять доступ к репозиторию пакетов через интернет, можно использовать протокол `file://` для прямого указания на репозиторий:

```
custom: {
    url: "file:///usr/local/poudriere/data/packages/11amd64",
    enabled: yes,
}
```

4.7. Пост-установочные вопросы

Независимо от того, было ли программное обеспечение установлено из бинарного пакета или порта, большинство сторонних приложений требуют некоторой настройки после установки. Следующие команды и расположения могут помочь определить, что было установлено вместе с приложением.

- Большинство приложений устанавливают как минимум один конфигурационный файл по умолчанию в `/usr/local/etc`. В случаях, когда приложение имеет большое количество конфигурационных файлов, создаётся подкаталог для их хранения. Часто устанавливаются примеры конфигурационных файлов, которые имеют окончание, например, `.sample`. Конфигурационные файлы следует просмотреть и, возможно, отредактировать в соответствии с потребностями системы. Для редактирования файла-примера сначала скопируйте его без расширения `.sample`.
- Приложения, которые предоставляют документацию, устанавливают её в `/usr/local/share/doc`, а многие приложения также устанавливают страницы руководств. Перед продолжением следует ознакомиться с этой документацией.
- Некоторые приложения запускают службы, которые необходимо добавить в `/etc/rc.conf` перед запуском приложения. Обычно такие приложения устанавливают скрипт запуска в `/usr/local/etc/rc.d`. Дополнительную информацию можно найти в Запуск служб.

По замыслу, приложения не запускают свои стартовые скрипты при установке, а также не выполняют скрипты остановки при удалении или обновлении. Это решение остаётся на усмотрение системного администратора.

- Пользователи csh(1) должны выполнить `rehash`, чтобы перестроить список известных двоичных файлов в `PATH` оболочки.
- Используйте `pkg info`, чтобы определить, какие файлы, man-страницы и бинарные файлы были установлены вместе с приложением.

4.8. Работа с неработающими портами

Когда порт не собирается или не устанавливается, попробуйте следующее:

1. Поищите, есть ли исправление для порта в базе данных отчётов о проблемах. Если оно есть, применение предложенного исправления может решить проблему.
2. Обратитесь к сопровождающему порта за помощью. Введите `make maintainer` в скелете портов или прочитайте `Makefile` порта, чтобы найти адрес электронной почты сопровождающего. Не забудьте включить вывод, предшествующий ошибке, в письмо сопровождающему.

Некоторые порты поддерживаются не отдельными людьми, а группой сопровождающих, представленной рассылкой. Многие, но не все, такие адреса выглядят как freebsd-listname@FreeBSD.org. Учтите это при отправке письма.

В частности, порты, поддерживаемые ports@FreeBSD.org, не курируются конкретным человеком. Вместо этого исправления и поддержка поступают от сообщества в целом, участники которого подписаны на этот список рассылки. Добровольцы всегда нужны!

Если на письмо не получен ответ, используйте Bugzilla для отправки отчёта об ошибке, следуя инструкциям в Составление отчётов о проблемах в FreeBSD.

3. Исправьте это! В Руководстве FreeBSD по созданию портов содержится подробная информация об инфраструктуре портов, так что можно исправить случайно сломанный порт или даже предложить свой собственный! Установите пакет вместо порта по инструкции в разделе Использование pkg для управления бинарными пакетами.

Глава 5. Система X Window

5.1. Обзор

Установка FreeBSD с помощью bsdinstall(8) не включает автоматическую установку графического интерфейса пользователя. В этой главе описано, как установить и настроить Xorg(1), который предоставляет открытую реализацию системы X Window (часто сокращается до X11), используемую для создания графической среды.

Прежде чем читать эту главу, необходимо:

- Знать, как устанавливать дополнительное стороннее программное обеспечение, как описано в Установка приложений: Пакеты и Порты.

Прочтите эту главу, чтобы узнать:

- Как выбрать и установить драйверы для графического процессора (GPU).
- Различные компоненты X Window System и их взаимодействие.
- Как установить и настроить сервер X.org.
- Как установить и настроить шрифты в системе X Windows.

5.2. Драйверы видеокарт

***аннотация**: Определите свой графический процессор, порт, предоставляющий для него драйвер, установите его, затем включите его запуск при последующей загрузке с помощью sysrc(8).*

Прежде чем FreeBSD сможет отображать графическую среду, ей необходим модуль ядра для управления графическим процессором. Графические драйверы являются быстро развивающимся кроссплатформенным программным обеспечением, поэтому они разрабатываются и распространяются отдельно от базовой системы FreeBSD.

Следующая таблица показывает различные графические процессоры, поддерживаемые FreeBSD, их соответствующие модули, и порт предоставляет их поддержку:

Таблица 8. Поддерживаемые графические устройства

Тип	Лицензия	Модуль	Порт
Intel®	Открытое программное обеспечение	`i915kms`	graphics/drm-kmod
AMD®	Открытое программное обеспечение	`amdgpu` или `radeonkms`	graphics/drm-kmod

Тип	Лицензия	Модуль	Порт
NVIDIA®	Проприетарный	`nvidia-drm`, `nvidia-modeset` или `nvidia`	graphics/nvidia-drm-kmod или x11/nvidia-driver
Буфер кадров системной консоли	Открытое программное обеспечение	`scfb`	x11-drivers/xf86-video-scfb
Расширение VESA BIOS	Открытое программное обеспечение	`vesa`	x11-drivers/xf86-video-vesa
VirtualBox®	Открытое программное обеспечение	`vboxvideo`	emulators/virtualbox-ose-additions
VMware®	Открытое программное обеспечение	`vmwgfx`	x11-drivers/xf86-video-vmware

Поддерживается несколько поколений технологий драйверов.

- Драйверы прямой визуализации (Direct Rendering), позволяющие использовать PRIME offloading. PRIME позволяет сосуществовать нескольким поставщикам графической обработки. PRIME подробно описан в Графические конфигурация.
- Kernel Modesetting (KMS) Это позволяет драйверу напрямую задавать режим отображения. Это необходимо для поддержки приостановки и возобновления работы при использовании драйвера консоли vt(4).
- User Modesetting Самая старая категория драйверов по-прежнему поддерживается, однако они могут использоваться только с консолью sc(4) и более старыми версиями графической среды Xorg(1).

Следующая команда позволяет определить, какой графический процессор установлен в системе:

```
% pciconf -lv | grep -B3 display
```

Вывод должен быть похож на следующий:

```
vgapci1@pci0:0:2:0:     class=0x030000 rev=0x0c hdr=0x00 vendor=0x8086 device=0x46a6
subvendor=0x1028 subdevice=0x0b29
    vendor     = 'Intel Corporation'
    device     = 'Alder Lake-P GT2 [Iris Xe Graphics]'
    class      = display
```

Подробные инструкции по установке и включению этих драйверов приведены в следующих подразделах.

Если графический процессор не поддерживается драйверами Intel®, AMD® или NVIDIA®, следует использовать модули SCFB или VESA. Модуль SCFB должен использоваться при загрузке в режиме UEFI, а модуль VESA — при загрузке в режиме BIOS.

Эту команду можно использовать для проверки режима загрузки:

```
% sysctl machdep.bootmethod
```

Вывод должен быть похож на следующий:

```
machdep.bootmethod: UEFI
```

5.2.1. Intel® Graphics

Пакет graphics/drm-kmod косвенно предоставляет набор модулей ядра для использования с Intel® Graphics. В последних версиях эти модули могут работать совместно с другими графическими процессорами в режиме PRIME без дополнительной настройки.

Драйвер Intel® Graphics можно установить, выполнив следующую команду:

```
# pkg install drm-kmod
```

Затем добавьте модуль в файл /etc/rc.conf, выполнив следующую команду:

```
# sysrc kld_list+=i915kms
```

5.2.2. AMD® Graphics

Пакет graphics/drm-kmod косвенно предоставляет модули ядра для набора графических процессоров AMD®. В зависимости от поколения оборудования могут использоваться модули `amdgpu` или `radeonkms`. Проект FreeBSD поддерживает матрицу поддержки AMD graphics, чтобы показать уровень поддержки и для определения необходимого драйвера.

Драйверы AMD® Graphics можно установить, выполнив следующую команду:

```
# pkg install drm-kmod
```

Активируйте текущий модуль, добавив его в файл /etc/rc.conf, для чего необходимо выполнить следующую команду:

```
# sysrc kld_list+=amdgpu
```

Для старых видеокарт (до HD7000/Tahiti) активируйте старый модуль, добавив модуль в файл /etc/rc.conf, выполнив следующую команду:

```
# sysrc kld_list+=radeonkms
```

5.2.3. NVIDIA® Graphics

NVIDIA® выпускает автономные или дискретные графические процессоры и предоставляет проприетарный драйвер для FreeBSD. Коллекция портов FreeBSD предоставляет более десятилетия драйверов для поддержки поколений графики NVIDIA.

Администраторам следует устанавливать последнюю версию драйвера, поддерживаемую их оборудованием.

Следующая таблица показывает порт, содержащий драйвер, рекомендуемый для загрузки модуль ядра, и ссылку на список оборудования, поддерживаемого этим драйвером:

Таблица 9. Поддерживаемые версии драйверов NVIDIA® Graphics

Порт	Модуль	Поддерживаемое оборудование
graphics/nvidia-drm-kmod	`nvidia` или `nvidia-modeset`	поддерживаемое оборудование
x11/nvidia-driver-470	`nvidia-modeset`	поддерживаемое оборудование
x11/nvidia-driver-390 или x11/nvidia-secondary-driver-390	`nvidia-modeset`	поддерживаемое оборудование
x11/nvidia-driver-340	`nvidia`	поддерживаемое оборудование
x11/nvidia-driver-304	`nvidia`	поддерживаемое оборудование

Последнюю версию драйвера NVIDIA® Graphics можно установить, выполнив следующую команду:

```
# pkg install nvidia-drm-kmod
```

Чтобы активировать драйвер, добавьте модуль в файл /etc/rc.conf, выполнив следующую команду:

```
# sysrc kld_list+=nvidia-drm
```

Это драйвер прямой визуализации KMS.

Kernel modesetting — это опция для установки графического режима в ядре. Включите её для последующих загрузок с помощью следующей настройки loader.conf(5):

```
hw.nvidiadrm.modeset="1"
```

Как PRIME, так и Wayland требуют режим kernel modesetting.

Предыдущие версии драйвера не поддерживают прямую визуализацию (Direct Rendering). Вместо этого используйте модуль modesetting, выполнив следующую команду:

```
# sysrc kld_list+=nvidia-modeset
```

Если требуются драйверы Nvidia версии ниже 390, учтите, что они не поддерживают режим kernel modesetting, и поэтому должны использоваться с устаревшим драйвером консоли sc(4) и версией x11/xorg-server ниже 1.20.

Затем добавьте модуль в файл /etc/rc.conf, выполнив следующую команду:

```
# sysrc kld_list+=nvidia
```

Некоторым графическим картам требуется специальное микропрограммное обеспечение, которое необходимо загрузить для корректной работы устройства. Утилита fwget(8) может загрузить и установить необходимые файлы в каталог /boot/firmware. fwget(8) выполнит сканирование устройств 'PCI' и загрузит микропрограммное обеспечение для тех устройств, которым оно нужна.

5.3. Обзор системы X Window

Система X Window представляет собой наследуемый графический стек для платформ UNIX®, поддерживающий новейшие технологии при сохранении совместимости с приложениями, созданными за многие поколения. Приложения, включая компоненты рабочего стола, обслуживаются сервером Xorg(1). Данная система обладает сетевыми возможностями, и её различные компоненты могут взаимодействовать через сети.

5.4. Установка сервера X.org

***аннотация**: Для размещения рабочего стола должен быть установлен сервер x11/xorg. Пользователи должны быть добавлены в группу `video` для его использования.*

После установки и включения графического драйвера сервер X.org может быть установлен как метапакет или скомпилирован локально с помощью дерева портов.

Полный метапакет можно быстро установить, но с меньшими возможностями для настройки:

```
# pkg install xorg
```

При такой установке будет установлена полная система X Window System, включая традиционный оконный менеджер twm(1) и сопутствующий традиционный набор приложений для рабочего стола. Большинству пользователей захочется установить и настроить современный рабочий стол по своему выбору.

Текущий пользователь должен быть членом группы `video`, чтобы запускать графическую среду. Чтобы добавить пользователя в группу `video`, выполните следующую команду:

```
# pw groupmod video -m username
```

Для запуска системы X Window System используйте startx(1) из пакета x11/xinit, либо установите и настройте дисплейный менеджер для запуска графического входа при загрузке.

Меньшая версия системы X Windows, подходящая для опытных пользователей, доступна в пакете x11/xorg-minimal. Большинство документов, библиотек и приложений установлено не будет. Некоторым приложениям требуются эти дополнительные компоненты для работы.

5.5. Конфигурация X.org

***аннотация**: Если настройки по умолчанию для монитора или устройств ввода не устраивают, в рабочем окружении включены графические интерфейсы для их настройки, либо их можно настроить вручную.*

Сервер X.org поддерживает большинство распространённых графических процессоров, мониторов и устройств ввода. Сначала попробуйте настройки по умолчанию. В этом подразделе представлен обзор их конфигурации.

5.5.1. Файлы конфигурации X.org

Исторически сервер X.org настраивался с помощью файлов в /usr/local/etc/X11/. Это до сих пор поддерживается для особых случаев, но конфликтует с динамической автоконфигурацией.

Не создавайте конфигурацию сервер X.org в файле xorg.conf и не запускайте `Xorg -configure`, если автоматическая настройка не дала сбоев.

Сервер X.org ищет конфигурационные файлы в нескольких каталогах. /usr/local/etc/X11/ — это **рекомендуемый** каталог для таких файлов в FreeBSD. Использование этого каталога помогает отделять файлы приложений от файлов операционной системы.

Проще использовать несколько файлов, каждый из которых настраивает определённый параметр, чем традиционный единый файл xorg.conf. Эти файлы хранятся в подкаталоге

/usr/local/etc/X11/xorg.conf.d/.

5.5.2. Графические конфигурация

Прямая визуализация предоставляет возможность бесшовного использования дискретного графического процессора (dGPU) совместно с интегрированным графическим процессором (iGPU), что называется PRIME. Драйверы автоматически перекладывают ресурсоемкие задачи на dGPU, когда это требуется, и отключают его питание, когда это возможно.

Чтобы запускать приложения на более мощном GPU в PRIME, используйте переменную окружения `DRI_PRIME=1`.

Если несколько графических драйверов конфликтуют, драйвер графического процессора можно указать в каталоге /usr/local/etc/X11/xorg.conf.d/.

Для настройки драйвера Intel® в файле конфигурации:

Пример 14. Выберите драйвер видео Intel® Graphics в файле

/usr/local/etc/X11/xorg.conf.d/20-intel.conf

```
Section "Device"
    Identifier "Card0"
    Driver     "intel"
EndSection
```

Для настройки драйвера AMD® в файле конфигурации:

Пример 15. Выберите видеодрайвер AMD® Graphics в файле

/usr/local/etc/X11/xorg.conf.d/20-radeon.conf

```
Section "Device"
    Identifier "Card0"
    Driver     "radeon"
EndSection
```

Для настройки драйвера NVIDIA® в файле конфигурации:

Пример 16. Выберите видеодрайвер NVIDIA® Graphics в файле

/usr/local/etc/X11/xorg.conf.d/20-nvidia.conf

```
Section "Device"
    Identifier "Card0"
    Driver     "nvidia"
```

```
EndSection
```

Пакет x11/nvidia-xconfig также может быть использован для базового управления параметрами конфигурации, доступными в драйвере NVIDIA.

Для настройки драйвера SCFB в файле конфигурации:

Пример 17. Выберите видеодрайвер SCFB Graphics в файле

/usr/local/etc/X11/xorg.conf.d/20-scfb.conf

```
Section "Device"
    Identifier "Card0"
    Driver     "scfb"
EndSection
```

Для настройки драйвера VESA в файле конфигурации:

Пример 18. Выберите видеодрайвер VESA Graphics в файле

/usr/local/etc/X11/xorg.conf.d/20-vesa.conf

```
Section "Device"
    Identifier "Card0"
    Driver     "vesa"
EndSection
```

Для настройки нескольких графических процессоров можно добавить параметр `BusID`. Список `ID` шин графических процессоров можно вывести, выполнив:

```
% pciconf -lv | grep -B3 display
```

Вывод должен быть похож на следующий:

```
vgapci0@pci0:0:2:0:     class=0x030000 rev=0x0c hdr=0x00 vendor=0x8086 device=0x46a6
subvendor=0x1028 subdevice=0x0b29
    vendor     = 'Intel Corporation'
    device     = 'Alder Lake-P GT2 [Iris Xe Graphics]'
    class      = display
--
vgapci0@pci0:1:0:0:     class=0x030200 rev=0xa1 hdr=0x00 vendor=0x10de device=0x25b9
subvendor=0x1028 subdevice=0x0b29
    vendor     = 'NVIDIA Corporation'
```

```
    device     = 'GA107GLM [RTX A1000 Laptop GPU]'
    class      = display
```

Пример 19. Выберите драйвер видео Intel® Graphics и драйвер видео NVIDIA® Graphics в файле

/usr/local/etc/X11/xorg.conf.d/20-drivers.conf

```
Section "Device"
    Identifier "Card0"
    Driver     "intel"
    BusID     "pci0:0:2:0"
EndSection

Section "Device"
    Identifier "Card1"
    Driver     "nvidia"
    BusID     "pci0:1:0:0"
EndSection
```

5.5.3. Конфигурация монитора

Почти все мониторы поддерживают стандарт Extended Display Identification Data (EDID). X.org использует EDID для взаимодействия с монитором и определения поддерживаемых разрешений и частот обновления. Затем он выбирает наиболее подходящую комбинацию настроек для работы с этим монитором.

Другие разрешения, поддерживаемые монитором, можно выбрать атомарно после запуска X-сервера с помощью xrandr(1) или в конфигурационных файлах сервера X.org.

Использование RandR (Изменение размера и поворот)

Выполните xrandr(1) в сессии X без параметров, чтобы увидеть список видеовыходов и обнаруженных режимов монитора:

```
% xrandr
```

Вывод должен быть похож на следующий:

```
Screen 0: minimum 320 x 200, current 2560 x 960, maximum 8192 x 8192
LVDS-1 connected 1280x800+0+0 (normal left inverted right x axis y axis) 261mm x 163mm
   1280x800      59.99*+  59.81    59.91    50.00
   1280x720      59.86    59.74
   1024x768      60.00
   1024x576      59.90    59.82
   960x540       59.63    59.82
   800x600       60.32    56.25
```

```
   864x486       59.92    59.57
   640x480       59.94
   720x405       59.51    58.99
   640x360       59.84    59.32
VGA-1 connected primary 1280x960+1280+0 (normal left inverted right x axis y axis)
410mm x 257mm
   1280x1024     75.02    60.02
   1440x900      74.98    60.07
   1280x960      60.00*
   1280x800      74.93    59.81
   1152x864      75.00
   1024x768      75.03    70.07    60.00
   832x624       74.55
   800x600       72.19    75.00    60.32    56.25
   640x480       75.00    72.81    66.67    59.94
   720x400       70.08
HDMI-1 disconnected (normal left inverted right x axis y axis)
DP-1 disconnected (normal left inverted right x axis y axis)
HDMI-2 disconnected (normal left inverted right x axis y axis)
DP-2 disconnected (normal left inverted right x axis y axis)
DP-3 disconnected (normal left inverted right x axis y axis)
```

Это показывает, что выход `VGA-1` используется для отображения экрана с разрешением 1280x960 пикселей и частотой обновления около 60 Гц. Выход `LVDS-1` используется как дополнительный монитор с разрешением 1280x800 пикселей и частотой обновления около 60 Гц. Мониторы не подключены к разъёмам `HDMI-1`, `HDMI-2`, `DP-1`, `DP-2` и `DP-3`.

Любой из других режимов отображения можно выбрать с помощью xrandr(1). Например, для переключения на 1280x1024 при 60 Гц:

```
% xrandr --output LVDS-1 --mode 1280x720 --rate 60
```

Часто черный экран при запуске X можно исправить, добавив шаг `xrandr --auto` в процесс инициализации.

Использование файлов конфигурации X.org

Конфигурацию монитора также можно задать в файле конфигурации.

Установить разрешение экрана 1024x768 в конфигурационном файле:

Пример 20. Установка разрешения экрана в файле

/usr/local/etc/X11/xorg.conf.d/10-monitor.conf

```
Section "Screen"
    Identifier "Screen0"
    Device     "Card0"
```

```
    SubSection "Display"
    Modes      "1024x768"
    EndSubSection
EndSection
```

5.5.4. Конфигурация устройств ввода

Сервер Xorg(1) предоставляет библиотеку x11/libinput, которая представляет собой кроссплатформенное решение для поддержки всех сенсорных, указывающих и клавиатурных устройств в рамках единой библиотеки. Если не указано иное, она загружается автоматически.

Индивидуальные настройки устройств для libinput(4) можно настроить в графическом интерфейсе рабочего стола или вручную с помощью x11/xinput и x11/setxkbmap.

В качестве альтернативы, в пакете x11-drivers категории x11/xf86-input-[foo] доступны более старые, легковесные отдельные драйверы для конкретных устройств ввода. Этот подход требует ручной настройки сервера X.org. Оба метода описаны в данном подразделе.

Использование файла конфигурации атомарного ввода

Устройства, поддерживаемые libinput(4), могут быть настроены с помощью графических утилит, включённых в рабочую среду, или вручную и атомарно во время выполнения с помощью x11/xinput и x11/setxkbmap.

Чтобы узнать, к каким устройствам в данный момент подключён libinput(4), запустите xinput(1) без аргументов:

```
$ xinput
```

Её вывод должен быть похож на следующий:

```
⎡ Virtual core pointer                    id=2    [master pointer  (3)]
⎜   ↳ Virtual core XTEST pointer              id=4    [slave  pointer  (2)]
⎜   ↳ System mouse                            id=7    [slave  pointer  (2)]
⎜   ↳ VEN_0488:00 0488:1031 Mouse             id=11   [slave  pointer  (2)]
⎜   ↳ VEN_0488:00 0488:1031 TouchPad          id=12   [slave  pointer  (2)]
⎣ Virtual core keyboard                   id=3    [master keyboard (2)]
    ↳ Virtual core XTEST keyboard             id=5    [slave  keyboard (3)]
    ↳ System keyboard multiplexer             id=6    [slave  keyboard (3)]
    ↳ Power Button                            id=8    [slave  keyboard (3)]
    ↳ Sleep Button                            id=9    [slave  keyboard (3)]
    ↳ AT keyboard                             id=10   [slave  keyboard (3)]
```

Все поддерживаемые этими устройствами настройки предоставляются в виде свойств, которые можно выводить и устанавливать атомарно. Указывающие устройства имеют множество настраиваемых свойств, клавиатурам обычно не требуется ничего.

Для настройки клавиатуры обратитесь к setxkbmap(1).

Удовлетворившись своей конфигурацией, просто добавьте строки в скрипт инициализации X, например, в ~/.Xsession или ~/.xinitrc.

Использование файлов конфигурации X.org

Некоторые среды рабочего стола (например, KDE Plasma) предоставляют графический интерфейс для настройки этих параметров. Проверьте, доступен ли такой вариант, прежде чем приступать к ручному редактированию конфигурации.

Например, чтобы в ручную настроить у сервера X.org раскладку клавиатуры:

Пример 21. Установка раскладки клавиатуры

/usr/local/etc/X11/xorg.conf.d/00-keyboard.conf

```
Section "InputClass"
        Identifier "Keyboard1"
        MatchIsKeyboard "on"
        Option "XkbLayout" "es, fr"
        Option "XkbModel" "pc104"
        Option "XkbVariant" ",qwerty"
        Option "XkbOptions" "grp:win_space_toggle"
EndSection
```

5.6. Использование шрифтов в системе X Window

***аннотация**: Дополнительные шрифты можно установить из категории x11-fonts или разместить в ~/.fonts. Они становятся доступны немедленно для современных приложений. Также доступна и описана конфигурация для старых приложений.*

Система X Window предоставляет библиотеку интерфейса X FreeType (Xft(3)) для отображения векторных или контурных шрифтов, а также традиционную систему X Logical Font Description, сохраняющую совместимость с поколениями приложений и шрифтов.

Пользователей в основном интересуют два типа шрифтов:

- Шрифты OpenType или TrueType® предназначены для отображения на экране.
- Шрифты Adobe® PostScript® Type 1 предназначены для печати на бумаге.

Это векторные или контурные шрифты, также существуют растровые шрифты.

Коллекция портов FreeBSD включает в себя широкий и постоянно растущий каталог бесплатных высококачественных шрифтов, доступных для установки в категории x11-fonts.

Системные пакеты шрифтов, установленные из коллекции портов, находятся в

/usr/local/share/fonts/. Шрифты для отдельного пользователя можно разместить в ~/.fonts/ или ~/.local/share/fonts/.

Шрифты в каталоге или подкаталогах станут доступны для немедленного использования после перестроения кэша информации о шрифтах. Для принудительного запуска этого процесса выполните:

```
% fc-cache
```

В дереве портов доступно множество бесплатных высококачественных шрифтов обоих типов, которые можно легко использовать с X Window System. В этой главе представлен краткий обзор обоих типов, а также настройка интерфейса X FreeType.

Для получения дополнительной информации о том, как установить и настроить шрифты в FreeBSD, прочитайте статью Шрифты и FreeBSD.

5.6.1. Шрифты TrueType®

В X.org есть встроенная поддержка отображения шрифтов TrueType®. Для этого доступны два различных модуля. В данном примере используется модуль freetype, так как он более согласован с другими механизмами отображения шрифтов. Чтобы включить модуль freetype, добавьте следующую строку в раздел "Module" файла /usr/local/etc/X11/xorg.conf.d/90-fonts.conf.

```
Load  "freetype"
```

Теперь создайте каталог для шрифтов TrueType® (например, /usr/local/share/fonts/TrueType) и скопируйте все шрифты TrueType® в этот каталог. Учтите, что шрифты TrueType® нельзя напрямую брать из Apple® Mac®; они должны быть в формате UNIX®/MS-DOS®/Windows® для использования в X.org. После копирования файлов в этот каталог используйте mkfontscale для создания fonts.dir, чтобы X-сервер знал, что новые файлы установлены. mkfontscale можно установить как пакет:

```
# pkg install mkfontscale
```

Затем создайте индекс файлов шрифтов X в каталоге:

```
# cd /usr/local/share/fonts/TrueType
# mkfontscale
```

Теперь добавьте каталог TrueType® в путь шрифтов. Это делается так же, как описано в Шрифты Type1:

```
% xset fp+ /usr/local/share/fonts/TrueType
```

```
% xset fp rehash
```

или добавьте строку `FontPath` в файл xorg.conf.

Теперь Gimp, LibreOffice и все другие X-приложения должны распознавать установленные шрифты TrueType®. Очень мелкие шрифты (например, текст на веб-странице с высоким разрешением) и очень крупные шрифты (в LibreOffice) теперь будут выглядеть значительно лучше.

5.6.2. Шрифты Type1

Коллекция шрифтов URW (x11-fonts/urwfonts) включает высококачественные версии стандартных шрифтов type1 (Times Roman™, Helvetica™, Palatino™ и другие). Коллекция Freefonts (x11-fonts/freefonts) содержит гораздо больше шрифтов, но большинство из них предназначены для использования в графических программах, таких как Gimp, и не являются достаточно полными для использования в качестве экранных шрифтов.

Для установки указанных коллекций шрифтов Type1 из бинарных пакетов выполните следующие команды:

```
# pkg install urwfonts
```

Аналогично с коллекцией freefont или другими. Чтобы X-сервер, сконфигурированный в ручную) обнаружил эти шрифты, добавьте соответствующую строку в файл конфигурации X-сервера (/usr/local/etc/X11/xorg.conf.d/90-fonts.conf), которая выглядит следующим образом:

```
Section "Files"
  FontPath "/usr/local/share/fonts/urwfonts/"
EndSection
```

Или в командной строке в сеансе X выполните:

```
% xset fp+ /usr/local/share/fonts/urwfonts
% xset fp rehash
```

Это будет работать, но изменения пропадут после завершения сеанса X, если не добавить их в файл автозагрузки (~/.xinitrc для обычного сеанса `startx` или ~/.xsession при входе через графический менеджер входа, например XDM). Третий способ — использовать новый файл /usr/local/etc/fonts/local.conf, как показано в Анти-алиасинг шрифтов.

5.6.3. Анти-алиасинг шрифтов

Все шрифты в X.org, находящиеся в /usr/local/share/fonts/ и ~/.fonts/, автоматически становятся доступными для сглаживания приложениям, поддерживающим Xft. Большинство современных приложений поддерживают Xft, включая KDE, GNOME и Firefox.

Для управления тем, какие шрифты сглаживаются, или для настройки свойств сглаживания создайте (или отредактируйте, если он уже существует) файл /usr/local/etc/fonts/local.conf. С помощью этого файла можно настроить несколько расширенных возможностей системы шрифтов Xft; в этом разделе описаны лишь некоторые простые варианты. Подробнее см. в fonts-conf(5).

Этот файл должен быть в формате XML. Внимательно следите за регистром символов и убедитесь, что все теги правильно закрыты. Файл начинается со стандартного заголовка XML, за которым следует определение DOCTYPE, а затем тег `<fontconfig>`:

```
<?xml version="1.0"?>
      <!DOCTYPE fontconfig SYSTEM "fonts.dtd">
      <fontconfig>
```

Как уже упоминалось, все шрифты в /usr/local/share/fonts/, а также в ~/.fonts/ уже доступны для приложений, поддерживающих Xft. Чтобы добавить другой каталог вне этих двух деревьев каталогов, добавьте строку следующего вида в /usr/local/etc/fonts/local.conf:

```
<dir>/path/to/my/fonts</dir>
```

После добавления новых шрифтов, особенно новых каталогов со шрифтами, перестройте кэши шрифтов:

```
# fc-cache -f
```

Антиалиасинг делает границы слегка размытыми, что улучшает читаемость очень мелкого текста и убирает "лесенки" у крупного текста, но может вызывать усталость глаз, если применяется к обычному тексту. Чтобы исключить из антиалиасинга шрифты размером меньше 14 пунктов, добавьте следующие строки:

```
    <match target="font">
        <test name="size" compare="less">
        <double>14</double>
        </test>
        <edit name="antialias" mode="assign">
        <bool>false</bool>
        </edit>
    </match>
    <match target="font">
        <test name="pixelsize" compare="less" qual="any">
        <double>14</double>
        </test>
        <edit mode="assign" name="antialias">
        <bool>false</bool>
        </edit>
```

```
    </match>
```

Интервал для некоторых моноширинных шрифтов также может быть некорректным при сглаживании. Это особенно актуально для KDE. Одно из возможных решений — принудительно установить интервал для таких шрифтов в значение 100. Добавьте следующие строки:

```
    <match target="pattern" name="family">
       <test qual="any" name="family">
           <string>fixed</string>
       </test>
       <edit name="family" mode="assign">
           <string>mono</string>
       </edit>
    </match>
    <match target="pattern" name="family">
        <test qual="any" name="family">
        <string>console</string>
        </test>
        <edit name="family" mode="assign">
        <string>mono</string>
        </edit>
    </match>
```

(это создаёт псевдонимы для других распространённых названий моноширинных шрифтов как "mono"), а затем добавляем:

```
    <match target="pattern" name="family">
         <test qual="any" name="family">
         <string>mono</string>
         </test>
         <edit name="spacing" mode="assign">
         <int>100</int>
         </edit>
     </match>
```

Некоторые шрифты, такие как Helvetica, могут иметь проблемы при сглаживании. Обычно это проявляется в виде шрифта, который кажется разрезанным пополам по вертикали. В худшем случае это может привести к сбою приложений. Чтобы избежать этого, рекомендуется добавить следующее в local.conf:

```
    <match target="pattern" name="family">
         <test qual="any" name="family">
         <string>Helvetica</string>
         </test>
         <edit name="family" mode="assign">
         <string>sans-serif</string>
```

```
        </edit>
    </match>
```

После редактирования local.conf убедитесь, что файл завершается тегом `</fontconfig>`. Если этого не сделать, изменения будут проигнорированы.

Пользователи могут добавить индивидуальные настройки, создав собственный файл ~/.config/fontconfig/fonts.conf. Этот файл использует тот же формат XML, который описан выше.

Последний момент: при использовании ЖК-экрана может потребоваться субпиксельная выборка. По сути, это раздельная обработка красного, зелёного и синего компонентов (разделённых горизонтально) для улучшения горизонтального разрешения; результат может быть впечатляющим. Чтобы включить эту функцию, добавьте следующую строку в файл local.conf:

```
    <match target="font">
        <test qual="all" name="rgba">
        <const>unknown</const>
        </test>
        <edit name="rgba" mode="assign">
        <const>rgb</const>
        </edit>
    </match>
```

В зависимости от типа дисплея, возможно, потребуется изменить `rgb` на `bgr`, `vrgb` или `vbgr`: поэкспериментируйте и выберите наиболее подходящий вариант.

Глава 6. Wayland на FreeBSD

6.1. Обзор

Установка FreeBSD с помощью bsdinstall не включает автоматически графический интерфейс пользователя. В этой главе описано, как выбрать, установить и настроить композитор Wayland, который предоставляет графическую среду.

Прежде чем читать эту главу, необходимо:

- Знать, как установить дополнительное стороннее программное обеспечение.
- Знать, как определить и настроить драйверы для вашего графического оборудования.

Прочтите эту главу, чтобы узнать:

- Как настроить FreeBSD для размещения графической среды Wayland.
- Как установить и настроить композитор Wayland.
- Как запускать программы, предназначенные для старой версии X Window System.
- Как настроить удалённый доступ к графической среде Wayland.

6.2. Обзор Wayland

Wayland — это коммуникационный протокол, который может заменить сервер дисплеев, такой как X.org. Он отличается от X.org несколькими важными способами. Во-первых, Wayland — это только протокол, который выступает в роли посредника между клиентами, используя механизм, который устраняет зависимость от X-сервера. X.org включает в себя как протокол X11, используемый для работы с удалёнными дисплеями, так и X-сервер, который принимает соединения и отображает окна. В случае с Wayland, композитор или оконный менеджер предоставляет сервер дисплеев вместо традиционного X-сервера.

Поскольку Wayland не является X-сервером, для удалённого управления рабочим столом традиционные соединения с экраном X потребуют использования других методов, таких как VNC или RDP. Во-вторых, Wayland может управлять композитными взаимодействиями между клиентами и композитором как отдельная сущность, которой не требуется поддержка протоколов X.

Wayland относительно нов, и не все программы были обновлены для нативной работы без поддержки `Xwayland`. Поскольку Wayland не предоставляет X-сервер, а ожидает, что композиторы обеспечат эту поддержку, оконные менеджеры X11, которые ещё не поддерживают Wayland, потребуют, чтобы `Xwayland` не запускался с параметром `-rootless`. Удаление параметра `-rootless` восстановит поддержку оконных менеджеров X11.

Текущий драйвер NVIDIA® должен работать с большинством композиторов wlroots, но может быть немного нестабильным и не поддерживать все функции на данный момент. Требуются добровольцы для помощи в работе над NVIDIA® DRM.

В настоящее время множество программного обеспечения, включая Firefox, работает с минимальными проблемами в Wayland. Также доступны несколько окружений рабочего стола, например, замена Compiz Fusion под названием Wayfire и замена менеджера окон i3 — Sway.

По состоянию на май 2021 года plasma5-kwin поддерживает Wayland в FreeBSD. Для использования Plasma под Wayland используйте параметр `startplasma-wayland` с `ck-launch-session` и подключите dbus следующим образом: `dbus-launch --exit-with-x11 ck-launch-session startplasma-wayland`, чтобы заставить это работать.

Для композиторов необходимо наличие ядра с поддержкой драйвера evdev(4) для использования функциональности привязки клавиш. Он включён по умолчанию в ядро GENERIC; однако, если ядро было изменено и поддержка evdev(4) была удалена, потребуется загрузить модуль evdev(4). Кроме того, пользователям `Wayland` необходимо быть членами группы `video`. Чтобы быстро внести это изменение, используйте команду `pw`:

```
pw groupmod video -m user
```

Установка Wayland проста; сам протокол не требует значительной настройки. Большая часть композиции будет зависеть от выбранного композитора. Установив `seatd` сейчас, можно пропустить один шаг в процессе установки и настройки композитора, так как `seatd` необходим для предоставления непривилегированного доступа к некоторым устройствам.

Все описанные здесь композиторы должны работать с драйверами с открытым исходным кодом graphics/drm-kmod; однако видеокарты NVIDIA® могут иметь проблемы при использовании проприетарных драйверов. Для начала установите следующие пакеты:

```
# pkg install wayland seatd
```

После установки протокола и необходимых пакетов, композитор должен создать пользовательский интерфейс. В следующих разделах будут рассмотрены несколько композиторов. Все композиторы, использующие Wayland, требуют наличия runtime-каталога, определённого в окружении. Начиная с FreeBSD 14.1, он создаётся и определяется автоматически. Для более ранних версий это можно сделать с помощью следующей команды в оболочке bourne:

```
% export XDG_RUNTIME_DIR=/var/run/user/`id -u`
```

Важно отметить, что большинство композиторов ищут файлы конфигурации в каталоге `XDG_RUNTIME_DIR`. В приведённых здесь примерах будет использоваться параметр для указания файла конфигурации в ~/.config, чтобы разделить временные файлы и файлы конфигурации. Рекомендуется настроить псевдоним для каждого композитора, чтобы загружать указанный файл конфигурации.

Сообщается, что пользователи ZFS могут столкнуться с проблемами при работе с некоторыми клиентами Wayland, так как им требуется доступ к `posix_fallocate()` в runtime-каталоге. Хотя автор не смог воспроизвести эту проблему на своей системе с ZFS, рекомендуемым решением является отказ от использования ZFS для runtime-каталога и использование `tmpfs` для /var/run. В этом случае файловая система `tmpfs` используется для /var/run и монтируется командой `mount -t tmpfs tmpfs /var/run`, после чего это изменение можно сделать постоянным после перезагрузок через /etc/fstab. Переменная окружения XDG_RUNTIME_DIR может быть настроена на использование /var/run/user/$UID, чтобы избежать потенциальных проблем с ZFS. Учитывайте этот сценарий при изучении примеров конфигурации в следующих разделах.

Демон seatd помогает управлять доступом к общим системным устройствам для непривилегированных пользователей в композиторах; это включает графические карты. Для традиционных менеджеров X11, таких как Plasma и GNOME, `seatd` не требуется, но для обсуждаемых здесь композиторов Wayland он должен быть включён в системе и работать перед запуском окружения композитора. Чтобы включить и запустить демон `seatd` сейчас и при инициализации системы:

```
# sysrc seatd_enable="YES"
# service seatd start
```

После этого необходимо установить композитор, который аналогичен рабочему столу X11, для графической среды. Здесь рассматриваются три варианта, включая базовые настройки, настройку блокировки экрана и рекомендации для получения дополнительной информации.

6.3. Композитор Wayfire

Wayfire — это композитор, который стремится быть легковесным и настраиваемым. Доступно несколько функций, и он возвращает некоторые элементы из ранее выпущенного рабочего стола Compiz Fusion. Все части выглядят красиво на современном оборудовании. Чтобы запустить Wayfire, начните с установки необходимых пакетов:

```
# pkg install wayfire wf-shell alacritty swaylock-effects swayidle wlogout kanshi mako
wlsunset
```

Пакет `alacritty` предоставляет эмулятор терминала. Однако он не является строго обязательным, так как другие эмуляторы терминалов, такие как `kitty` и `Terminal` XFCE-4, были протестированы и подтверждены для работы под композитором Wayfire. Конфигурация Wayfire относительно проста; она использует файл, который следует изучить перед внесением любых изменений. Для начала скопируйте пример файла в каталог конфигурации среды выполнения, а затем отредактируйте файл:

```
% mkdir ~/.config/wayfire
% cp /usr/local/share/examples/wayfire/wayfire.ini ~/.config/wayfire
```

Настройки по умолчанию подойдут большинству пользователей. В файле конфигурации уже предустановлены параметры, такие как известный cube, а также приведены инструкции по доступным настройкам. Вот несколько основных параметров, на которые стоит обратить внимание:

```
[output]
mode = 1920x1080@60000
position = 0,0
transform = normal
scale = 1.000000
```

В этом примере из файла конфигурации вывод на экран должен соответствовать указанному режиму с указанной частотой. Например, параметр mode должен быть установлен как ширинахвысота@частота_обновления. Параметр position размещает вывод в указанной позиции (в пикселях). Значение по умолчанию подходит для большинства пользователей. Наконец, transform задаёт фоновое преобразование, а scale масштабирует вывод до указанного коэффициента. Значения по умолчанию для этих параметров обычно приемлемы; дополнительную информацию см. в документации.

Как уже упоминалось, Wayland — это новая технология, и не все приложения пока работают с этим протоколом. На данный момент sddm не поддерживает запуск и управление композиторами в Wayland. В этих примерах вместо него использовалась утилита swaylock. Файл конфигурации содержит параметры для запуска swayidle и swaylock для управления простоем и блокировкой экрана.

Эта опция для определения действия при простое системы указана как:

```
idle = swaylock
```

И таймаут блокировки настраивается с помощью следующих строк:

```
[idle]
toggle = <super> KEY_Z
screensaver_timeout = 300
dpms_timeout = 600
```

Первый вариант заблокирует экран через 300 секунд, а через следующие 300 секунд экран выключится с помощью опции dpms_timeout.

Последнее, на что стоит обратить внимание, — это клавиша <super>. В большинстве конфигураций упоминается эта клавиша, и она соответствует традиционной клавише Windows на клавиатуре. На большинстве клавиатур эта клавиша super доступна; однако, если

её нет, её следует переназначить в этом конфигурационном файле. Например, чтобы заблокировать экран, зажмите клавишу super, клавишу `shift` и нажмите клавишу `escape`. Если сопоставления не изменены, это запустит приложение swaylock. Конфигурация по умолчанию для `swaylock` отображает серый экран; однако приложение легко настраивается и хорошо документировано. Кроме того, поскольку была установлена версия swaylock-effects, доступно несколько дополнительных возможностей, таких как эффект размытия, который можно увидеть, используя следующую команду:

```
% swaylock --effect-blur 7x5
```

Также есть параметр `--clock`, который отображает часы с датой и временем на экране блокировки. При установке пакета x11/swaylock-effects включается конфигурация pam.d по умолчанию. Она предоставляет стандартные настройки, которые подойдут большинству пользователей. Доступны и более расширенные варианты; дополнительную информацию можно найти в документации PAM.

На этом этапе пришло время протестировать Wayfire и проверить, сможет ли он запуститься в системе. Просто введите следующую команду:

```
% wayfire -c ~/.config/wayfire/wayfire.ini
```

Теперь композитор должен запуститься и отобразить фоновое изображение вместе с панелью меню в верхней части экрана. Wayfire попытается перечислить установленные совместимые приложения для рабочего стола и отобразит их в этом выпадающем меню; например, если установлен файловый менеджер XFCE-4, он появится в этом меню. Если конкретное приложение совместимо и достаточно важно для назначения горячей клавиши, его можно привязать к комбинации клавиш с помощью конфигурационного файла wayfire.ini. Wayfire также имеет инструмент настройки под названием Wayfire Config Manager. Он доступен в выпадающем меню, но его также можно запустить через терминал, выполнив следующую команду:

```
% wcm
```

Различные параметры конфигурации Wayfire, включая специальные эффекты композитора, могут быть включены, отключены или настроены через это приложение. Кроме того, для более удобного взаимодействия, в конфигурационном файле могут быть активированы менеджер фона, панель и док-приложение:

```
panel = wf-panel
dock = wf-dock
background = wf-background
```

Изменения, внесённые через `wcm`, перезапишут пользовательские настройки в конфигурационном файле wayfire.ini. Настоятельно рекомендуется создать

> резервную копию файла wayfire.ini, чтобы можно было восстановить важные изменения.

Наконец, стандартный лаунчер, указанный в wayfire.ini, — это x11/wf-shell, который может быть заменён другими панелями по желанию пользователя.

6.4. Композитор Hikari

Композитор Hikari использует несколько концепций, ориентированных на продуктивность, такие как листы (sheets), рабочие пространства (workspaces) и другие. В этом отношении он напоминает тайловый оконный менеджер. Если разбирать подробнее, композитор начинается с одного рабочего пространства, которое аналогично виртуальным рабочим столам. Hikari использует одно рабочее пространство (или виртуальный рабочий стол) для взаимодействия с пользователем. Рабочее пространство состоит из нескольких представлений (views) — это рабочие окна в композиторе, сгруппированные либо в листы (sheets), либо в группы (groups). И листы, и группы состоят из набора представлений — то есть окон, объединённых вместе. При переключении между листами или группами активный лист или группа становятся рабочим пространством. В руководстве (man-странице) эти функции описаны более подробно, но в данном документе достаточно рассматривать одно рабочее пространство с одним листом. Установка Hikari включает один пакет x11-wm/hikari и терминальный эмулятор `alacritty`:

```
# pkg install hikari alacritty
```

> Другие оболочки, такие как `kitty` или `Terminal` в Plasma, будут работать под Wayland. Пользователям стоит поэкспериментировать с предпочитаемым терминальным редактором, чтобы проверить совместимость.

Hikari использует файл конфигурации hikari.conf, который может быть размещен либо в XDG_RUNTIME_DIR, либо указан при запуске с помощью параметра `-c`. Файл конфигурации автозапуска не обязателен, но может немного упростить переход на этот композитор. Начало настройки заключается в создании каталога конфигурации Hikari и копировании файла конфигурации для редактирования:

```
% mkdir ~/.config/hikari
% cp /usr/local/etc/hikari/hikari.conf ~/.config/hikari
```

Конфигурация разбита на различные разделы, такие как `ui`, `outputs`, `layouts` и другие. Для большинства пользователей значений по умолчанию будет достаточно, однако некоторые важные изменения всё же стоит внести. Например, переменная `$TERMINAL` обычно не установлена в окружении пользователя. Измените эту переменную или отредактируйте файл hikari.conf, указав:

```
terminal = "/usr/local/bin/alacritty"
```

Запустит терминал `alacritty` при нажатии связанной клавиши. При изучении конфигурационного файла следует обратить внимание, что заглавные буквы используются для сопоставления клавиш пользователю. Например, клавиша `L` для запуска терминала `L` + `Return` на самом деле является ранее упомянутой клавишей super или клавишей с логотипом Windows. Таким образом, удерживая `L/super/Windows` и нажимая `Enter`, вы откроете указанный эмулятор терминала с конфигурацией по умолчанию. Для сопоставления других клавиш приложениям необходимо создать определение действия. Для этого пункт действия должен быть указан в разделе actions, например:

```
actions {
  terminal = "/usr/local/bin/alacritty"
  browser = "/usr/local/bin/firefox"
}
```

Затем действие может быть отображено в разделе `keyboard`, который определен внутри раздела `bindings`:

```
bindings {
  keyboard {
SNIP
    "L+Return" = action-terminal
    "L+b" = action-browser
SNIP
```

После перезапуска Hikari удержание кнопки с логотипом Windows и нажатие клавиши `b` на клавиатуре запустит веб-браузер. Композитор не имеет строки меню, и рекомендуется настроить, как минимум, терминальный эмулятор перед миграцией. Руководство содержит множество документации, его следует прочитать перед полной миграцией. Ещё один положительный аспект Hikari заключается в том, что при переходе на этот композитор его можно запускать в средах рабочего стола Plasma и GNOME, что позволяет опробовать его перед полной миграцией.

Заблокировать экран в Hikari просто, так как пакет включает файл конфигурации pam.d и утилиту разблокировки. Сочетание клавиш для блокировки экрана — `L` (клавиша с логотипом Windows)+ `Shift` `Backspace`. Следует отметить, что все представления, не помеченные как публичные, будут скрыты. Эти представления не будут принимать ввод, когда экран заблокирован, но будьте осторожны с отображением конфиденциальной информации. Некоторым пользователям может быть проще перейти на другую утилиту блокировки экрана, например swaylock-effects, которая обсуждается в этом разделе. Чтобы запустить Hikari, используйте следующую команду:

```
% hikari -c ~/.config/hikari/hikari.conf
```

6.5. Композитор Sway

Композитор Sway — это тайловый композитор, предназначенный для замены оконного менеджера i3. Он должен работать с текущей конфигурацией i3 пользователя, однако для использования новых функций может потребоваться дополнительная настройка.

Перед установкой Sway убедитесь, что графическая карта (GPU) установлена и настроена правильно. Обратитесь к разделу Драйверы видеокарт для получения инструкций. Этот шаг необходим для корректной работы композитора Sway.

В следующих примерах предполагается новая установка без переноса каких-либо конфигураций i3. Для установки Sway и полезных компонентов выполните следующую команду от имени пользователя root:

```
# pkg install sway swayidle swaylock-effects alacritty dmenu-wayland dmenu
```

Для базового файла конфигурации выполните следующие команды, а затем отредактируйте файл конфигурации после его копирования:

```
% mkdir ~/.config/sway
% cp /usr/local/etc/sway/config ~/.config/sway
```

Базовый файл конфигурации содержит множество настроек по умолчанию, которые подойдут большинству пользователей. Однако следует внести несколько важных изменений, таких как:

```
# Logo key. Use Mod1 for Alt.
input * xkb_rules evdev
set $mod Mod4
# The preferred terminal emulator
set $term alacritty
set $lock swaylock -f -c 000000
output "My Workstation" mode 1366x768@60Hz position 1366 0
output * bg ~/wallpapers/mywallpaper.png stretch
### Idle configuration
exec swayidle -w \
          timeout 300 'swaylock -f -c 000000' \
          timeout 600 'swaymsg "output * dpms off"' resume 'swaymsg "output * dpms
on"' \
          before-sleep 'swaylock -f -c 000000'
```

В предыдущем примере загружаются правила `xkb` для событий evdev(4), а клавиша $mod устанавливается в значении логотипа Windows для привязки клавиш. Далее эмулятор терминала был установлен как `alacritty`, и определена команда блокировки экрана; подробнее об этом позже. Ключевое слово `output`, режим, позиция, фоновое изображение, а также Sway указано растягивать это изображение для заполнения экрана. Наконец, `swayidle`

настроен на работу в режиме демона и блокировку экрана после таймаута в 300 секунд, переводя экран или монитор в режим сна после 600 секунд. Здесь также определён цвет заблокированного фона 000000, что соответствует чёрному. С использованием swaylock-effects можно также отображать часы с параметром `--clock`. Дополнительные параметры см. на странице руководства. Также следует ознакомиться с sway-output(5); она содержит множество информации по настройке доступных параметров вывода.

В Sway, чтобы вызвать меню приложений, удерживайте клавишу с логотипом Windows (mod) и нажмите `d`. Меню можно перемещаться с помощью клавиш со стрелками на клавиатуре. Также есть способ изменить расположение панели и добавить трей; подробнее см. на sway-bar(5). В стандартной конфигурации в верхнем правом углу отображается дата и время. Пример можно найти в разделе `Bar` конфигурационного файла. По умолчанию конфигурация не включает блокировку экрана, за исключением приведённого выше примера, а также таймер блокировки. Для создания привязки клавиши блокировки добавьте следующую строку в раздел `Key bindings`:

```
# Lock the screen manually
bindsym $mod+Shift+Return exec $lock
```

Теперь экран можно заблокировать с помощью комбинации: удерживая клавишу с логотипом Windows, нажать и удерживать Shift, а затем нажать Enter. При установке Sway, будь то из пакета или коллекции портов FreeBSD, устанавливается файл по умолчанию для pam.d. Конфигурация по умолчанию подходит для большинства пользователей, но доступны и более продвинутые варианты. Для получения дополнительной информации ознакомьтесь с документацией PAM.

Наконец, чтобы выйти из Sway и вернуться к оболочке, удерживайте клавишу с логотипом Windows, клавишу Shift и нажмите `e`. Появится запрос с возможностью выхода из Sway. Во время миграции Sway можно запустить через эмулятор терминала в X11-окружении, например, Plasma. Это упрощает тестирование различных изменений и сочетаний клавиш перед полным переходом на этот композитор. Чтобы запустить Sway, выполните следующую команду:

```
% sway -c ~/.config/sway/config
```

6.6. Использование Xwayland

При установке Wayland бинарный файл `Xwayland` должен быть установлен, если только Wayland не собран без поддержки X11. Если файл /usr/local/bin/Xwayland отсутствует, установите его с помощью следующей команды:

```
# pkg install xwayland
```

Рекомендуется использовать разрабатываемую версию Xwayland, которая, скорее всего, была установлена вместе с пакетом Wayland. Каждый

композитор имеет свой способ включения или отключения этой функции.

После установки Xwayland настройте его в выбранном композиторе. Для Wayfire в файле wayfire.ini требуется следующая строка:

```
xwayland = true
```

Для композитора Sway Xwayland должен быть включен по умолчанию. Тем не менее, рекомендуется вручную добавить строку конфигурации в файл ~/.config/sway/config, например, следующую:

```
xwayland enable
```

Наконец, для Hikari не требуется никаких изменений. Поддержка Xwayland включена по умолчанию. Чтобы отключить эту поддержку, пересоберите пакет из коллекции портов и отключите поддержку Xwayland во время сборки.

После внесения этих изменений запустите композитор из командной строки и откройте терминал с помощью назначенных клавиш. В этом терминале выполните команду env и найдите переменные DISPLAY. Если композитор успешно запустил X-сервер Xwayland, эти переменные окружения должны выглядеть примерно следующим образом:

```
% env | grep DISPLAY
```

```
WAYLAND_DISPLAY=wayland-1
DISPLAY=:0
```

В этом выводе есть дисплей Wayland по умолчанию и дисплей, настроенный для сервера Xwayland. Другой способ проверить, что Xwayland работает корректно, — установить и протестировать небольшой package:[x11/eyes], а затем проверить вывод. Если приложение xeyes запускается и глаза следят за указателем мыши, значит, Xwayland работает правильно. Если же отображается ошибка, например, следующая, значит, что-то произошло во время инициализации Xwayland, и, возможно, его потребуется переустановить:

```
Error: Cannot open display wayland-0
```

Одной из особенностей безопасности Wayland является отсутствие дополнительного сетевого слушателя при отсутствии запущенного X-сервера. После включения Xwayland данная особенность безопасности перестает применяться в системе.

Для некоторых композиторов, таких как Wayfire, Xwayland может не запуститься правильно. В таком случае, env покажет следующую информацию о переменных окружения DISPLAY:

```
% env | grep DISPLAY
```

```
DISPLAY=wayland-1
WAYLAND_DISPLAY=wayland-1
```

Хотя `Xwayfire` был установлен и настроен, приложения X11 не запускаются, выдавая ошибку дисплея. Чтобы обойти эту проблему, убедитесь, что уже существует экземпляр `Xwayland`, использующий UNIX-сокет, с помощью двух методов. Сначала проверьте вывод команды `sockstat` и найдите X11-unix:

```
% sockstat | grep x11
```

Должна быть информация, аналогичная следующей:

```
trhodes  Xwayland   2734  8  stream /tmp/.X11-unix/X0
trhodes  Xwayland   2734  9  stream /tmp/.X11-unix/X0
trhodes  Xwayland   2734  10 stream /tmp/.X11-unix/X0
trhodes  Xwayland   2734  27 stream /tmp/.X11-unix/X0_
trhodes  Xwayland   2734  28 stream /tmp/.X11-unix/X0
```

Это указывает на наличие сокета X11. Это можно дополнительно проверить, попытавшись вручную запустить `Xwayland` в эмуляторе терминала, работающем под композитором:

```
% Xwayland
```

Если сокет X11 уже доступен, пользователю должно быть показано следующее сообщение об ошибке:

```
(EE)
Fatal server error:
(EE) Server is already active for display 0
    If this server is no longer running, remove /tmp/.X0-lock
    and start again.
(EE)
```

Поскольку доступен активный X-дисплей с использованием дисплея ноль, переменная окружения была просто задана неправильно. Чтобы исправить это, измените переменную окружения `DISPLAY` на `:0` и попробуйте запустить приложение снова. В следующем примере используется пакет mail/claws-mail в качестве приложения, которому требуется сервис `Xwayland`:

```
export DISPLAY=:0
```

После этого изменения приложение mail/claws-mail должно начать использовать Xwayland и работать как ожидается.

6.7. Удалённый рабочий стол с использованием VNC

Ранее в этом документе упоминалось, что Wayland не предоставляет такой же доступ в стиле X-сервера, как Xorg. Вместо этого пользователи могут выбрать любой протокол удалённого рабочего стола, например RDP или VNC. Коллекция портов FreeBSD включает wayvnc, который поддерживает композиторы на основе wlroots, такие как рассмотренные здесь. Это приложение можно установить с помощью:

```
# pkg install wayvnc
```

В отличие от некоторых других пакетов, wayvnc не поставляется с файлом конфигурации. К счастью, справочная страница содержит описание важных параметров, и их можно использовать для создания простого файла конфигурации:

```
address=0.0.0.0
enable_auth=true
username=username
password=password
private_key_file=/path/to/key.pem
certificate_file=/path/to/cert.pem
```

Необходимо сгенерировать ключевые файлы, и настоятельно рекомендуется их использовать для повышения безопасности соединения. При запуске wayvnc будет искать файл конфигурации в ~/.config/wayvnc/config. Это можно переопределить с помощью опции `-C файл_конфигурации` при запуске сервера. Таким образом, чтобы запустить сервер wayvnc, выполните следующую команду:

```
% wayvnc -C ~/.config/wayvnc/config
```

На момент написания этого документа нет rc.d-скрипта для запуска wayvnc при инициализации системы. Если требуется такая функциональность, необходимо создать локальный стартовый файл. Вероятно, это следует оформить как запрос на добавление функции для сопровождающего порта.

6.8. Логин менеджер Wayland

Хотя существует несколько менеджеров входа в систему, которые постепенно переходят на Wayland, одним из вариантов является текстовый менеджер входа x11/ly. Требуя минимальной настройки, ly запускает Sway, Wayfire и другие окружения, отображая окно входа при инициализации системы. Для установки ly выполните следующую команду:

```
# pkg install ly
```

Будут представлены некоторые подсказки по настройке, основные шаги - добавить следующие строки в /etc/gettytab:

```
Ly:\
  :lo=/usr/local/bin/ly:\
  :al=root:
```

И затем измените строку ttyv1 в файле /etc/ttys, чтобы она соответствовала следующей строке:

```
ttyv1 "/usr/libexec/getty Ly" xterm onifexists secure
```

После перезагрузки системы должно появиться приглашение на вход. Для настройки конкретных параметров, таких как язык, отредактируйте файл /usr/local/etc/ly/config.ini. Как минимум, в этом файле должен быть указан tty, который был ранее задан в /etc/ttys.

Если для терминала ttyv0 настроен вход в систему, может потребоваться нажать клавиши `alt` и `F1`, чтобы корректно отображалось окно входа.

Когда появляется окно входа, использование стрелок влево и вправо позволяет переключаться между различными поддерживаемыми оконными менеджерами.

6.9. Полезные утилиты

Один полезный инструмент для Wayland, который могут использовать все композиторы, — это `waybar`. Хотя Wayfire и поставляется с меню запуска, удобная и быстрая панель задач — это полезное дополнение для любого композитора или менеджера рабочего стола. `waybar` — это совместимая с Wayland панель задач, которая быстро работает и легко настраивается. Чтобы установить пакет и вспомогательную утилиту управления аудио, выполните следующую команду:

```
# pkg install pavucontrol waybar
```

Для создания каталога конфигурации и копирования стандартного файла конфигурации выполните следующие команды:

```
% mkdir ~/.config/waybar
% cp /usr/local/etc/xdg/waybar/config ~/.config/waybar
```

Утилита `lavalauncher` предоставляет панель запуска для различных приложений. Вместе с пакетом не поставляется пример файла конфигурации, поэтому необходимо выполнить

следующие действия:

```
mkdir ~/.config/lavalauncher
```

Пример файла конфигурации, который включает только Firefox и расположен справа, приведён ниже:

```
global-settings {
    watch-config-file = true;
}

bar {
    output            = eDP-1;
    position          = bottom;
    background-colour = "#202020";

    # Condition for the default configuration set.
    condition-resolution = wider-than-high;

    config {
        position = right;
    }

    button {
        image-path          =
/usr/local/lib/firefox/browser/chrome/icons/default/default48.png;
        command[mouse-left] =     /usr/local/bin/firefox;
    }
    button {
      image-path           =   /usr/local/share/pixmaps/thunderbird.png;
      command[mouse-left]  =   /usr/local/bin/thunderbird;
}
```

Глава 7. Сеть

7.1. Обзор

Эта глава углубляется в тему настройки сети и производительности, демонстрируя мощные сетевые возможности операционной системы FreeBSD. Независимо от работы с проводными или беспроводными сетями, эта глава предоставляет всеобъемлющее руководство по настройке и оптимизации сетевого подключения в FreeBSD.

Прежде чем углубляться в детали, читателям будет полезно иметь базовое понимание сетевых концепций, таких как протоколы, сетевые интерфейсы и адресация.

Эта глава охватывает:

- Возможности настройки проводных сетей в FreeBSD, включая настройку сетевых интерфейсов, адресацию и параметры конфигурации.
- Навыки, необходимые для настройки беспроводных сетей в FreeBSD, включая настройку беспроводных сетевых интерфейсов, протоколы безопасности и методы устранения неполадок.
- Возможности сетевых подключений FreeBSD и её репутация в области превосходной сетевой производительности.
- Понимание различных сетевых служб и протоколов, поддерживаемых FreeBSD, с инструкциями по настройке DNS, DHCP и других.

Дополнительная информация о настройке расширенных конфигураций сети приведена в разделе Сложные вопросы работы в сети.

7.2. Настройка сети

Настройка проводного или беспроводного подключения — распространённая задача для пользователя FreeBSD. В этом разделе показано, как определить проводные и беспроводные сетевые адаптеры и как их настроить.

Прежде чем приступить к настройке, необходимо знать следующие сетевые данные:

- Есть ли в сети DHCP
- Если в сети нет DHCP, используемый статический IP-адрес
- Маска сети
- IP-адрес шлюза по умолчанию

Сетевое соединение могло быть настроено во время установки с помощью bsdinstall(8).

7.2.1. Определение сетевых адаптеров

FreeBSD поддерживает широкий спектр сетевых адаптеров как для проводных, так и для беспроводных сетей. Проверьте список совместимости оборудования для используемого выпуска FreeBSD, чтобы убедиться, что сетевой адаптер поддерживается.

Чтобы получить список сетевых адаптеров, используемых в системе, выполните следующую команду:

```
% pciconf -lv | grep -A1 -B3 network
```

Вывод должен быть похож на следующий:

```
em0@pci0:0:25:0:        class=0x020000 rev=0x03 hdr=0x00 vendor=0x8086 device=0x10f5
subvendor=0x17aa subdevice=0x20ee
    vendor     = 'Intel Corporation' ①
    device     = '82567LM Gigabit Network Connection' ②
    class      = network
    subclass   = ethernet
--
iwm0@pci0:3:0:0:        class=0x028000 rev=0x00 hdr=0x00 vendor=0x8086 device=0x4237
subvendor=0x8086 subdevice=0x1211
    vendor     = 'Intel Corporation' ①
    device     = 'PRO/Wireless 5100 AGN [Shiloh] Network Connection' ②
    class      = network
```

Текст перед символом '@' — это название драйвера, управляющего устройством. В данном случае это em(4) и iwn(4).

① Показывает название производителя

② Показывает название устройства

Требуется загружать модуль сетевой карты только в том случае, если FreeBSD не определил его правильно.

Например, для загрузки модуля alc(4) выполните следующую команду:

```
# kldload if_alc
```

В качестве альтернативы, для загрузки драйвера как модуля во время загрузки, добавьте следующую строку в /boot/loader.conf:

```
if_alc_load="YES"
```

7.3. Проводные сети

После загрузки нужного драйвера необходимо настроить сетевой адаптер. В FreeBSD для именования сетевого интерфейса используется имя драйвера, за которым следует номер устройства. Номер устройства указывает порядок обнаружения адаптера при загрузке или позднее.

Например, `em0` — это первая сетевая карта (NIC) в системе, использующая драйвер em(4).

Для отображения конфигурации сетевого интерфейса введите следующую команду:

```
% ifconfig
```

Вывод должен быть похож на следующий:

```
em0: flags=8863<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500

options=481249b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM,LRO,WOL_MAGIC,VLAN_H
WFILTER,NOMAP>
        ether 00:1f:16:0f:27:5a
        inet6 fe80::21f:16ff:fe0f:275a%em0 prefixlen 64 scopeid 0x1
        inet 192.168.1.19 netmask 0xffffff00 broadcast 192.168.1.255
        media: Ethernet autoselect (1000baseT <full-duplex>)
        status: active
        nd6 options=23<PERFORMNUD,ACCEPT_RTADV,AUTO_LINKLOCAL>
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> metric 0 mtu 16384
        options=680003<RXCSUM,TXCSUM,LINKSTATE,RXCSUM_IPV6,TXCSUM_IPV6>
        inet6 ::1 prefixlen 128
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x2
        inet 127.0.0.1 netmask 0xff000000
        groups: lo
        nd6 options=21<PERFORMNUD,AUTO_LINKLOCAL>
```

В этом примере были отображены следующие устройства:

- `em0`: Интерфейс Ethernet.
- `lo0`: Интерфейс loop представляет собой механизм программной петли (loopback), который может использоваться для анализа производительности, тестирования программного обеспечения и/или локального взаимодействия. Подробнее см. в lo(4).

Пример показывает, что `em0` работает и активен.

Ключевые показатели:

1. `UP` означает, что интерфейс настроен и готов к работе.
2. Интерфейс имеет IPv4 адрес в Интернете (`inet`) — `192.168.1.19`.
3. Интерфейс имеет IPv6-адрес (`inet6`) в Интернете: `fe80::21f:16ff:fe0f:275a%em0`.

4. У него есть действительная маска подсети (`netmask`), где `0xffffff00` эквивалентно `255.255.255.0`.
5. У него есть корректный широковещательный адрес, `192.168.1.255`.
6. MAC-адрес интерфейса (`ether`) — `00:1f:16:0f:27:5a`.
7. Выбор физической среды находится в режиме автоопределения (`media: Ethernet autoselect (1000baseT <full-duplex>)`).
8. Состояние соединения (`status`) имеет значение `active`, что указывает на обнаружение несущего сигнала. Для `em0` состояние `status: no carrier` является нормальным, если кабель Ethernet не подключен к интерфейсу.

Если вывод ifconfig(8) показал что-то похожее на следующий результат, это означало бы, что интерфейс не настроен:

```
em0: flags=8822<BROADCAST,SIMPLEX,MULTICAST> metric 0 mtu 1500

options=481249b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM,LRO,WOL_MAGIC,VLAN_H
WFILTER,NOMAP>
        ether 00:1f:16:0f:27:5a
        media: Ethernet autoselect
        status: no carrier
        nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
```

7.3.1. Настройка статического IPv4-адреса

В этом разделе представлено руководство по настройке статического IPv4-адреса в системе FreeBSD.

Настройку сетевой интерфейсной карты можно выполнить из командной строки с помощью ifconfig(8), но она не сохранится после перезагрузки, если конфигурация также не добавлена в /etc/rc.conf.

Если сеть была настроена во время установки с помощью bsdinstall(8), некоторые записи для сетевой интерфейсной карты (NIC) могут уже присутствовать. Проверьте /etc/rc.conf перед выполнением sysrc(8).

IP-адрес можно задать, выполнив следующую команду:

```
# ifconfig em0 inet 192.168.1.150/24
```

Чтобы изменение сохранялось после перезагрузки, выполните следующую команду:

```
# sysrc ifconfig_em0="inet 192.168.1.150 netmask 255.255.255.0"
```

Добавьте маршрут по умолчанию, выполнив следующую команду:

```
# sysrc defaultrouter="192.168.1.1"
```

Добавьте DNS-записи в /etc/resolv.conf:

```
nameserver 8.8.8.8
nameserver 8.8.4.4
```

Затем перезапустите `netif` и `routing`, выполнив следующую команду:

```
# service netif restart && service routing restart
```

Соединение можно проверить с помощью ping(8):

```
% ping -c2 www.FreeBSD.org
```

Вывод должен быть похож на следующий:

```
PING web.geo.FreeBSD.org (147.28.184.45): 56 data bytes
64 bytes from 147.28.184.45: icmp_seq=0 ttl=51 time=55.173 ms
64 bytes from 147.28.184.45: icmp_seq=1 ttl=51 time=53.093 ms

--- web.geo.FreeBSD.org ping statistics ---
2 packets transmitted, 2 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 53.093/54.133/55.173/1.040 ms
```

7.3.2. Настройка динамического IPv4-адреса

Если в сети есть DHCP-сервер, то настроить сетевой интерфейс для использования DHCP очень просто. FreeBSD использует dhclient(8) в качестве DHCP-клиента. dhclient(8) автоматически предоставит IP-адрес, маску сети и маршрутизатор по умолчанию.

Чтобы настроить интерфейс для работы с DHCP, выполните следующую команду:

```
# sysrc ifconfig_em0="DHCP"
```

dhclient(8) можно использовать вручную, выполнив следующую команду:

```
# dhclient em0
```

Вывод должен быть похож на следующий:

```
DHCPREQUEST on em0 to 255.255.255.255 port 67
```

```
DHCPACK from 192.168.1.1
unknown dhcp option value 0x7d
bound to 192.168.1.19 -- renewal in 43200 seconds.
```

Таким образом можно убедиться, что назначение адреса с помощью DHCP работает корректно.

Клиент dhclient(8) может быть запущен в фоновом режиме. Это может вызвать проблемы с приложениями, зависящими от рабочей сети, но во многих случаях обеспечит более быстрый запуск.

Для выполнения dhclient(8) в фоновом режиме выполните следующую команду:

```
# sysrc background_dhclient="YES"
```

Затем перезапустите `netif`, выполнив следующую команду:

```
# service netif restart
```

Соединение можно проверить с помощью ping(8):

```
% ping -c2 www.FreeBSD.org
```

Вывод должен быть похож на следующий:

```
PING web.geo.FreeBSD.org (147.28.184.45): 56 data bytes
64 bytes from 147.28.184.45: icmp_seq=0 ttl=51 time=55.173 ms
64 bytes from 147.28.184.45: icmp_seq=1 ttl=51 time=53.093 ms

--- web.geo.FreeBSD.org ping statistics ---
2 packets transmitted, 2 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 53.093/54.133/55.173/1.040 ms
```

7.3.3. IPv6

IPv6 — это новая версия широко известного протокола IP, также известного как IPv4.

IPv6 предоставляет несколько преимуществ по сравнению с IPv4, а также множество новых функций:

- Его 128-битное адресное пространство позволяет использовать 340 282 366 920 938 463 463 374 607 431 768 211 456 адресов. Это решает проблему нехватки IPv4-адресов и неизбежного исчерпания IPv4-адресного пространства.

- Маршрутизаторы хранят в своих таблицах маршрутизации только агрегированные сетевые адреса, что сокращает средний размер таблицы маршрутизации до 8192 записей. Это решает проблемы масштабируемости, связанные с IPv4, где каждый выделенный блок IPv4-адресов должен был обмениваться между интернет-маршрутизаторами, что приводило к чрезмерному увеличению их таблиц маршрутизации и затрудняло эффективную маршрутизацию.
- Автоконфигурация адресов (RFC4862).
- Обязательные multicast-адреса.
- Встроенный IPsec (безопасность IP).
- Упрощённая структура заголовка.
- Поддержка мобильного IP.
- Механизмы перехода с IPv6 на IPv4.

FreeBSD включает в себя эталонную реализацию IPv6 от проекта KAME и поставляется со всем необходимым для работы с IPv6.

Этот раздел посвящён настройке и запуску IPv6.

Существует три различных типа IPv6-адресов:

Unicast

Пакет, отправленный на одноадресный (unicast) адрес, поступает на интерфейс, принадлежащий этому адресу.

Anycast

Эти адреса синтаксически неотличимы от одноадресных, но они обозначают группу интерфейсов. Пакет, предназначенный для адреса anycast, будет доставлен на ближайший интерфейс.

Multicast

Эти адреса идентифицируют группу интерфейсов. Пакет, предназначенный для multicast-адреса, будет доставлен на все интерфейсы, входящие в multicast-группу. IPv4 широковещательный адрес, обычно `xxx.xxx.xxx.255`, в IPv6 выражается через multicast-адреса.

При чтении IPv6-адреса каноническая форма представляется как `x:x:x:x:x:x:x:x`, где каждый `x` соответствует 16-битному шестнадцатеричному значению. Пример: `FEBC:A574:382B:23C1:AA49:4592:4EFE:9982`.

Часто в адресе встречаются длинные последовательности нулей. Для замены одной такой последовательности в адресе можно использовать `::` (двойное двоеточие). Кроме того, в каждом шестнадцатеричном числе можно опустить до трёх ведущих нулей. Например, `fe80::1` соответствует канонической форме `fe80:0000:0000:0000:0000:0000:0000:0001`.

Третья форма записи — указание последних 32 битов в известной IPv4 нотации. Например, `2002::10.0.0.1` соответствует шестнадцатеричной канонической записи `2002:0000:0000:0000:0000:0000:0a00:0001`, которая, в свою очередь, эквивалентна `2002::a00:1`.

Чтобы просмотреть IPv6-адрес системы FreeBSD, выполните следующую команду:

```
# ifconfig
```

Вывод должен быть похож на следующий:

```
em0: flags=8863<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500

options=481249b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM,LRO,WOL_MAGIC,VLAN_H
WFILTER,NOMAP>
        ether 00:1f:16:0f:27:5a
        inet 192.168.1.150 netmask 0xffffff00 broadcast 192.168.1.255
        inet6 fe80::21f:16ff:fe0f:275a%em0 prefixlen 64 scopeid 0x1
        media: Ethernet autoselect (1000baseT <full-duplex>)
        status: active
        nd6 options=23<PERFORMNUD,ACCEPT_RTADV,AUTO_LINKLOCAL>
```

В этом примере интерфейс `em0` использует `fe80::21f:16ff:fe0f:275a%em0` — автоматически сконфигурированный link-local адрес, который был автоматически сгенерирован из MAC-адреса.

Некоторые IPv6-адреса зарезервированы. Список зарезервированных адресов можно посмотреть в следующей таблице:

Таблица 10. Пример зарезервированных адресов IPv6

IPv6 address	Описание	Заметки
`::/128`	не указано	Эквивалентно `0.0.0.0` в IPv4.
`::1/128`	loopback-адрес (адрес обратной петли)	Эквивалент `127.0.0.1` в IPv4.
`::ffff:0.0.0.0/96`	IPv4-отображенный IPv6-адрес	Младшие 32 бита представляют IPv4-адрес для совместимости с хостами и маршрутизаторами, поддерживающих только IPv4.
`fe80::/10`	link-local unicast	Эквивалентно 169.254.0.0/16 в IPv4.
`fc00::/7`	unique-local	Уникальные локальные адреса предназначены для локального взаимодействия и доступны для маршрутизации только в пределах группы взаимодействующих сайтов.

IPv6 address	Описание	Заметки
ff00::/8	multicast	
2000::/3	global unicast	Все глобальные одноадресные адреса выделяются из этого пула. Первые 3 бита имеют значение 001.
2001:db8::/32, 3fff::/20	documentation	Адресный префикс IPv6 для использования в документации.

Для получения дополнительной информации о структуре IPv6-адресов обратитесь к RFC4291.

7.3.4. Настройка статического IPv6-адреса

Для настройки системы FreeBSD в качестве IPv6-клиента со статическим IPv6-адресом необходимо установить IPv6-адрес.

Выполните следующие команды для выполнения требований:

```
# sysrc ifconfig_em0_ipv6="inet6 2001:db8:4672:6565:2026:5043:2d42:5344 prefixlen 64"
```

Чтобы назначить маршрутизатор по умолчанию, укажите его адрес, выполнив следующую команду:

```
# sysrc ipv6_defaultrouter="2001:db8:4672:6565::1"
```

Для настройки дополнительного IPv6 anycast-адреса укажите адрес anycast как `_aliasN`(указано в rc.conf(5)) с добавлением опции `anycast`:

```
# sysrc ifconfig_em0_alias0="inet6 2001:db8:4672:6565::a anycast"
```

Имейте в виду, что приложения не могут привязываться к anycast-адресам; в этом случае необходимо использовать алиас-адрес.

7.3.5. Настройка динамического адреса IPv6

Для динамической настройки IPv6-адреса интерфейса с использованием SLAAC, выполните следующие команды:

```
# sysrc ifconfig_em0_ipv6="inet6 accept_rtadv"
# sysrc rtsold_enable="YES"
```

Обратите внимание, что при включённой IPv6-маршрутизации (т.е. `ipv6_gateway_enable=YES`), система не будет настраивать SLAAC-адрес, если переменная sysctl(8) `net.inet6.ip6.rfc6204w3` не установлена в 1.

7.3.6. Объявление маршрутизатора и автонастройка хоста

Этот раздел демонстрирует, как настроить rtadvd(8) на IPv6-маршрутизаторе для объявления префикса IPv6-сети и маршрута по умолчанию.

Чтобы включить rtadvd(8), выполните следующую команду:

```
# sysrc rtadvd_enable="YES"
```

Важно указать интерфейс, на котором будет выполняться IPv6-анонсирование маршрутизатора. Например, чтобы указать rtadvd(8) использовать `em0`:

```
# sysrc rtadvd_interfaces="em0"
```

Далее создайте файл конфигурации /etc/rtadvd.conf, как показано в этом примере:

```
em0:\
    :addrs#1:addr="2001:db8:1f11:246::":prefixlen#64:tc=ether:
```

Замените `em0` на используемый интерфейс и `2001:db8:1f11:246::` на префикс выделенного диапазона.

Для выделенной подсети `/64` больше ничего менять не нужно. В противном случае измените значение `prefixlen#` на соответствующее.

7.3.7. Соответствие адресов IPv6 и IPv4

Когда на сервере включен IPv6, может возникнуть необходимость разрешить использование IPv4-отображенных IPv6-адресов. Эта опция совместимости позволяет представлять IPv4-адреса в виде IPv6-адресов. Разрешение IPv6-приложениям взаимодействовать с IPv4 и наоборот может представлять угрозу безопасности.

Эта опция может не требоваться в большинстве случаев и доступна только для совместимости. Она позволит IPv6-приложениям работать с IPv4 в двухстековой среде. Это особенно полезно для сторонних приложений, которые могут не поддерживать IPv6-окружение.

Для включения этой функции выполните следующую команду:

```
# sysrc ipv6_ipv4mapping="YES"
```

7.4. Беспроводные сети

Большинство беспроводных сетей основаны на стандартах IEEE® 802.11.

FreeBSD поддерживает сети, работающие по стандартам 802.11a, 802.11b, 802.11g и 802.11n.

802.11ac поддержка в FreeBSD в настоящее время находится в стадии разработки.

Базовая беспроводная сеть состоит из нескольких станций, взаимодействующих через радиомодули, работающие в диапазоне 2,4 ГГц или 5 ГГц, хотя это может различаться в зависимости от региона и также меняется для обеспечения связи в диапазонах 2,3 ГГц и 4,9 ГГц.

Настройка беспроводной сети включает три основных этапа:

1. Сканировать и выбрать точку доступа
2. Аутентификация станции
3. Настройте IP-адрес или используйте DHCP.

Следующие разделы описывают каждый шаг.

7.4.1. Быстрое начало работы: подключение к беспроводной сети

Подключение FreeBSD к существующей беспроводной сети — весьма распространённая ситуация.

Эта процедура быстрого старта показывает шаги, необходимые для подключения к сети с использованием базовой аутентификации. Более подробная процедура приведена в следующем разделе.

Первый шаг — получить Service Set Identifier (SSID) и Pre-Shared Key (PSK) для беспроводной сети у администратора сети.

Второй шаг — добавить запись для этой сети в файл /etc/wpa_supplicant.conf.

Если файл не существует, создайте его. Подробности о форматировании записей в этом файле см. в wpa_supplicant.conf(5).

```
ctrl_interface=/var/run/wpa_supplicant
eapol_version=1
ap_scan=1
fast_reauth=1

network={
 ssid="myssid" ①
 psk="mypsk" ②
}
```

① Это SSID беспроводной сети. Замените его на имя беспроводной сети.

② Это PSK беспроводной сети. Замените его на пароль от беспроводной сети.

Третий шаг — добавить запись о сетевом интерфейсе для настройки сети при загрузке. Получите имя сетевого интерфейса с помощью sysctl `net.wlan.devices`. В приведённом ниже примере вывод этой команды показывает, что сетевым интерфейсом является "iwm0".

```
# sysctl net.wlan.devices
```

Вывод должен быть похож на следующий:

```
net.wlan.devices: iwm0
```

В следующей строке с командой sysrc замените "iwm0" на вывод команды sysctl, если это необходимо.

```
# sysrc wlans_iwm0="wlan0"
# sysrc ifconfig_wlan0="WPA DHCP"
```

- И последний шаг — перезапуск службы `netif` выполнением следующей команды:

```
# service netif restart
```

7.4.2. Базовая настройка беспроводной сети

Эта секция предоставляет детальный пример настройки беспроводной сетевой карты. Чтобы узнать, какие беспроводные сетевые карты есть в системе, ознакомьтесь с разделом Определение сетевых адаптеров.

```
# ifconfig wlan0 create wlandev iwm0
```

Чтобы изменение сохранялось после перезагрузки, выполните следующую команду:

```
# sysrc wlans_iwm0="wlan0"
```

Поскольку нормативные требования различаются в разных частях мира, необходимо правильно настроить домены, соответствующие вашему местоположению, чтобы получить корректную информацию о доступных каналах.

Доступные определения регионов можно найти в /etc/regdomain.xml. Для установки данных во время выполнения используйте `ifconfig`:

```
# ifconfig wlan0 regdomain etsi2 country AT
```

Чтобы сохранить настройки, добавьте их в /etc/rc.conf:

```
# sysrc create_args_wlan0="country AT regdomain etsi2"
```

7.4.3. Сканирование беспроводных сетей

Доступные беспроводные сети можно просканировать с помощью ifconfig(8).

Для вывода списка беспроводных сетей выполните следующую команду:

```
# ifconfig wlan0 up list scan
```

Вывод должен быть похож на следующий:

```
SSID/MESH ID                      BSSID              CHAN RATE    S:N     INT CAPS
FreeBSD                           e8:d1:1b:1b:58:ae    1   54M  -47:-96   100 EP   RSN
BSSLOAD HTCAP WPS WME
NetBSD                            d4:b9:2f:35:fe:08    1   54M  -80:-96   100 EP   RSN
BSSLOAD HTCAP WPS WME
OpenBSD                           fc:40:09:c6:31:bd   36   54M  -94:-96   100 EPS
VHTPWRENV APCHANREP RSN WPS BSSLOAD HTCAP VHTCAP VHTOPMODE WME
GNU-Linux                         dc:f8:b9:a0:a8:e0   44   54M  -95:-96   100 EP   WPA
RSN WPS HTCAP VHTCAP VHTOPMODE WME VHTPWRENV
Windows                           44:48:b9:b3:c3:ff   44   54M  -84:-96   100 EP
BSSLOAD VHTPWRENV HTCAP WME RSN VHTCAP VHTOPMODE WPS
MacOS                             46:48:b9:b3:c3:ff   44   54M  -84:-96   100 EP
BSSLOAD VHTPWRENV HTCAP WME RSN VHTCAP VHTOPMODE WPS
```

1. SSID/MESH ID идентифицирует название сети.
2. BSSID идентифицирует MAC-адрес точки доступа.
3. Поле `CAPS` определяет тип каждой сети и возможности работающих в ней станций (подробности см. в определении `list scan` в ifconfig(8)).

7.4.4. Подключение и аутентификация в беспроводной сети

После выбора беспроводной сети из списка обнаруженных необходимо выполнить подключение и аутентификацию. В подавляющем большинстве беспроводных сетей аутентификация осуществляется с помощью пароля, настроенного в маршрутизаторе. Другие схемы требуют выполнения криптографического рукопожатия перед началом передачи данных, используя либо предварительно распределённые ключи или секреты, либо более сложные схемы, включающие серверные службы, такие как RADIUS.

Аутентификация с WPA2/WPA/Personal

Процесс аутентификации в беспроводной сети управляется с помощью wpa_supplicant(8).

Конфигурация wpa_supplicant(8) задаётся в файле /etc/wpa_supplicant.conf. Подробнее см. wpa_supplicant.conf(5).

После завершения сканирования беспроводных сетей, выбора сети и получения пароля (PSK), эта информация будет добавлена в файл /etc/wpa_supplicant.conf, как показано в следующем примере:

```
network={
        scan_ssid=1 ①
        ssid="FreeBSD" ②
        psk="12345678" ③
}
```

① Метод сканирования SSID. Используйте эту опцию только если сеть скрыта.

② Имя сети.

③ Пароль беспроводной сети.

Следующим шагом будет настройка беспроводного соединения в файле /etc/rc.conf.

Для использования статического адреса необходимо выполнить следующую команду:

```
# sysrc ifconfig_wlan0="inet 192.168.1.20 netmask 255.255.255.0"
```

Для использования динамического адреса необходимо выполнить следующую команду:

```
# sysrc ifconfig_wlan0="WPA DHCP"
```

Затем перезапустите сеть, выполнив следующую команду:

```
# service netif restart
```

Дополнительная информация о более сложных методах аутентификации доступна в Расширенная аутентификация беспроводных сетей.

Аутентификация в открытых сетях

Важно, чтобы пользователь был **очень** осторожен при подключении к открытым сетям без какой-либо аутентификации.

После завершения сканирования беспроводной сети и выбора SSID выполните следующую команду:

```
# ifconfig wlan0 ssid SSID
```

Затем выполните dhclient(8), чтобы настроить адрес:

```
# dhclient wlan0
```

7.4.5. Использование проводного и беспроводного подключений одновременно

Проводное соединение обеспечивает лучшую производительность и надёжность, тогда как беспроводное соединение предоставляет гибкость и мобильность. Пользователи ноутбуков, как правило, хотят свободно переключаться между этими двумя типами соединений.

На FreeBSD можно объединить два или более сетевых интерфейса в режиме "отказоустойчивости". Такой тип конфигурации использует наиболее предпочтительное и доступное соединение из группы сетевых интерфейсов, а операционная система автоматически переключается при изменении состояния канала.

Агрегация и отказоустойчивость каналов описаны в Агрегация и отказоустойчивость каналов, а пример использования как проводных, так и беспроводных соединений приведён в Режим отказоустойчивости между проводными и беспроводными интерфейсами.

7.5. Имя сайта

Имя хоста представляет собой полностью определённое доменное имя (FQDN) хоста в сети.

Если для хоста не задано имя, FreeBSD назовет себя `Amnesiac`.

7.5.1. Проверить текущее имя хоста

hostname(1) может использоваться для проверки текущего имени хоста:

```
$ hostname
```

Вывод должен быть похож на следующий:

```
freebsdhostname.example.com
```

7.5.2. Изменить имя хоста

Для изменения имени хоста с сохранением после перезагрузки выполните следующую команду:

```
# sysrc hostname="freebsdhostname.example.com"
```

7.6. DNS

DNS можно представить как телефонный справочник, в котором IP-адрес сопоставляется с именем хоста и наоборот.

Существует три файла, которые управляют взаимодействием системы FreeBSD с DNS. Эти три файла: hosts(5), resolv.conf(5) и nsswitch.conf(5)

Если иное не указано в файле /etc/nsswitch.conf, FreeBSD сначала проверит адреса в файле /etc/hosts, а затем DNS-информацию в файле /etc/resolv.conf.

Файл nsswitch.conf(5) определяет, как должен работать nsdispatch (диспетчер переключения службы имен).

По умолчанию раздел hosts в файле /etc/nsswitch.conf будет выглядеть следующим образом:

```
hosts: files dns
```

Например, в случае использования службы nscd(8). Порядок предпочтения можно изменить, оставив строку следующей:

```
hosts: files cache dns
```

7.6.1. Локальные адреса

Файл /etc/hosts представляет собой простую текстовую базу данных, которая сопоставляет имена хостов с IP-адресами. В него можно добавить записи для локальных компьютеров, подключённых через LAN, чтобы использовать простые имена вместо настройки DNS-сервера. Кроме того, /etc/hosts может использоваться для хранения локальных записей интернет-имен, что уменьшает необходимость запрашивать внешние DNS-серверы для часто используемых имен.

Например, если в локальной среде имеется локальный экземпляр www/gitlab-ce, его можно добавить следующим образом в файл /etc/hosts:

```
192.168.1.150 git.example.com git
```

7.6.2. Настройка сервера имен

Как система FreeBSD обращается к системе доменных имён (DNS) в Интернете, управляется с помощью resolv.conf(5).

Наиболее распространённые записи в /etc/resolv.conf:

`nameserver`	IP-адрес сервера имен, к которому должен обращаться резолвер. Серверы опрашиваются в указанном порядке, максимум три.
`search`	Список поиска для разрешения имён хостов. Обычно определяется доменом локального имени хоста.
`domain`	Локальное доменное имя.

Типичный файл /etc/resolv.conf выглядит следующим образом:

```
search example.com
nameserver 147.11.1.11
nameserver 147.11.100.30
```

Должна использоваться только одна из опций — `search` или `domain`.

При использовании DHCP dhclient(8) обычно перезаписывает /etc/resolv.conf информацией, полученной от DHCP-сервера.

Если машина, в которой производится настройка, **не** является DNS-сервером, для улучшения производительности DNS-запросов можно использовать local-unbound(8).

Чтобы включить его при загрузке, выполните следующую команду:

```
# sysrc local_unbound_enable="YES"
```

Чтобы запустить службу local-unbound(8), выполните следующую команду:

```
# service local_unbound start
```

7.7. Устранение неполадок

При устранении неполадок в конфигурации оборудования и программного обеспечения сначала проверяйте самые простые вещи.

- Вставлен ли сетевой кабель?
- Правильно ли настроены сетевые службы?
- Настроен ли межсетевой экран правильно?
- Поддерживается ли сетевая карта (NIC) в FreeBSD?

- Работает ли маршрутизатор правильно?

Прежде чем отправить отчёт об ошибке, всегда проверяйте раздел "Аппаратные требования" на странице релиза FreeBSD, обновите версию FreeBSD до последней STABLE версии, проверьте архивы списков рассылки и выполните поиск в Интернете.

7.7.1. Устранение неполадок в проводных сетях

Если карта работает, но производительность низкая, ознакомьтесь с tuning(7). Также проверьте настройки сети, так как неправильные параметры могут быть причиной медленного соединения.

Сообщения `No route to host` возникают, если система не может найти маршрут для пакета до целевого хоста. Это может произойти, если не указан маршрут по умолчанию или если кабель отключен. Проверьте вывод команды `netstat -rn` и убедитесь, что существует действительный маршрут до хоста. Если его нет, ознакомьтесь с разделом Шлюзы и маршруты.

Сообщения об ошибках `ping: sendto: Permission denied` часто возникают из-за неправильно настроенного межсетевого экрана. Если на FreeBSD включен межсетевой экран, но не определены правила, политика по умолчанию — запрещать весь трафик, включая ping(8). Дополнительную информацию можно найти в Межсетевые экраны.

7.7.2. Устранение неполадок в беспроводных сетях

В этом разделе описаны шаги для устранения распространённых проблем с беспроводными сетями.

- Если точка доступа не отображается при сканировании, проверьте, что конфигурация не ограничивает беспроводное устройство определённым набором каналов.
- Если устройство не может подключиться к точке доступа, убедитесь, что конфигурация соответствует настройкам на точке доступа. Это включает схему аутентификации и все протоколы безопасности. Упростите конфигурацию насколько это возможно. Если используется протокол безопасности, такой как WPA2 или WPA, настройте точку доступа на открытую аутентификацию без защиты, чтобы проверить, проходит ли трафик.
- Как только система сможет подключиться к точке доступа, диагностируйте конфигурацию сети с помощью таких инструментов, как ping(8).
- Для низкоуровневой отладки существует множество инструментов. Сообщения отладки можно включить в слое поддержки протокола 802.11 с помощью wlandebug(8).

Часть II: Стандартные задачи

Теперь, когда основы рассмотрены, в этой части книги обсуждаются некоторые часто используемые возможности FreeBSD. Эти главы:

- Представляют популярные и полезные настольные приложения: браузеры, инструменты для работы, программы для просмотра документов и многое другое.
- Знакомят с несколькими мультимедийными инструментами, доступными для FreeBSD.
- Объясняют процесс сборки ядра FreeBSD со своими настройками для включения дополнительной функциональности.
- Подробно описывают систему печати, как для настольных, так и для сетевых принтеров.
- Показывают, как запускать Linux-приложения в системе FreeBSD.

Некоторые из этих глав рекомендуют предварительное ознакомление с другими материалами, что указано в аннотации в начале каждой главы.

Глава 8. Рабочие столы

8.1. Обзор

Хотя FreeBSD популярна в качестве сервера благодаря своей производительности и стабильности, она также отлично подходит для повседневного использования в качестве настольной системы. Более чем 37000 приложений, доступных в дереве портов FreeBSD, позволяют легко создать настроенный рабочий стол, способный запускать широкий спектр настольных приложений. В этой главе показано, как установить популярные окружения рабочего стола, а также настольные приложения, такие как веб-браузеры, офисные программы, средства просмотра документов и финансовое ПО.

Предварительные требования:

- Читатели этой главы уже должны знать, как установить либо X Window System, либо Wayland в FreeBSD.
- В этой главе читателям рекомендуется устанавливать официальные пакеты. Для сборки настраиваемых пакетов из портов обратитесь к разделу использование коллекции портов.

8.2. Рабочие столы

Этот раздел описывает, как установить и настроить некоторые популярные графические среды на системе FreeBSD. Графическая среда может варьироваться от простого оконного менеджера до полного набора приложений для рабочего стола.

Таблица 11. Поддерживаемые рабочие окружения

Имя	Лицензия	Пакет
Plasma	GPL 2.0 или более поздняя	x11/kde
GNOME	GPL 2.0 или более поздняя	x11/gnome
XFCE	GPL, LGPL, BSD	x11-wm/xfce4
MATE	GPL 2.0, LGPL 2.0	x11/mate
Cinnamon	GPL 2.0 или более поздняя	x11/cinnamon
LXQT	GPL, LGPL	x11-wm/lxqt

8.2.1. Plasma

Plasma — это удобная среда рабочего стола, созданная сообществом KDE. Она предоставляет набор приложений с единообразным внешним видом и поведением, стандартизированным меню и панелями инструментов, сочетаниями клавиш, цветовыми схемами, поддержкой интернационализации и централизованной настройкой рабочего стола через диалоговые окна.

Дополнительную информацию о KDE можно найти на сайте KDE homepage. Информацию,

специфичную для FreeBSD, см. на FreeBSD homepage at KDE.

Установка мета-пакета Plasma

Для установки мета-пакета рабочего стола Plasma с базовым набором KDE Applications, выполните:

```
# pkg install kde
```

Дополнительные приложения KDE помечены категорией `kde-applications`. Чтобы найти их, выполните следующую команду:

```
% pkg search -S origin kde-applications
```

Минимальная установка Plasma

Для установки минимальной версии рабочего стола Plasma выполните:

```
# pkg install plasma6-plasma
```

Эта установка **действительно** минимальна. Эмулятор терминала Konsole необходимо установить отдельно, выполнив:

```
# pkg install konsole
```

Настройка Plasma

Plasma использует dbus-daemon(1) в качестве шины сообщений и для аппаратной абстракции. Это приложение автоматически устанавливается как зависимость Plasma.

Включите службу D-BUS в rc.conf(5) для запуска при загрузке системы:

```
# sysrc dbus_enable="YES"
```

Запуск Plasma

Предпочтительным способом запустить Plasma автоматически является дисплейный менеджер x11/sddm. Для установки x11/sddm выполните:

```
# pkg install sddm
```

Включите службу SDDM в rc.conf(5) для автоматического запуска при загрузке системы:

```
# sysrc sddm_enable="YES"
```

Язык клавиатуры можно настроить в SDDM, выполнив следующую команду (например, для испанского языка):

```
# sysrc sddm_lang="es_ES"
```

Или запустите Plasma в сессии Xorg, выполнив вызов startx(1) в ручную. Для этого необходимо добавить следующую строку в файл ~/.xinitrc:

```
% echo "exec dbus-launch --exit-with-x11 ck-launch-session startplasma-x11" >
~/.xinitrc
```

В порядке эксперимента запустите Plasma вручную в сеансе Wayland, выполнив следующую команду:

```
% exec dbus-launch ck-launch-session startplasma-wayland
```

8.2.2. GNOME

GNOME — это удобная графическая среда рабочего стола. Она включает панель для запуска приложений и отображения состояния, рабочий стол, набор инструментов и приложений, а также набор соглашений, которые упрощают взаимодействие и обеспечение согласованности между приложениями.

Установка метапакета GNOME

Для установки метапакета GNOME с рабочим столом GNOME и приложениями выполните:

```
# pkg install gnome
```

Минимальная установка GNOME

Для установки мета-пакета GNOME-lite с облегчённой версией рабочего стола GNOME, содержащей только основы, выполните:

```
# pkg install gnome-lite
```

Настройка GNOME

Для работы GNOME необходимо подключить `/proc`. Добавьте следующую строку в `/etc/fstab`, чтобы автоматически подключать эту файловую систему при загрузке системы:

```
# Device                Mountpoint      FStype  Options         Dump    Pass#
proc                    /proc           procfs  rw              0       0
```

GNOME использует dbus-daemon(1) в качестве шины сообщений и для аппаратной абстракции. Это приложение автоматически устанавливается как зависимость GNOME.

Включите службу D-BUS в `/etc/rc.conf` для запуска при загрузке системы:

```
# sysrc dbus_enable="YES"
```

Запуск GNOME

GNOME Display Manager — это предпочтительный дисплейный менеджер для GNOME. GDM устанавливается как часть пакета GNOME.

Включите GDM в `/etc/rc.conf` для запуска при загрузке системы:

```
# sysrc gdm_enable="YES"
```

Второй способ запуска GNOME — это ручной вызов startx(1). Для этого необходимо добавить следующую строку в файл `~/.xinitrc`:

```
% echo "exec gnome-session" > ~/.xinitrc
```

8.2.3. XFCE

XFCE — это среда рабочего стола, основанная на GTK+, легковесная и предоставляющая простой, эффективный и удобный рабочий стол. Она полностью настраиваема, имеет главную панель с меню, апплетами и запусками приложений, предоставляет файловый менеджер и менеджер звука, а также поддерживает темы. Благодаря своей быстроте, легкости и эффективности, она идеально подходит для старых или медленных машин с ограниченными ресурсами памяти.

Установка XFCE

Для установки метапакета XFCE выполните:

```
# pkg install xfce
```

Настройка XFCE

XFCE использует dbus-daemon(1) для шины сообщений и абстракции оборудования. Это приложение автоматически устанавливается как зависимость XFCE.

Включите D-BUS в `/etc/rc.conf` для запуска при загрузке системы:

```
# sysrc dbus_enable="YES"
```

Запуск XFCE

x11/lightdm — это дисплейный менеджер, поддерживающий различные технологии отображения. Он является хорошим выбором, так как очень легковесен, требует мало памяти и обладает высокой производительностью.

Для установки выполните:

```
# pkg install lightdm lightdm-gtk-greeter
```

Включите lightdm в `/etc/rc.conf` для автоматического запуска при загрузке системы:

```
# sysrc lightdm_enable="YES"
```

Второй способ запустить XFCE — вручную вызвать startx(1). Для этого в `~/.xinitrc` должна быть следующая строка:

```
% echo '. /usr/local/etc/xdg/xfce4/xinitrc' > ~/.xinitrc
```

8.2.4. MATE

Среда рабочего стола MATE является продолжением GNOME 2. Она предоставляет интуитивно понятную и привлекательную среду рабочего стола, использующую традиционные метафоры.

Установка метапакета MATE

Для установки метапакета MATE, который включает в себя MATE Desktop с дополнительными приложениями, такими как текстовый редактор, менеджер архивов и т.д., выполните:

```
# pkg install mate
```

Минимальная установка MATE

Чтобы установить мета-пакет MATE lite с облегчённой версией рабочего стола MATE, содержащий только основы, выполните:

```
# pkg install mate-base
```

Настройка MATE

MATE требует подключения `/proc`. Добавьте следующую строку в `/etc/fstab`, чтобы автоматически подключать эту файловую систему при загрузке системы:

```
# Device                Mountpoint      FStype  Options         Dump    Pass#
proc                    /proc           procfs  rw              0       0
```

MATE использует dbus-daemon(1) для шины сообщений и абстракции оборудования. Это приложение автоматически устанавливается как зависимость MATE. Для запуска D-BUS при загрузке системы включите его в `/etc/rc.conf`:

```
# sysrc dbus_enable="YES"
```

Запуск MATE

x11/lightdm — это дисплейный менеджер, поддерживающий различные технологии отображения. Он является хорошим выбором, так как очень легковесен, требует мало памяти и обладает высокой производительностью.

Для установки выполните:

```
# pkg install lightdm lightdm-gtk-greeter
```

Включите lightdm в `/etc/rc.conf` для автоматического запуска при загрузке системы:

```
# sysrc lightdm_enable="YES"
```

Второй способ запустить MATE — вручную вызвать startx(1). Для этого необходимо добавить следующую строку в файл `~/.xinitrc`:

```
% echo "exec dbus-launch --exit-with-x11 ck-launch-session mate-session" > ~/.xinitrc
```

8.2.5. Cinnamon

Cinnamon — это UNIX®-совместимая рабочая среда, которая предоставляет передовые инновационные функции и традиционный пользовательский опыт. Компоновка рабочего стола похожа на Gnome 2. Базовые технологии являются ответвлением от Gnome Shell. Основной акцент сделан на том, чтобы пользователи чувствовали себя как дома, предоставляя им простую и комфортную рабочую среду.

Установка Cinnamon

Для установки пакета Cinnamon выполните:

```
# pkg install cinnamon
```

Настройка Cinnamon

Для работы Cinnamon требуется подключённый `/proc`. Добавьте следующую строку в `/etc/fstab`, чтобы автоматически подключать эту файловую систему при загрузке системы:

```
# Device                Mountpoint      FStype  Options         Dump    Pass#
proc                    /proc           procfs  rw              0       0
```

Cinnamon использует dbus-daemon(1) для шины сообщений и абстракции оборудования. Это приложение автоматически устанавливается как зависимость Cinnamon. Для запуска D-BUS при загрузке системы включите его в `/etc/rc.conf`:

```
# sysrc dbus_enable="YES"
```

Запуск Cinnamon

x11/lightdm — это дисплейный менеджер, поддерживающий различные технологии отображения. Он является хорошим выбором, так как очень легковесен, требует мало памяти и обладает высокой производительностью.

Для установки выполните:

```
# pkg install lightdm lightdm-gtk-greeter
```

Включите lightdm в `/etc/rc.conf` для автоматического запуска при загрузке системы:

```
# sysrc lightdm_enable="YES"
```

Второй способ запустить Cinnamon — вручную вызвать startx(1). Для этого необходимо добавить следующую строку в файл `~/.xinitrc`:

```
% echo "exec dbus-launch --exit-with-x11 ck-launch-session cinnamon-session" >
~/.xinitrc
```

8.2.6. LXQT

LXQt — это современная, простая в использовании и быстрая среда рабочего стола, основанная на технологиях Qt. Она создана для пользователей, которые ценят простоту, скорость и интуитивно понятный интерфейс. В отличие от большинства сред рабочего стола, LXQt также хорошо работает на менее производительных компьютерах.

Установка LXQT

Чтобы установить метапакет LXQT, выполните:

```
# pkg install lxqt
```

Настройка LXQT

LXQT требует подключения `/proc`. Добавьте следующую строку в `/etc/fstab`, чтобы автоматически подключать эту файловую систему при загрузке системы:

```
# Device                Mountpoint      FStype  Options         Dump    Pass#
proc                    /proc           procfs  rw              0       0
```

LXQT использует dbus-daemon(1) в качестве шины сообщений и для аппаратной абстракции. Это приложение автоматически устанавливается как зависимость LXQT.

Включите D-BUS в `/etc/rc.conf` для запуска при загрузке системы:

```
# sysrc dbus_enable="YES"
```

Запуск LXQT

Предпочтительным дисплейным менеджером для LXQT является x11/sddm. Для установки x11/sddm выполните:

```
# pkg install sddm
```

Включите службу SDDM в `/etc/rc.conf` для автоматического запуска при загрузке системы:

```
# sysrc sddm_enable="YES"
```

Язык клавиатуры можно настроить в SDDM, выполнив следующую команду (например, для испанского):

```
# sysrc sddm_lang="es_ES"
```

Второй способ запуска LXQT — вручную вызвать startx(1). Для этого потребуется добавить следующую строку в `~/.xinitrc`:

```
% echo "exec dbus-launch --exit-with-x11 ck-launch-session startlxqt" > ~/.xinitrc
```

8.3. Браузеры

В этом разделе описано, как установить и настроить некоторые популярные веб-браузеры в системе FreeBSD — от полнофункциональных браузеров с высоким потреблением ресурсов до текстовых браузеров с пониженным использованием ресурсов.

Таблица 12. Поддерживаемые браузеры

Имя	Лицензия	Пакет	Необходимые ресурсы
Firefox	MPL 2.0	www/firefox	Много
Chromium	BSD-3 и другие	www/chromium	Много
Браузер Iridium	BSD-3 и другие	www/iridium-browser	Много
Falkon	MPL 2.0	www/falkon-qtonly	Много
Konqueror	GPL 2.0 или более поздняя	x11-fm/konqueror	Средне
Gnome Web (Epiphany)	GPL 3.0 или новее	www/epiphany	Средне
qutebrowser	GPL 3.0 или новее	www/qutebrowser	Средне
Dillo	GPL 3.0 или новее	www/dillo2	Мало
Ссылки	GPL 2.0 или более поздняя	www/links	Мало
w3m	MIT	www/w3m	Мало

8.3.1. Firefox

Firefox — это браузер с открытым исходным кодом, который включает в себя движок отображения HTML, соответствующий стандартам, вкладки, блокировку всплывающих окон, расширения, улучшенную безопасность и многое другое. Firefox основан на кодовой базе Mozilla.

Для установки пакета последней выпущенной версии Firefox выполните:

```
# pkg install firefox
```

Для установки Firefox Extended Support Release (ESR) выполните:

```
# pkg install firefox-esr
```

8.3.2. Chromium

Chromium — это проект браузера с открытым исходным кодом, целью которого является создание более безопасного, быстрого и стабильного веб-браузера. Chromium поддерживает вкладки, блокировку всплывающих окон, расширения и многое другое. Chromium — это

проект с открытым исходным кодом, на основе которого создан браузер Google Chrome.

Чтобы установить Chromium, выполните:

```
# pkg install chromium
```

Исполняемый файл Chromium находится по пути /usr/local/bin/chrome, а не /usr/local/bin/chromium.

Поисковые системы в Chromium привязаны к регионам. Обратитесь к секции Использование локализации для информации о конфигурировании соответствующих настроек.

8.3.3. Браузер Iridium

Iridium — это бесплатный, открытый и свободный модифицированный браузер на основе кодовой базы Chromium, в котором усилена защита приватности в нескольких ключевых аспектах. Автоматическая передача частичных запросов, ключевых слов и метрик в центральные службы блокируется и происходит только с согласия пользователя.

Для установки Iridium выполните:

```
# pkg install iridium-browser
```

8.3.4. Falkon

Falkon — это относительно новый и очень быстрый браузер на основе QtWebEngine. Он стремится быть легковесным веб-браузером, доступным на всех основных платформах. Falkon обладает всеми стандартными функциями, которые можно ожидать от веб-браузера, включая закладки, историю (доступные также в боковой панели) и вкладки. Кроме того, плагин AdBlock может блокировать рекламу, Click2Flash — Flash-содержимое, а SSL Manager позволяет редактировать локальную базу данных сертификатов CA.

Для установки Falkon выполните:

```
# pkg install falkon
```

8.3.5. Konqueror

Konqueror — это не просто веб-браузер, он также является файловым менеджером и мультимедийным просмотрщиком. Он поддерживает WebKit, механизм визуализации, используемый многими современными браузерами, включая Chromium, а также собственный движок KHTML.

Чтобы установить Konqueror, выполните:

```
# pkg install konqueror
```

8.3.6. Gnome Web (Epiphany)

Gnome Web (Epiphany) — это веб-браузер, разработанный для максимальной легкости и скорости, в ущерб многим функциям, присутствующим в других браузерах.

Для установки Gnome Web (Epiphany) выполните:

```
# pkg install epiphany
```

8.3.7. qutebrowser

Qutebrowser — это браузер с минимальным графическим интерфейсом, ориентированный на управление с клавиатуры. Он написан на Python и PyQt5 и распространяется как свободное программное обеспечение под лицензией GPL.

Для установки qutebrowser выполните:

```
# pkg install qutebrowser
```

8.3.8. Dillo

Dillo стремится быть мультиплатформенным альтернативным браузером, который отличается компактностью, стабильностью, удобством для разработчиков, практичностью, скоростью работы и расширяемостью. Эта новая, экспериментальная версия Dillo основана на инструментарии FLTK, а не на GTK1, и была существенно переработана.

Для установки Dillo выполните:

```
# pkg install dillo
```

8.3.9. Ссылки

Веб-браузер, подобный lynx, с текстовым и графическим режимами, поддерживающий множество функций, таких как отображение таблиц, меню и т.д.

Для установки Links выполните:

```
# pkg install links
```

8.3.10. w3m

w3m — это программа для просмотра текстовых страниц (пейджер) и текстовый веб-браузер. Это приложение похоже на Lynx, но обладает рядом функций, которых нет в Lynx, таких как отображение таблиц и фреймов.

Для установки w3m выполните:

```
# pkg install w3m
```

8.4. Инструменты разработки

Этот раздел описывает, как установить и настроить некоторые популярные инструменты разработки в системе FreeBSD.

Таблица 13. Поддерживаемые инструменты разработки

Имя	Лицензия	Пакет	Необходимые ресурсы
Visual Studio Code	MIT	editors/vscode	Много
Qt Creator	QtGPL	devel/qtcreator	Много
Kdevelop	GPL 2.0 или более поздняя и LGPL 2.0 или более поздняя	devel/kdevelop	Много
Eclipse IDE	EPL	java/eclipse	Много
Vim	VIM	editors/vim	Мало
Neovim	Apache 2.0	editors/neovim	Мало
GNU Emacs	GPL 3.0 или новее	editors/emacs	Мало

8.4.1. Visual Studio Code

Visual Studio Code — это инструмент, который сочетает простоту редактора кода с необходимыми для разработчиков возможностями цикла редактирования, сборки и отладки. Он предоставляет комплексную поддержку редактирования и отладки, модель расширяемости и лёгкую интеграцию с существующими инструментами.

Для установки Visual Studio Code выполните:

```
# pkg install vscode
```

8.4.2. Qt Creator

Qt Creator — это кроссплатформенная IDE (интегрированная среда разработки), созданная с учётом потребностей разработчиков Qt. Функциональные возможности Qt Creator

включают:

- редактор кода с поддержкой C++, QML и ECMAscript;
- быстрые инструменты для навигации по коду;
- статическая проверка кода и подсказки по стилю во время набора;
- контекстно-зависимая справка;
- визуальный отладчик;
- интегрированный графический интерфейс и конструктор форм.

Для установки Qt Creator выполните:

```
# pkg install qtcreator
```

8.4.3. kdevelop

Открытая, многофункциональная IDE с поддержкой расширений через плагины для языков программирования C/C++ и других. Основана на KDevPlatform, а также библиотеках KDE и Qt, и разрабатывается с 1998 года.

Для установки kdevelop выполните:

```
# pkg install kdevelop
```

8.4.4. Eclipse IDE

Платформа Eclipse — это открытая расширяемая среда разработки (IDE) для чего угодно и в то же время ничего конкретного. Платформа Eclipse предоставляет строительные блоки и основу для создания и запуска интегрированных инструментов разработки программного обеспечения. Платформа Eclipse позволяет разработчикам инструментов независимо создавать инструменты, которые интегрируются с инструментами других разработчиков.

Для установки Eclipse IDE выполните:

```
# pkg install eclipse
```

8.4.5. Vim

Vim — это высоконастраиваемый текстовый редактор, созданный для эффективного редактирования текста. Это улучшенная версия редактора vi, поставляемого с большинством UNIX-систем.

Vim часто называют «редактором программиста», и он настолько полезен для программирования, что многие считают его полноценной IDE. Однако он предназначен не только для программистов. Vim идеально подходит для любого вида редактирования текста

— от написания электронных писем до правки конфигурационных файлов.

Для установки Vim выполните:

```
# pkg install vim
```

8.4.6. Neovim

Neovim представляет собой радикальный рефакторинг editors/vim. Это полная переработка кодовой базы с множеством улучшений, включая разумные настройки по умолчанию, встроенный эмулятор терминала, асинхронную архитектуру плагинов и мощные API, ориентированные на скорость и расширяемость. При этом сохраняется полная совместимость с почти всеми плагинами и скриптами Vim.

Для установки Neovim выполните:

```
# pkg install neovim
```

8.4.7. GNU Emacs

GNU Emacs — это расширяемый, настраиваемый, свободный текстовый редактор. В его основе лежит интерпретатор Emacs Lisp, диалекта языка программирования Lisp с расширениями для поддержки редактирования текста.

Чтобы установить GNU Emacs, выполните:

```
# pkg install emacs
```

8.5. Настольные офисные приложения

Когда речь заходит о продуктивности, пользователи часто ищут офисный пакет или простой в использовании текстовый редактор. Хотя некоторые окружения рабочего стола, такие как KDE Plasma, предоставляют офисный пакет, стандартного пакета для продуктивности не существует. Для FreeBSD доступно несколько офисных пакетов и графических текстовых процессоров, независимо от установленных окружений рабочего стола.

В этом разделе показано, как установить следующее популярное программное обеспечение для повышения продуктивности, а также указано, является ли приложение ресурсоемким, требует ли оно много времени для компиляции из портов или имеет какие-либо значительные зависимости.

Таблица 14. Поддерживаемые офисные пакеты для настольных компьютеров

Имя	Лицензия	Пакет	Необходимые ресурсы
LibreOffice	MPL 2.0	editors/libreoffice	Много
Calligra Suite	LGPL и GPL	editors/calligra	Средне
AbiWord	GPL 2.0 или более поздняя	editors/abiword	Средне

8.5.1. LibreOffice

LibreOffice — это свободный офисный пакет, разрабатываемый The Document Foundation. Он совместим с другими крупными офисными пакетами и доступен на различных платформах. Это переименованная форк Apache OpenOffice, включающая приложения, входящие в полный офисный пакет: текстовый процессор, электронные таблицы, программу для создания презентаций, графический редактор, систему управления базами данных и инструмент для создания и редактирования математических формул. LibreOffice доступен на множестве языков, а интернационализация охватывает интерфейсы, средства проверки орфографии и словари. Дополнительную информацию о LibreOffice можно найти на libreoffice.org.

Для установки LibreOffice выполните:

```
# pkg install libreoffice
```

Пакет LibreOffice по умолчанию поставляется только на английском языке. Для получения локализованной версии LibreOffice необходимо установить языковой пакет. Например, для версии на испанском языке необходимо установить пакет editors/libreoffice-es с помощью команды:

```
# pkg install es-libreoffice
```

8.5.2. Calligra

Среда рабочего стола KDE Plasma включает офисный пакет, который можно установить отдельно от KDE Plasma. Calligra содержит стандартные компоненты, которые можно встретить в других офисных пакетах. Words — это текстовый процессор, Sheets — программа для работы с электронными таблицами, Stage предназначен для управления презентациями, а Karbon используется для создания графических документов.

Для установки Calligra выполните:

```
# pkg install calligra
```

8.5.3. AbiWord

AbiWord — это бесплатная программа для обработки текстов, внешне и функционально схожая с Microsoft® Word. Она быстрая, обладает множеством функций и удобна в использовании.

AbiWord может импортировать или экспортировать множество форматов файлов, включая некоторые проприетарные, такие как Microsoft® .rtf.

Для установки AbiWord выполните:

```
# pkg install abiword
```

8.6. Просмотрщики документов

Некоторые новые форматы документов приобрели популярность со времен появления UNIX®, и программы для их просмотра могут отсутствовать в базовой системе. В этом разделе показано, как установить следующие программы для просмотра документов:

Таблица 15. Поддерживаемые программы для просмотра документов

Имя	**Лицензия**	**Пакет**	**Необходимые ресурсы**
Okular	GPL 2.0	graphics/okular	Много
Evince	GPL 2.0	graphics/evince	Средне
ePDFView	GPL 2.0	graphics/epdfview	Средне
Xpdf	GPL 2.0	graphics/xpdf	Мало
Zathura	Zlib	graphics/zathura	Мало

8.6.1. Okular

Okular — это универсальный просмотрщик документов, часть проекта KDE Plasma.

Okular сочетает в себе превосходную функциональность с универсальностью поддержки различных типов документов, таких как PDF, Postscript, DjVu, CHM, XPS, ePub и других.

Чтобы установить Okular, выполните:

```
# pkg install okular
```

8.6.2. Evince

Evince — это программа для просмотра документов в различных форматах, включая PDF и Postscript. Часть проекта GNOME. Цель Evince — заменить такие программы для просмотра документов, как ggv и gpdf, на одно простое приложение.

Для установки Evince выполните:

```
# pkg install evince
```

8.6.3. ePDFView

ePDFView — это легковесная программа для просмотра PDF-документов, использующая только библиотеки Gtk+ и Poppler. Цель ePDFView — создать простой просмотрщик PDF-документов, аналогичный Evince, но без использования библиотек GNOME.

Для установки ePDFView выполните:

```
# pkg install epdfview
```

8.6.4. Xpdf

Для пользователей, предпочитающих небольшой просмотрщик PDF в FreeBSD, Xpdf предоставляет легковесное и эффективное решение, требующее минимум ресурсов. Он использует стандартные шрифты X и не требует дополнительных инструментариев.

Для установки Xpdf выполните:

```
# pkg install xpdf
```

8.6.5. Zathura

Zathura — это высоконастраиваемая и функциональная программа для просмотра документов. Она предоставляет минималистичный и компактный интерфейс, а также простоту использования, ориентированную в основном на взаимодействие с клавиатурой.

Для установки zathura с поддержкой PDF выполните:

```
# pkg install zathura zathura-pdf-mupdf
```

Кроме того, можно установить graphics/zathura-pdf-poppler для альтернативной поддержки PDF, graphics/zathura-ps для поддержки PostScript, graphics/zathura-djvu для поддержки DjVu и graphics/zathura-cb для поддержки комиксов.

8.7. Финансы

Для управления личными финансами на рабочем столе FreeBSD можно установить мощные и удобные приложения. Некоторые из них совместимы с распространёнными форматами файлов, такими как форматы, используемые Quicken и Excel.

Этот раздел охватывает следующие программы:

Таблица 16. Поддерживаемые программы для финансов

Имя	Лицензия	Пакет	Необходимые ресурсы
KMyMoney	GPL 2.0	finance/kmymoney	Много
GnuCash	GPL 2.0 и GPL 3.0	finance/gnucash	Много

8.7.1. KMyMoney

KMyMoney — это приложение для управления личными финансами, созданное сообществом KDE. KMyMoney стремится предоставить важные функции, которые можно встретить в коммерческих приложениях для управления финансами. Среди его особенностей — простота использования и поддержка двойной бухгалтерии. KMyMoney поддерживает импорт из стандартных файлов Quicken QIF, отслеживание инвестиций, работу с несколькими валютами и предоставляет множество отчётов.

Для установки KMyMoney выполните:

```
# pkg install kmymoney
```

8.7.2. GnuCash

GnuCash — это часть проекта GNOME, направленного на предоставление пользователям удобных, но мощных приложений. GnuCash позволяет отслеживать доходы и расходы, банковские счета и акции. Он сочетает интуитивно понятный интерфейс с профессиональным функционалом.

GnuCash предоставляет умный регистр, иерархическую систему счетов, а также множество сочетаний клавиш и методов автодополнения. Он позволяет разделять одну транзакцию на несколько более детализированных частей. GnuCash поддерживает импорт и объединение файлов Quicken QIF. Также программа работает с большинством международных форматов дат и валют.

Для установки GnuCash выполните:

```
# pkg install gnucash
```

Глава 9. Мультимедиа

9.1. Обзор

Глава о мультимедиа предоставляет обзор поддержки мультимедиа в FreeBSD. Мультимедийные приложения и технологии стали неотъемлемой частью современных вычислений, и FreeBSD обеспечивает надёжную и стабильную поддержку широкого спектра мультимедийного оборудования и программного обеспечения. В этой главе рассматриваются различные мультимедийные компоненты, такие как аудио, видео и обработка изображений. Также обсуждаются различные медиаформаты и кодеки, а также инструменты и приложения для создания и воспроизведения мультимедиа. Дополнительно глава охватывает настройку мультимедийной системы, устранение неполадок и оптимизацию. FreeBSD предлагает надёжную платформу для мультимедийной работы и энтузиастам мультимедиа, и профессиональным создателям контента. Эта глава призвана помочь максимально использовать мультимедийные возможности FreeBSD, предоставляя полезную информацию и практические примеры для начала работы.

9.2. Настройка звуковой карты

По умолчанию FreeBSD автоматически определяет звуковую карту, используемую в системе. FreeBSD поддерживает широкий спектр звуковых карт. Список поддерживаемых звуковых карт можно найти в sound(4).

Для загрузки модуля звуковой карты необходимо только в том случае, если FreeBSD не определила его корректно.

Где неизвестно, какая звуковая карта установлена в системе или какой модуль использовать, можно загрузить метадрайвер `snd_driver`, выполнив следующую команду:

```
# kldload snd_driver
```

В качестве альтернативы, для загрузки драйвера как модуля во время загрузки, добавьте следующую строку в /boot/loader.conf:

```
snd_driver_load="YES"
```

9.2.1. Тестирование звука

Чтобы подтвердить, что звуковая карта обнаружена, можно выполнить следующую команду:

```
% dmesg | grep pcm
```

Вывод должен быть похож на следующий:

```
pcm0: <Conexant CX20561 (Hermosa) (Analog 2.0+HP/2.0)> at nid 26,22 and 24 on hdaa0
pcm1: <Conexant CX20561 (Hermosa) (Internal Analog Mic)> at nid 29 on hdaa0
```

Состояние звуковой карты также можно проверить с помощью этой команды:

```
# cat /dev/sndstat
```

Вывод должен быть похож на следующий:

```
Installed devices:
pcm0: <Conexant CX20561 (Hermosa) (Analog 2.0+HP/2.0)> (play/rec) default
pcm1: <Conexant CX20561 (Hermosa) (Internal Analog Mic)> (rec)
```

Если в списке нет устройств `pcm`, проверьте, что загружен правильный драйвер устройства. Если всё в порядке, звуковая карта должна теперь работать в FreeBSD.

beep(1) может использоваться для создания звука, подтверждающего, что звуковая карта работает:

```
% beep
```

9.2.2. Микшер

В FreeBSD существуют различные утилиты для установки и отображения значений микшера звуковой карты, работающие на основе FreeBSD Sound System:

Таблица 17. Поддерживаемые пакеты микшера

Имя	**Лицензия**	**Пакет**	`Toolkit`
mixer(8)	BSD-2	Включено в базовую систему	CLI
dsbmixer	BSD-2	audio/dsbmixer	Qt
Виджет аудио KDE Plasma	GPL 2.0	audio/plasma6-plasma-pa	Qt
mixertui	BSD-2	audio/mixertui	TUI

9.2.3. Звук графической карты

Графические карты часто оснащены собственными встроенными звуковыми устройствами, и может быть неясно, какое из них используется по умолчанию. Чтобы это проверить, выполните команду `dmesg` и найдите записи `pcm`, чтобы определить, как система перечисляет выходы. Выполните следующую команду:

```
% dmesg | grep pcm
```

Вывод выглядит примерно так:

```
pcm0: <HDA NVIDIA (Unknown) PCM #0 DisplayPort> at cad 0 nid 1 on hdac0
pcm1: <HDA NVIDIA (Unknown) PCM #0 DisplayPort> at cad 1 nid 1 on hdac0
pcm2: <HDA NVIDIA (Unknown) PCM #0 DisplayPort> at cad 2 nid 1 on hdac0
pcm3: <HDA NVIDIA (Unknown) PCM #0 DisplayPort> at cad 3 nid 1 on hdac0
hdac1: HDA Codec #2: Realtek ALC889
pcm4: <HDA Realtek ALC889 PCM #0 Analog> at cad 2 nid 1 on hdac1
pcm5: <HDA Realtek ALC889 PCM #1 Analog> at cad 2 nid 1 on hdac1
pcm6: <HDA Realtek ALC889 PCM #2 Digital> at cad 2 nid 1 on hdac1
pcm7: <HDA Realtek ALC889 PCM #3 Digital> at cad 2 nid 1 on hdac1
```

Графическая карта (NVIDIA®) была перечислена до звуковой карты (Realtek®), при этом звуковая карта отображается как pcm4. Систему можно настроить на использование звуковой карты в качестве устройства по умолчанию, выполнив следующую команду:

```
# sysctl hw.snd.default_unit=4
```

Чтобы сделать это изменение постоянным, добавьте следующую строку в /etc/sysctl.conf:

```
hw.snd.default_unit=4
```

9.2.4. Автоматическое переключение на наушники

Некоторые системы могут испытывать трудности при переключении между аудиовыходами, но, к счастью, FreeBSD позволяет настроить автоматическое переключение в device.hints.

Определите, как система перечисляет аудиовыходы, выполнив следующую команду:

```
% dmesg | grep pcm
```

Вывод выглядит примерно так:

```
pcm0: <Realtek ALC892 Analog> at nid 23 and 26 on hdaa0
pcm1: <Realtek ALC892 Right Analog Headphones> at nid 22 on hdaa0
```

Добавьте следующие строки в файл /boot/device.hints:

```
hint.hdac.0.cad0.nid22.config="as=1 seq=15 device=Headphones"
```

```
hint.hdac.0.cad0.nid26.config="as=2 seq=0 device=speakers"
```

Имейте в виду, что эти значения приведены для указанного выше примера. Они могут отличаться в зависимости от системы.

9.2.5. Качество ресемплинга

Процесс изменения частоты дискретизации аудиопотока называется ресемплингом.

Обычно это происходит автоматически в таких ситуациях, как запись звука через DI (цифровой интерфейс) или просто при воспроизведении аудиофайлов с очень низким разрешением.

Пример: DI использует 192000 кГц, а приложение принимает только 48000 кГц (или наоборот).

Если возможно, это следует настраивать за пределами драйвера звукового устройства, но для случаев, когда это невозможно (или оборудование плохо выполняет ресемплинг), у вас есть возможность влиять на качество, используемое при ресемплинге.

Если вы не уверены, нужна ли эта настройка, то для обычного прослушивания музыки, проведения встреч и тому подобного значение по умолчанию вполне подходит, и большинство приложений в любом случае используют 48000 кГц, так что передискретизация происходит редко.

При создании музыки, записи звуков в других случаях, где качество имеет значение, или если вы аудиофил, вы можете повысить качество ресемплинга.

Для достижения максимально возможного качества передискретизации выполните следующую команду:

```
# sysctl hw.snd.feeder_rate_quality=4①
# sysctl hw.snd.feeder_rate_round=0②
```

① 100 дБ в полосе задерживания, 164 отвода и 97% полосы пропускания

② Крупные простые деления для высокой точности

Чтобы сохранить эти значения после перезагрузки, их можно добавить в `/etc/sysctl.conf`.

Важно помнить, что изменение качества ресемплинга влияет на использование процессора и задержку.

Более подробную информацию о ресемплинге можно найти в sound(4)

9.2.6. Устранение неполадок со звуком

Некоторые распространённые сообщения об ошибках и их решения:

Таблица 18. Общие сообщения об ошибках

Ошибка	Решение
`xxx: can't open /dev/dsp!`	Наберите `fstat | grep dsp` для проверки, удерживает ли другое приложение устройство открытым. Основные источники проблем — это `esound` и поддержка звука в `KDE`.

Программы, использующие audio/pulseaudio, могут потребовать перезапуска демона audio/pulseaudio для применения изменений в `hw.snd.default_unit`. В качестве альтернативы, настройки audio/pulseaudio можно изменить на лету. pacmd(1) открывает командное соединение с демоном audio/pulseaudio:

```
# pacmd
Welcome to PulseAudio 14.2! Use "help" for usage information.
>>>
```

Следующая команда изменяет устройство вывода по умолчанию на карту номер 4, как в предыдущем примере:

```
set-default-sink 4
```

Не используйте команду `exit` для выхода из интерфейса командной строки. Это приведёт к завершению работы демона audio/pulseaudio. Вместо этого используйте `Ctrl` + `D`.

9.3. Аудиоплееры

В этом разделе представлено некоторое программное обеспечение из Коллекции портов FreeBSD, которое можно использовать для воспроизведения аудио.

Таблица 19. Пакеты аудиоплееров

Имя	Лицензия	Пакет	`Toolkit`
Elisa	LGPL 3.0	audio/elisa	Qt
GNOME Music	GPL 2.0	audio/gnome-music	GTK+
Audacious	BSD-2	multimedia/audacious	Qt
cmus — C* музыкальный проигрыватель	GPL 2.0	audio/cmus	TUI

9.3.1. Elisa

Elisa — это музыкальный проигрыватель, разработанный сообществом KDE, который стремится быть простым и приятным в использовании.

Для установки Elisa выполните:

```
# pkg install elisa
```

9.3.2. GNOME Music

GNOME Music - это новое приложение для воспроизведения музыки в среде GNOME. Оно сочетает в себе элегантный и захватывающий интерфейс для просмотра музыки с простым и интуитивно понятным управлением.

Для установки GNOME Music выполните:

```
# pkg install gnome-music
```

9.3.3. Audacious

Audacious — это аудиоплеер с открытым исходным кодом. Потомок XMMS, он воспроизводит музыку так эффективно используя ресурсы компьютера.

Для установки Audacious выполните:

```
# pkg install audacious-qt6 audacious-plugins-qt6
```

Audacious изначально поддерживает OSS, но его необходимо настроить в параметрах на вкладке Audio.

9.3.4. cmus — C* музыкальный проигрыватель

cmus — это небольшой, быстрый и мощный консольный музыкальный плеер для Unix-подобных операционных систем.

Помимо различных плагинов ввода и вывода, он поддерживает CUE-листы, использует Unicode, поддерживает скробблинг (например, last.fm или libre.fm), может запускать внешние программы для выбранных файлов, такие как редакторы тегов, управляется через UNIX-сокеты и многое другое.

Для установки cmus выполните:

```
# pkg install cmus
```

9.4. Видеоплееры

В этом разделе представлено некоторое программное обеспечение из Коллекции портов FreeBSD, которое можно использовать для воспроизведения видео.

Таблица 20. Пакеты видеоплееров

Имя	Лицензия	Пакет	Toolkit
mpv	GPL 2.0	multimedia/mpv	CLI
SMPlayer	GPL 2.0	multimedia/smplayer	Qt
VLC media player	GPL 2.0	multimedia/vlc	Qt
Kodi (XBMC)	GPL 2.0	multimedia/kodi	X11

9.4.1. mpv

mpv — это свободный (в смысле свободы, а не бесплатности) медиаплеер для командной строки. Он поддерживает широкий спектр форматов медиафайлов, аудио- и видеокодеков, а также типов субтитров. mpv может использовать большинство API аппаратного декодирования на всех платформах.

Ориентированный также на видеолюбителей, он имеет несколько высококачественных алгоритмов масштабирования вверх/вниз, которыми, как и всеми настройками, можно управлять через профили. Профили можно выбирать с помощью параметров командной строки или на основе условий, таких как имя каталога.

Она также поддерживает высококачественное преобразование HDR в SDR с тонкой настройкой управления, позволяя каждому наслаждаться HDR-контентом на любом дисплее.

Для установки mpv выполните:

```
# pkg install mpv
```

Вот короткий пример ~/.config/mpv/mpv.conf

```
vo=gpu-next①
gpu-context=wayland②
hwdec=auto③
screenshot-jpeg-quality=100
scale=ewa_lanczos4sharpest④
dscale=ewa_lanczos4sharpest

[TTRPG]⑤
profile-cond=path:find('ttrpg')⑥
profile=fast⑦
save-position-on-quit=yes

[hdr]⑧
scale=bilinear
dscale=bilinear
dither=no
correct-downscaling=no
```

```
linear-downscaling=no
sigmoid-upscaling=no
hdr-compute-peak=yes
hdr-peak-percentile=99.995
hdr-contrast-recovery=0.30
```

① Использует новый видеовыход gpu-next. Многие функции требуют этого и/или обеспечивают более высокое качество, если этот параметр включен

② Пользователи Wayland должны вручную установить это при появлении графических сбоев или возникновении сбоев программы

③ Автоматически определяет доступность аппаратного декодирования GPU и использует его

④ Наилучший алгоритм масштабирования. Отключите, если ваше оборудование недостаточно быстрое и видео начинает пропускать кадры

⑤ Пример профиля для длинных видео записанных сессий TTRPG

⑥ Применяется только в том случае, если путь содержит ttrpg

⑦ Использует встроенный быстрый профиль для минимально возможного использования CPU/GPU

⑧ Пример профиля для пользователей с медленной встроенной графикой (iGPU), которые всё таки хотят наслаждаться HDR-контентом

Используя эту примерную конфигурацию, профиль TTRPG автоматически применяется, если выполняется условие "Path contains ttrpg".

Чтобы просмотреть HDR-видео на iGPU, вручную укажите профиль с помощью следующей команды:

```
$ mpv --profile=hdr hdr_video.mkv
```

За дополнительными примерами использования mpv обращайтесь к справочному руководству mpv.

9.4.2. SMPlayer

SMPlayer предназначен быть полноценным фронтендом для MPlayer, начиная с базовых функций, таких как воспроизведение видео, DVD и VCD, до более продвинутых возможностей, включая поддержку фильтров MPlayer и многое другое.

Для установки SMPlayer выполните:

```
# pkg install smplayer
```

9.4.3. VLC media player

VLC media player — это высокопортативный мультимедийный проигрыватель, поддерживающий различные аудио- и видеоформаты (MPEG-1, MPEG-2, MPEG-4, DivX, mp3, ogg и другие), а также DVD, VCD и различные протоколы потокового вещания. Он также может использоваться в качестве сервера для трансляции в режиме unicast или multicast по IPv4 или IPv6 в высокоскоростных сетях. VLC также умеет транскодировать медиафайлы на лету для потоковой передачи или сохранения на диск.

Для установки VLC выполните:

```
# pkg install vlc
```

9.4.4. Kodi (XBMC)

Kodi (ранее известный как XBMC) — это бесплатный и открытый кроссплатформенный медиаплеер и развлекательный центр. Он позволяет пользователям воспроизводить и просматривать большинство видео, музыки, подкастов и других цифровых медиафайлов с локальных и сетевых носителей, а также из интернета.

Для установки Kodi выполните:

```
# pkg install kodi
```

9.5. Конференции и встречи

Рабочая среда FreeBSD может быть использована для участия в видеоконференциях. В этом разделе будет объяснено, как настроить веб-камеру и какие приложения для видеоконференций поддерживаются в FreeBSD.

9.5.1. Настройка веб-камеры

Для предоставления FreeBSD доступа к веб-камере и её настройки необходимо установить определённые утилиты:

- multimedia/webcamd — это демон, который обеспечивает работу сотен различных USB-вебкамер и DVB USB-устройств.
- multimedia/pwcview — это приложение, которое можно использовать для просмотра видеопотока с веб-камеры.

Для установки необходимых утилит выполните:

```
# pkg install webcamd pwcview
```

Включите службу webcamd(8) в `/etc/rc.conf`, чтобы она запускалась при загрузке системы:

```
# sysrc webcamd_enable=YES
```

Пользователь должен состоять в группе `webcamd`. Чтобы добавить пользователя в группу `webcamd`, выполните следующую команду:

```
# pw groupmod webcamd -m username
```

Поскольку multimedia/webcamd требует модуль cuse(3), этот модуль должен быть загружен выполнением следующей команды:

```
# kldload cuse
```

Чтобы загрузить cuse(3) при запуске системы, выполните команду:

```
# sysrc kld_list+=cuse
```

После установки утилит список доступных веб-камер можно просмотреть с помощью webcamd(8):

```
# webcamd -l
```

Вывод должен быть похож на следующий:

```
webcamd [-d ugen0.2] -N SunplusIT-Inc-HP-TrueVision-HD-Camera -S unknown -M 0 ①
webcamd [-d ugen1.3] -N Realtek-802-11n-WLAN-Adapter -S 00e04c000001 -M 0
```

① Доступные веб-камеры

Настройте доступную веб-камеру, выполнив следующую команду:

```
# sysrc webcamd_0_flags="-N SunplusIT-Inc-HP-TrueVision-HD-Camera" ①
```

Обратите внимание, что если это веб-камера с поддержкой plug-and-play USB, то изменение USB-порта, к которому она подключена, приведёт к изменению вывода команды `webcamd -l`, а именно, идентификатора устройства, и запись в rc.conf может потребовать обновления. Для ноутбуков со встроенными USB-веб-камерами это не должно быть проблемой. Чтобы избежать такой ситуации, или используйте имя устройства (опция `-N`), или/и определяйте соответствие устройства по серийному номеру, если он известен (опция `-S`) и отображается в выводе команды `webcamd -l`. Для статических устройств, например, интегрированных видеокамер ноутбуков, вы можете использовать идентификатор устройства (опция `-d`).

Сервис webcamd(8) должен быть запущен выполнением следующей команды:

```
# service webcamd start
```

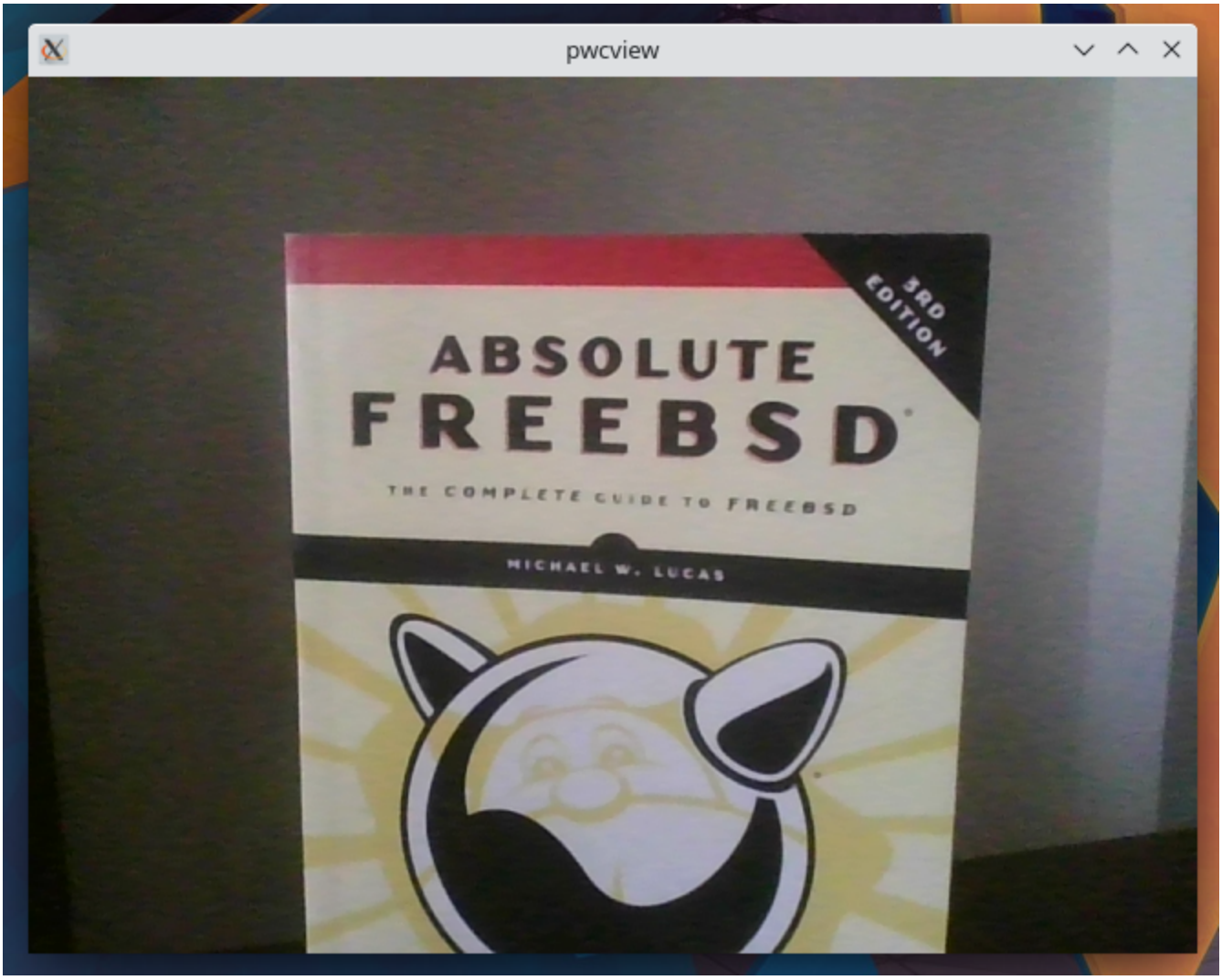
Вывод должен быть похож на следующий:

```
Starting webcamd.
webcamd 1616 - - Attached to ugen0.2[0]
```

multimedia/pwcview можно использовать для проверки правильной работы веб-камеры. Следующая команда может быть использована для запуска multimedia/pwcview:

```
% pwcview -f 30 -s vga
```

Тогда multimedia/pwcview отобразит изображение с веб-камеры:



9.5.2. Статус программного обеспечения для встреч

FreeBSD в настоящее время поддерживает следующие инструменты для проведения

видеоконференций.

Таблица 21. Установка программного обеспечения

Имя	Статус Firefox	Статус Chromium	Website
Microsoft Teams	Не работает	Работает	https://teams.live.com
Google Meet	Работает	Работает	https://meet.google.com/
Zoom	Работает	Работает	https://zoom.us
Jitsi	Не работает	Работает	https://meet.jit.si/
BigBlueButton	Не работает	Работает	https://bigbluebutton.org/

9.6. Сканеры изображений

В FreeBSD доступ к сканерам изображений обеспечивается через SANE (Scanner Access Now Easy), который доступен в Коллекции портов FreeBSD.

9.6.1. Проверка сканера

Прежде чем пытаться выполнить любую настройку, важно убедиться, что сканер поддерживается SANE.

При подключённом сканере выполните следующую команду, чтобы получить список всех подключённых USB-устройств:

```
# usbconfig list
```

Вывод должен быть похож на следующий:

```
ugen4.2: <LITE-ON Technology USB NetVista Full Width Keyboard.> at usbus4, cfg=0
md=HOST spd=LOW (1.5Mbps) pwr=ON (70mA)
ugen4.3: <Logitech USB Optical Mouse> at usbus4, cfg=0 md=HOST spd=LOW (1.5Mbps)
pwr=ON (100mA)
ugen3.2: <HP Deskjet 1050 J410 series> at usbus3, cfg=0 md=HOST spd=HIGH (480Mbps)
pwr=ON (2mA)
```

Выполните следующую команду для получения `idVendor` и `idProduct`:

```
# usbconfig -d 3.2 dump_device_desc
```

Обратите внимание, что сканер является устройством plug-and-play, и изменение USB-порта, к которому он подключен, приведёт к изменению вывода команды `usbconfig list`.

Вывод должен быть похож на следующий:

```
ugen3.2: <HP Deskjet 1050 J410 series> at usbus3, cfg=0 md=HOST spd=HIGH (480Mbps)
pwr=ON (2mA)

bLength = 0x0012
bDescriptorType = 0x0001
bcdUSB = 0x0200
bDeviceClass = 0x0000  <Probed by interface class>
bDeviceSubClass = 0x0000
bDeviceProtocol = 0x0000
bMaxPacketSize0 = 0x0040
idVendor = 0x03f0
idProduct = 0x8911
bcdDevice = 0x0100
iManufacturer = 0x0001  <HP>
iProduct = 0x0002  <Deskjet 1050 J410 series>
bNumConfigurations = 0x0001
```

После получения `idVendor` и `idProduct` необходимо проверить в списке поддерживаемых устройств SANE, поддерживается ли сканер, выполнив фильтрацию по idProduct.

9.6.2. Настройка SANE

SANE предоставляет доступ к сканеру через бэкенды. Для возможности сканирования в FreeBSD необходимо установить пакет graphics/sane-backends, выполнив следующую команду:

```
# pkg install sane-backends
```

Некоторые USB-сканеры требуют загрузки микропрограммы. Как, например, сканер HP, использованный в примере выше, для которого необходимо установить пакет print/hplip.

После установки необходимых пакетов необходимо настроить devd(8), чтобы FreeBSD получил доступ к сканеру.

Добавьте файл `saned.conf` в /usr/local/etc/devd/saned.conf со следующим содержимым:

```
notify 100 {
        match "system" "USB";
        match "subsystem" "INTERFACE";
        match "type" "ATTACH";
        match "cdev" "ugen[0-9].[0-9]";
        match "vendor" "0x03f0"; ①
        match "product" "0x8911"; ②
        action "chown -L cups:saned /dev/\$cdev && chmod -L 660 /dev/\$cdev";
```

```
};
```

① `vendor`: Это idVendor, полученный ранее при выполнении команды `usbconfig -d 3.2 dump_device_desc`.

② `product`: Это idProduct, полученный ранее при выполнении команды `usbconfig -d 3.2 dump_device_desc`.

После этого необходимо перезапустить devd(8), выполнив следующую команду:

```
# service devd restart
```

Бэкенды SANE включают scanimage(1), который можно использовать для вывода списка устройств и выполнения захвата изображения.

Выполните команду scanimage(1) с аргументом `-L` для вывода списка сканирующих устройств:

```
# scanimage -L
```

Вывод должен быть похож на следующий:

```
device `hpaio:/usb/Deskjet_1050_J410_series?serial=XXXXXXXXXXXXXX' is a Hewlett-
Packard Deskjet_1050_J410_series all-in-one
```

Если scanimage(1) не может идентифицировать сканер, появится следующее сообщение:

```
No scanners were identified. If you were expecting something different,
check that the scanner is plugged in, turned on and detected by the
sane-find-scanner tool (if appropriate). Please read the documentation
which came with this software (README, FAQ, manpages).
```

Как только scanimage(1) обнаружит сканер, настройка будет завершена, и сканер готов к использованию.

Чтобы активировать службу и запускать её при загрузке, выполните следующую команду:

```
# sysrc saned_enable=YES
```

В то время как scanimage(1) можно использовать для захвата изображения из командной строки, часто предпочтительнее применять графический интерфейс для сканирования изображений.

Таблица 22. Графические программы для сканирования

Имя	Лицензия	Пакет
skanlite	GPL 2.0	graphics/skanlite
GNOME Simple Scan	GPL 3.0	graphics/simple-scan
XSANE	GPL 2.0	graphics/xsane

Глава 10. Настройка ядра FreeBSD

10.1. Обзор

Ядро — это основа операционной системы FreeBSD. Оно отвечает за управление памятью, обеспечение контроля безопасности, работу с сетью, доступ к дискам и многое другое. Хотя большая часть FreeBSD динамически настраивается, некоторые пользователи могут захотеть настроить и скомпилировать собственное ядро.

Прочтите эту главу, чтобы узнать:

- Когда следует собирать собственный ядро.
- Как провести инвентаризацию оборудования.
- Как настроить файл конфигурации ядра.
- Как использовать файл конфигурации ядра для создания и сборки нового ядра.
- Как установить новое ядро.
- Как устранить неполадки, если что-то пойдет не так.

Все команды, приведённые в примерах этой главы, должны выполняться от имени пользователя `root`.

10.2. Зачем собирать собственное ядро?

Традиционно FreeBSD использовала монолитное ядро. Ядро представляло собой одну большую программу, поддерживало фиксированный список устройств, и для изменения его поведения необходимо было скомпилировать новое ядро, а затем перезагрузиться в него.

Сегодня большая часть функциональности ядра FreeBSD содержится в модулях, которые могут быть динамически загружены в ядро или выгружены из него по мере необходимости. Это позволяет работающему ядру немедленно адаптироваться к новому оборудованию и добавлять новую функциональность. Такое ядро называется модульным.

Время от времени всё ещё требуется выполнять статическую настройку ядра. Иногда необходимая функциональность настолько тесно связана с ядром, что её невозможно сделать динамически загружаемой. В некоторых средах с повышенными требованиями к безопасности запрещена загрузка и выгрузка модулей ядра, и требуется, чтобы только необходимая функциональность была статически скомпилирована в ядро.

Сборка собственного ядра часто является своего рода обрядом посвящения для опытных пользователей BSD. Этот процесс, хотя и отнимает много времени, может принести пользу системе FreeBSD. В отличие от ядра GENERIC, которое должно поддерживать широкий спектр оборудования, собственное ядро можно сократить до поддержки только аппаратного обеспечения данного компьютера. Это даёт ряд преимуществ, таких как:

- Более быстрая загрузка. Поскольку ядро будет проверять только оборудование, установленное в системе, время загрузки системы может сократиться.

- Снижение использования памяти. Собственное ядро часто потребляет меньше памяти, чем ядро GENERIC, за счёт исключения неиспользуемых функций и драйверов устройств. Это важно, поскольку код ядра постоянно находится в физической памяти, не позволяя использовать эту память приложениям. По этой причине собственное ядро полезно на системах с небольшим объёмом оперативной памяти.
- Дополнительная поддержка оборудования. Собственное ядро может добавить поддержку устройств, которые отсутствуют в ядре GENERIC.

При сборке собственного ядра важно учитывать, что нестандартные конфигурации тестируются менее тщательно, чем конфигурация GENERIC. Хотя настройка ядра может дать определённые преимущества, она также увеличивает риск возникновения проблем при сборке или во время работы. Пользовательские конфигурации ядра рекомендуются только опытным пользователям, у которых есть веская причина для внесения изменений и которые готовы при необходимости участвовать в процессе отладки.

Прежде чем собирать собственное ядро, стоит подумать о причине этого. Если требуется поддержка определённого оборудования, она может уже существовать в виде модуля.

Модули ядра находятся в /boot/kernel и могут быть динамически загружены в работающее ядро с помощью kldload(8). Большинство драйверов ядра имеют загружаемый модуль и страницу руководства. Например, драйвер беспроводной сети ath(4) содержит следующую информацию на своей странице руководства:

```
В качестве альтернативы, для загрузки драйвера в виде модуля при старте системы,
поместите следующую строку в loader.conf(5):

    if_ath_load="YES"
```

Добавление `if_ath_load="YES"` в /boot/loader.conf позволит динамически загрузить этот модуль во время загрузки системы.

В некоторых случаях связанный модуль отсутствует в /boot/kernel. Это в основном относится к определённым подсистемам.

10.3. Поиск информации об оборудовании системы

Прежде чем редактировать файл конфигурации ядра, рекомендуется составить перечень оборудования компьютера. На системе с двойной загрузкой этот перечень можно создать из другой операционной системы. Например, Диспетчер устройств Microsoft® содержит информацию об установленных устройствах.

Некоторые версии Microsoft® Windows® имеют значок "Система", который можно использовать для доступа к диспетчеру устройств.

Если FreeBSD — единственная установленная операционная система, используйте dmesg(8) для определения оборудования, обнаруженного и перечисленного во время загрузки.

Большинство драйверов устройств в FreeBSD имеют справочную страницу, в которой перечислено поддерживаемое оборудование. Например, следующие строки указывают, что драйвер psm(4) обнаружил мышь:

```
psm0: <PS/2 Mouse> irq 12 on atkbdc0
psm0: [GIANT-LOCKED]
psm0: [ITHREAD]
psm0: model Generic PS/2 mouse, device ID 0
```

Поскольку данное оборудование существует, этот драйвер не следует удалять из файла конфигурации собственного ядра.

Если вывод команды `dmesg` не отображает результаты загрузочного probing, вместо этого прочитайте содержимое файла /var/run/dmesg.boot.

Еще один инструмент для поиска оборудования — это pciconf(8), который предоставляет более подробный вывод. Например:

```
% pciconf -lv
ath0@pci0:3:0:0:        class=0x020000 card=0x058a1014 chip=0x1014168c rev=0x01 hdr
=0x00
    vendor     = 'Atheros Communications Inc.'
    device     = 'AR5212 Atheros AR5212 802.11abg wireless'
    class      = network
    subclass   = ethernet
```

Этот вывод показывает, что драйвер ath обнаружил беспроводное Ethernet-устройство.

Флаг `-k` утилиты man(1) может быть полезен для получения информации. Например, с его помощью можно вывести список страниц руководства, содержащих определённое название или марку устройства:

```
# man -k Atheros
ath(4)                  - Atheros IEEE 802.11 wireless network driver
ath_hal(4)              - Atheros Hardware Access Layer (HAL)
```

После составления списка оборудования обратитесь к нему, чтобы убедиться, что драйверы установленного оборудования не будут удалены при редактировании конфигурации собственного ядра.

10.4. Файл конфигурации

Для создания файла конфигурации собственного ядра и сборки кастомного ядра необходимо сначала установить полное дерево исходных кодов FreeBSD.

Если /usr/src/ не существует или пуст, исходный код не установлен. Исходный код можно

установить с помощью Git, следуя инструкциям в “Использование Git”.

После установки исходного кода ознакомьтесь с содержимым каталога /usr/src/sys. Этот каталог содержит несколько подкаталогов, включая те, которые соответствуют следующим поддерживаемым архитектурам: amd64, i386, powerpc и sparc64. Всё содержимое каталога конкретной архитектуры относится только к этой архитектуре, а остальной код является машинонезависимым и общим для всех платформ. Каждая поддерживаемая архитектура имеет подкаталог conf, который содержит файл конфигурации ядра GENERIC для данной архитектуры.

Не вносите изменения в файл GENERIC. Вместо этого скопируйте его под другим именем и редактируйте копию. По сложившейся практике имя файла должно состоять из заглавных букв. Если вы обслуживаете несколько машин FreeBSD с разным оборудованием, разумно назвать файл по имени хоста соответствующей машины. В следующем примере создаётся копия файла конфигурации GENERIC для архитектуры `amd64` с именем MYKERNEL:

```
# cd /usr/src/sys/amd64/conf
# cp GENERIC MYKERNEL
```

MYKERNEL теперь можно настроить с помощью любого текстового редактора, поддерживающего `ASCII`. Редактор по умолчанию — vi, но для новичков также установлен более простой редактор под названием ee.

Формат файла конфигурации ядра прост. Каждая строка содержит ключевое слово, представляющее устройство или подсистему, аргумент и краткое описание. Любой текст после `#` считается комментарием и игнорируется. Чтобы удалить поддержку устройства или подсистемы в ядре, поставьте `#` в начале строки, соответствующей этому устройству или подсистеме. Воздерживайтесь от добавления или удаления # для любой строки, которую вы не до конца понимаете.

Легко удалить поддержку устройства или опции и получить нерабочее ядро. Например, если драйвер ata(4) удалён из конфигурационного файла ядра, система, использующая драйверы дисков `ATA`, может не загрузиться. Если сомневаетесь — оставьте поддержку в ядре.

В дополнение к кратким описаниям, приведённым в этом файле, дополнительные описания содержатся в NOTES, которые можно найти в том же каталоге, что и GENERIC для данной архитектуры. Для архитектурно-независимых параметров обратитесь к /usr/src/sys/conf/NOTES.

После завершения настройки конфигурационного файла ядра сохраните резервную копию в расположении за пределами /usr/src.

Или можно сохранить файл конфигурации ядра в другом месте и создать символическую ссылку на него:

```
# cd /usr/src/sys/amd64/conf
```

```
# mkdir /root/kernels
# cp GENERIC /root/kernels/MYKERNEL
# ln -s /root/kernels/MYKERNEL
```

В конфигурационных файлах доступна директива `include`, которая позволяет включать содержимое другого конфигурационного файла в текущий. Это упрощает поддержку небольших изменений относительно существующего файла. Если требуется лишь несколько дополнительных параметров или драйверов, можно сохранять разницу относительно GENERIC, как показано в примере:

```
include GENERIC
ident MYKERNEL

options         IPFIREWALL
options         DUMMYNET
options         IPFIREWALL_DEFAULT_TO_ACCEPT
options         IPDIVERT
```

При использовании этого метода локальный конфигурационный файл отражает локальные отличия от ядра GENERIC. При выполнении обновлений новые функции, добавленные в GENERIC, также будут добавлены в локальное ядро, если они не запрещены явно с помощью `nooptions` или `nodevice`. Полный список директив конфигурации и их описания можно найти в config(5).

Чтобы создать файл, содержащий все доступные параметры, выполните следующую команду от имени `root`:

```
# cd /usr/src/sys/arch/conf && make LINT
```

10.5. Сборка и установка собственного ядра

После сохранения изменений в пользовательском конфигурационном файле исходный код ядра можно скомпилировать, выполнив следующие шаги:

Процедура: Сборка ядра

1. Перейдите в этот каталог:

   ```
   # cd /usr/src
   ```

2. Соберите новое ядро, указав имя файла конфигурации собственного ядра:

   ```
   # make buildkernel KERNCONF=MYKERNEL
   ```

3. Установите новое ядро, связанное с указанным файлом конфигурации ядра. Эта команда скопирует новое ядро в /boot/kernel/kernel, а старое ядро сохранит в /boot/kernel.old/kernel:

```
# make installkernel KERNCONF=MYKERNEL
```

4. Выключите систему и перезагрузитесь с новым ядром. Если возникнут проблемы, обратитесь к разделу Ядро не загружается.

По умолчанию при компиляции собственного ядра все модули ядра пересобираются. Чтобы ускорить обновление ядра или собрать только необходимые модули, отредактируйте /etc/make.conf перед началом сборки ядра.

Например, эта переменная задаёт список модулей для сборки вместо использования значения по умолчанию (сборка всех модулей):

```
MODULES_OVERRIDE = linux acpi
```

Или эта переменная указывает, какие модули исключить из процесса сборки:

```
WITHOUT_MODULES = linux acpi sound
```

Эта переменная указывает сборке пропустить модули ядра и компилировать только само ядро:

```
NO_MODULES = yes
```

Фреймворк портов включает драйверы и другие модули, которые зависят от интерфейсов ядра, например graphics/drm-kmod или emulators/virtualbox-ose-kmod. С помощью переменной PORTS_MODULES при каждой сборке ядра порты, содержащие модули ядра, пересобираются с учётом обновлённых исходников. Это гарантирует, что модуль ядра остаётся согласованным с самим ядром. Для максимальной совместимости дерево исходников ядра и дерево портов следует обновлять одновременно. Переменную PORTS_MODULES можно добавить в /etc/make.conf, чтобы все собираемые вами ядра также пересобирали эти модули.

```
PORTS_MODULES = drm-kmod virtualbox-ose-kmod
```

Доступны дополнительные переменные. Подробности смотрите в make.conf(5).

10.6. Если что-то пойдет не так

Существует четыре типа проблем, которые могут возникнуть при сборке собственного ядра:

`config` завершается с ошибкой

Если `config` завершается с ошибкой, он выводит номер строки, содержащей ошибку. Например, при получении следующего сообщения убедитесь, что строка 17 введена правильно, сравнив её с GENERIC или NOTES:

```
config: line 17: syntax error
```

`make` завершается с ошибкой

Если `make` завершается с ошибкой, обычно это связано с ошибкой в файле конфигурации ядра, которую `config` не смог обнаружить. Проверьте конфигурацию, и если проблема не очевидна, отправьте письмо в список рассылки Список рассылки, посвящённый вопросам и ответам пользователей FreeBSD, приложив файл конфигурации ядра.

Ядро не загружается

Если новый ядро не загружается или не распознаёт устройства, не паникуйте! К счастью, в FreeBSD есть отличный механизм восстановления после проблем с несовместимыми ядрами. Просто выберите ядро для загрузки в загрузчике FreeBSD. Это можно сделать при появлении меню загрузки системы, выбрав опцию «Escape to a loader prompt». В командной строке введите `boot kernel.old` или имя любого другого ядра, которое заведомо загружается правильно.

После загрузки с исправным ядром проверьте конфигурационный файл и попробуйте собрать его снова. Полезным ресурсом может быть /var/log/messages, где записываются сообщения ядра при каждой успешной загрузке. Также dmesg(8) выведет сообщения ядра текущей загрузки.

При устранении неполадок ядра обязательно сохраняйте копию работоспособного ядра, например, GENERIC. Это важно, потому что при каждой установке нового ядра файл kernel.old перезаписывается последним установленным ядром, которое может быть или не быть загружаемым. Как можно скорее переместите рабочее ядро, переименовав каталог, содержащий исправное ядро:

```
# mv /boot/kernel /boot/kernel.bad
# mv /boot/kernel.good /boot/kernel
```

Ядро работает, но ps(1) — нет

Если версия ядра отличается от той, с которой собраны системные утилиты, например, при установке ядра, собранного из исходников -CURRENT, на систему -RELEASE, многие команды для просмотра состояния системы, такие как ps(1) и vmstat(8), не будут работать. Чтобы исправить это, пересоберите и установите world, собранный из той же версии исходного дерева, что и ядро. Никогда не рекомендуется использовать версию ядра, отличную от остальной операционной системы.

Глава 11. Печать

Размещение информации на бумаге — это важная функция, несмотря на многочисленные попытки отказаться от неё. Печать состоит из двух основных компонентов. Данные должны быть переданы на принтер и представлены в форме, которую принтер может обработать.

11.1. Быстрый старт

Базовую печать можно быстро настроить. Принтер должен уметь печатать обычный текст ASCII. Для печати других типов файлов см. Фильтры.

1. Создайте каталог для хранения файлов во время печати:

```
# mkdir -p /var/spool/lpd/lp
# chown daemon:daemon /var/spool/lpd/lp
# chmod 770 /var/spool/lpd/lp
```

2. Как root, создайте файл /etc/printcap со следующим содержимым:

```
lp:\
lp=/dev/unlpt0:\  ①
sh:\
mx#0:\
sd=/var/spool/lpd/lp:\
lf=/var/log/lpd-errs:
```

① Эта строка предназначена для принтера, подключённого к порту USB.
Для принтера, подключённого к параллельному или «принтерному» порту, используйте:

```
:lp=/dev/lpt0:\
```

Для принтера, подключённого напрямую к сети, используйте:

```
:lp=:rm=network-printer-name:rp=raw:\
```

Замените *network-printer-name* на DNS имя хоста сетевого принтера.

3. Включите LPD, отредактировав файл /etc/rc.conf и добавив следующую строку:

```
lpd_enable="YES"
```

Запустите службу:

```
# service lpd start
Starting lpd.
```

4. Напечатайте тест:

```
# printf "1. This printer can print.\n2. This is the second line.\n" | lpr
```

Если обе строки начинаются не с левого края, а с отступом («ступеньками»), см. Предотвращение образования ступенек на простых текстовых принтерах.

Текстовые файлы теперь можно печатать с помощью `lpr`. Укажите имя файла в командной строке или передайте вывод напрямую в `lpr`.

```
% lpr textfile.txt
% ls -lh | lpr
```

11.2. Подключение принтеров

Принтеры подключаются к компьютерным системам различными способами. Небольшие настольные принтеры обычно подключаются непосредственно к USB-порту компьютера. Более старые принтеры подключаются к параллельному или «принтерному» порту. Некоторые принтеры подключаются напрямую к сети, что упрощает их совместное использование несколькими компьютерами. Некоторые принтеры используют редкое последовательное соединение через serial-порт.

FreeBSD может взаимодействовать со всеми типами принтеров.

USB

USB-принтеры можно подключить к любому свободному USB-порту компьютера.

Когда FreeBSD обнаруживает принтер USB, создаются два устройства: /dev/ulpt0 и /dev/unlpt0. Данные, отправленные на любое из этих устройств, будут переданы принтеру. После каждой печати ulpt0 сбрасывает порт USB. Сброс порта может вызывать проблемы с некоторыми принтерами, поэтому обычно используется устройство unlpt0. unlpt0 не сбрасывает порт USB вообще.

Параллельный порт (IEEE-1284)

Параллельный порт устройства это /dev/lpt0. Это устройство присутствует независимо от того, подключен принтер или нет, оно не определяется автоматически.

Производители в основном отказались от этих «устаревших» портов, и многие

компьютеры больше их не имеют. Для подключения параллельного принтера к порту USB можно использовать переходники. С таким переходником принтер можно использовать так, как если бы он был настоящим USB-принтером. Также для прямого подключения параллельных принтеров к сети можно использовать устройства, называемые *серверами печати*.

Последовательный порт (RS-232)

Последовательные порты — это ещё один устаревший интерфейс, который редко используется для принтеров, за исключением некоторых специализированных применений. Кабели, разъёмы и необходимая распиновка могут сильно различаться.

Для последовательных портов, встроенных в материнскую плату, имя последовательного устройства — /dev/cuau0 или /dev/cuau1. Также можно использовать последовательные адаптеры USB, которые будут отображаться как /dev/cuaU0.

Несколько параметров связи должны быть известны для взаимодействия с последовательным принтером. Наиболее важные из них — это *скорость передачи* или BPS (бит в секунду) и *чётность*. Значения могут различаться, но обычно последовательные принтеры используют скорость 9600 и отсутствие проверки на чётность.

Сеть

Сетевые принтеры подключаются напрямую к локальной компьютерной сети.

DNS имя хоста принтера должно быть известно. Если принтеру назначается динамический адрес через DHCP, DNS должен динамически обновляться, чтобы имя хоста всегда соответствовало правильному IP-адресу. Сетевые принтеры часто получают статические IP-адреса, чтобы избежать этой проблемы.

Большинство сетевых принтеров понимают задания печати, отправленные по протоколу LPD. Также можно указать имя очереди печати. Некоторые принтеры обрабатывают данные по-разному в зависимости от используемой очереди. Например, очередь `raw` печатает данные без изменений, а очередь `text` добавляет символы возврата каретки к обычному тексту.

Многие сетевые принтеры также могут печатать данные, отправленные напрямую на порт 9100.

11.2.1. Краткое содержание

Проводные сетевые подключения обычно проще всего настроить и обеспечивают наиболее быструю печать. Для прямого подключения к компьютеру предпочтительнее использовать USB из-за скорости и простоты. Параллельные подключения работают, но имеют ограничения по длине кабеля и скорости. Последовательные подключения сложнее настроить. Разводка кабелей различается в зависимости от модели, а параметры связи, такие как скорость передачи данных и биты чётности, добавляют сложности. К счастью, последовательные принтеры встречаются редко.

11.3. Языки описания страниц

Данные, отправляемые на принтер, должны быть на языке, который принтер может понять. Эти языки называются языками описания страниц или PDL (Page Description Languages).

`ASCII`

Обычный текст `ASCII` — это самый простой способ отправить данные на принтер. Символы соответствуют один к одному тому, что будет напечатано: символ `A` в данных напечатает `A` на странице. Доступно очень мало возможностей форматирования. Нет способа выбрать шрифт или пропорциональные интервалы. Вынужденная простота обычного `ASCII` означает, что текст можно печатать напрямую с компьютера с минимальным или вообще отсутствующим кодированием или преобразованием. Напечатанный вывод напрямую соответствует отправленным данным.

Некоторые недорогие принтеры не могут печатать обычный текст `ASCII`. Это усложняет их настройку, но обычно это всё равно возможно.

PostScript®

PostScript® почти противоположен `ASCII`. Вместо простого текста программа PostScript® представляет собой набор инструкций, которые рисуют итоговый документ. Можно использовать различные шрифты и графику. Однако за эту мощь приходится платить. Необходимо написать программу, которая рисует страницу. Обычно эта программа генерируется прикладным программным обеспечением, поэтому процесс остаётся невидимым для пользователя.

Недорогие принтеры иногда не поддерживают совместимость с PostScript® в целях экономии.

`PCL` **(Printer Command Language)**

`PCL` является расширением `ASCII`, добавляющим escape-последовательности для форматирования, выбора шрифтов и печати графики. Многие принтеры поддерживают `PCL5`. Некоторые поддерживают более новые версии `PCL6` или `PCLXL`. Эти более поздние версии являются надмножествами `PCL5` и могут обеспечивать более быструю печать.

Принтер на основе хоста

Производители могут снизить стоимость принтера, используя простой процессор и минимум памяти. Такие принтеры не способны печатать обычный текст. Вместо этого растровые изображения текста и графики создаются драйвером на главном компьютере и затем отправляются на принтер. Такие принтеры называются *host-based* (управляемыми хостом).

Взаимодействие между драйвером и принтером, подключённым к хосту, часто осуществляется по проприетарным или недокументированным протоколам, что делает их работоспособными только в наиболее распространённых операционных системах.

11.3.1. Преобразование PostScript® в другие PDL

Многие приложения из Коллекции портов и утилиты FreeBSD создают вывод в формате PostScript®. В следующей таблице представлены утилиты для преобразования этого формата в другие распространённые PDL:

Таблица 23. Языки PDL на выходе

PDL на выходе	Чем создаётся	Заметки
PCL или PCL5	print/ghostscript9-base	-sDEVICE=ljet4 для монохромной печати, -sDEVICE=cljet5 для цветной
PCLXL или PCL6	print/ghostscript9-base	-sDEVICE=pxlmono для монохромного режима, -sDEVICE=pxlcolor для цветного
ESC/P2	print/ghostscript9-base	-sDEVICE=uniprint
XQX	print/foo2zjs	

11.3.2. Краткое содержание

Для наиболее простой печати выберите принтер с поддержкой PostScript®. Принтеры, поддерживающие PCL, являются следующим предпочтительным вариантом. С помощью print/ghostscript9-base эти принтеры можно использовать так, как если бы они изначально понимали PostScript®. Принтеры, которые напрямую поддерживают PostScript® или PCL, почти всегда также поддерживают прямую печать обычных текстовых файлов в формате ASCII.

Строковые принтеры, такие как обычные струйные, обычно не поддерживают PostScript® или PCL. Они часто могут печатать простые текстовые файлы в формате ASCII. Пакет print/ghostscript9-base поддерживает языки описания страниц (PDL), используемые некоторыми из этих принтеров. Однако печать всей графической страницы на таких принтерах часто происходит очень медленно из-за большого объёма передаваемых и печатаемых данных.

Настроить принтеры, управляемые хостом, часто сложнее. Некоторые из них вообще нельзя использовать из-за проприетарных языков описания страниц (PDL). По возможности избегайте таких принтеров. Информацию о конкретных PDL, используемых различными моделями принтеров, можно найти на сайте http://www.openprinting.org/printers.

11.4. Прямая печать

Для периодической печати файлы можно отправлять непосредственно на устройство принтера без дополнительной настройки. Например, файл с именем sample.txt можно отправить на принтер с интерфейсом USB:

```
# cp sample.txt /dev/unlpt0
```

Прямая печать на сетевые принтеры зависит от возможностей принтера, но большинство из них принимают задания на печать через порт 9100, и для этого можно использовать nc(1). Чтобы напечатать тот же файл на принтере с DNS-именем хоста *netlaser*:

```
# nc netlaser 9100 < sample.txt
```

11.5. LPD (демон линейного принтера)

Печать файла в фоновом режиме называется спулингом. Спулер позволяет пользователю продолжать работу с другими программами на компьютере, не ожидая завершения медленного процесса печати.

FreeBSD включает спулер под названием lpd(8). Задания на печать отправляются с помощью lpr(1).

11.5.1. Начальная настройка

Создаётся каталог для хранения заданий печати, устанавливается владелец, а права доступа настраиваются так, чтобы другие пользователи не могли просматривать содержимое этих файлов:

```
# mkdir -p /var/spool/lpd/lp
# chown daemon:daemon /var/spool/lpd/lp
# chmod 770 /var/spool/lpd/lp
```

Принтеры определяются в файле /etc/printcap. Запись для каждого принтера включает такие данные, как имя, порт, к которому он подключен, и различные другие параметры. Создайте файл /etc/printcap со следующим содержимым:

```
lp:\                ①
    :lp=/dev/unlpt0:\   ②
    :sh:\           ③
    :mx#0:\         ④
    :sd=/var/spool/lpd/lp:\ ⑤
    :lf=/var/log/lpd-errs:  ⑥
```

① Имя этого принтера. lpr(1) отправляет задания на печать на принтер `lp`, если другой принтер не указан с помощью `-P`, поэтому принтер по умолчанию должен называться `lp`.

② Устройство, к которому подключен принтер. Замените эту строку на соответствующую для типа подключения, указанного здесь.

③ Подавить вывод титульной страницы в начале задания печати.

④ Не ограничивать максимальный размер задания на печать.

⑤ Путь к каталогу спулинга для этого принтера. Каждый принтер использует свой собственный каталог спулинга.

⑥ Файл журнала, в который будут записываться ошибки для этого принтера.

После создания /etc/printcap используйте chkprintcap(8) для проверки на ошибки:

```
# chkprintcap
```

Исправьте все выявленные проблемы перед продолжением.

Включите lpd(8) в /etc/rc.conf:

```
lpd_enable="YES"
```

Запустите службу:

```
# service lpd start
```

11.5.2. Печать с помощью lpr(1)

Документы отправляются на печать с помощью `lpr`. Файл для печати может быть указан в командной строке или передан в `lpr` через конвейер. Эти две команды эквивалентны и отправляют содержимое файла doc.txt на принтер по умолчанию:

```
% lpr doc.txt
% cat doc.txt | lpr
```

Принтеры можно выбрать с помощью `-P`. Для печати на принтере с именем *laser*:

```
% lpr -Plaser doc.txt
```

11.5.3. Фильтры

Примеры, показанные до этого, отправляли содержимое текстового файла напрямую на принтер. Пока принтер понимает содержимое этих файлов, вывод будет печататься корректно.

Некоторые принтеры не способны печатать обычный текст, а входной файл может даже не быть текстовым.

Фильтры позволяют преобразовывать или обрабатывать файлы. Обычно они используются для преобразования одного типа входных данных, например, простого текста, в формат,

который принтер может понять, например PostScript® или PCL. Фильтры также могут использоваться для предоставления дополнительных функций, таких как добавление номеров страниц или подсветка исходного кода для удобства чтения.

Обсуждаемые здесь фильтры являются *входными фильтрами* или *текстовыми фильтрами*. Эти фильтры преобразуют входящий файл в различные формы. Используйте su(1), чтобы стать root перед созданием файлов.

Фильтры указываются в /etc/printcap с идентификатором if=. Чтобы использовать /usr/local/libexec/lf2crlf в качестве фильтра, измените /etc/printcap следующим образом:

```
lp:\
    :lp=/dev/unlpt0:\
    :sh:\
    :mx#0:\
    :sd=/var/spool/lpd/lp:\
    :if=/usr/local/libexec/lf2crlf:\   ①
    :lf=/var/log/lpd-errs:
```

① if= определяет *входной фильтр*, который будет использоваться для входящего текста.

Символы обратной косой черты *продолжения строки* в конце строк в записях printcap показывают, что запись для принтера на самом деле представляет собой одну длинную строку с элементами, разделёнными двоеточиями. Предыдущий пример можно переписать в виде одной менее читаемой строки:

```
lp:lp=/dev/unlpt0:sh:mx#0:sd=/var/spool/lpd/lp:if=/usr/local/libexec/lf
2crlf:lf=/var/log/lpd-errs:
```

Предотвращение ступенчатости на простых текстовых принтерах

Типичные текстовые файлы FreeBSD содержат только один символ перевода строки в конце каждой строки. Эти строки будут выводиться "лесенкой" на стандартном принтере:

```
A printed file looks
                   like the steps of a staircase
                                              scattered by the wind
```

Фильтр может преобразовывать символы новой строки в возврат каретки и новую строку. Возврат каретки заставляет принтер возвращаться к левому краю после каждой строки. Создайте файл /usr/local/libexec/lf2crlf со следующим содержимым:

```
#!/bin/sh
CR=$'\r'
```

```
/usr/bin/sed -e "s/$/${CR}/g"
```

Установите разрешения и сделайте файл исполняемым:

```
# chmod 555 /usr/local/libexec/lf2crlf
```

Измените /etc/printcap, чтобы использовать новый фильтр:

```
:if=/usr/local/libexec/lf2crlf:\
```

Протестируйте фильтр, напечатав тот же текстовый файл. Возврат каретки приведет к тому, что каждая строка будет начинаться с левого края страницы.

Красивое оформление текста на PostScript® принтерах с помощью print/enscript

GNUEnscript преобразует обычные текстовые файлы в аккуратно отформатированный PostScript® для печати на принтерах PostScript®. Он добавляет номера страниц, переносит длинные строки и предоставляет множество других функций, чтобы сделать печать текстовых файлов более удобочитаемой. В зависимости от местного размера бумаги установите один из пакетов: print/enscript-letter или print/enscript-a4 из коллекции портов.

Создайте файл /usr/local/libexec/enscript со следующим содержимым:

```
#!/bin/sh
/usr/local/bin/enscript -o -
```

Установите разрешения и сделайте файл исполняемым:

```
# chmod 555 /usr/local/libexec/enscript
```

Измените /etc/printcap, чтобы использовать новый фильтр:

```
:if=/usr/local/libexec/enscript:\
```

Проверьте фильтр, напечатав простой текстовый файл.

Печать PostScript® на принтеры PCL

Многие программы создают документы в формате PostScript®. Однако недорогие принтеры часто поддерживают только простой текст или PCL. Этот фильтр преобразует файлы PostScript® в PCL перед отправкой на принтер.

Установите интерпретатор PostScript® Ghostscript, print/ghostscript9-base, из коллекции портов.

Создайте файл /usr/local/libexec/ps2pcl со следующим содержимым:

```
#!/bin/sh
/usr/local/bin/gs -dSAFER -dNOPAUSE -dBATCH -q -sDEVICE=ljet4 -sOutputFile=- -
```

Установите разрешения и сделайте файл исполняемым:

```
# chmod 555 /usr/local/libexec/ps2pcl
```

Поступающие на вход этого скрипта данные в формате PostScript® будут обработаны, преобразованы в PCL и отправлены на принтер.

Измените файл /etc/printcap, чтобы использовать новый входной фильтр:

```
:if=/usr/local/libexec/ps2pcl:\
```

Протестируйте фильтр, отправив в него небольшую программу PostScript®:

```
% printf "%%\!PS \n /Helvetica findfont 18 scalefont setfont \
72 432 moveto (PostScript printing successful.) show showpage \004" | lpr
```

Умные фильтры

Фильтр, который определяет тип входных данных и автоматически преобразует их в правильный формат для принтера, может быть очень удобным. Первые два символа файла PostScript® обычно %!. Фильтр может обнаружить эти два символа. Файлы PostScript® можно передавать на PostScript® принтер без изменений. Текстовые файлы можно преобразовать в PostScript® с помощью Enscript, как показано ранее. Создайте файл /usr/local/libexec/psif с таким содержимым:

```
#!/bin/sh
#
#  psif - Print PostScript or plain text on a PostScript printer
#
IFS="" read -r first_line
first_two_chars=`expr "$first_line" : '\(..\)'`

case "$first_two_chars" in
%!)
    # %! : PostScript job, print it.
    echo "$first_line" && cat && exit 0
    exit 2
    ;;
*)
    # otherwise, format with enscript
```

```
    ( echo "$first_line"; cat ) | /usr/local/bin/enscript -o - && exit 0
    exit 2
    ;;
esac
```

Установите разрешения и сделайте файл исполняемым:

```
# chmod 555 /usr/local/libexec/psif
```

Измените файл /etc/printcap, чтобы использовать новый входной фильтр:

```
:if=/usr/local/libexec/psif:\
```

Протестируйте фильтр, распечатав файлы в формате PostScript® и обычные текстовые файлы.

11.5.4. Множественные очереди

Записи в /etc/printcap фактически являются определениями *очередей*. Для одного принтера может быть несколько очередей. В сочетании с фильтрами множественные очереди предоставляют пользователям больше контроля над печатью их заданий.

В качестве примера рассмотрим сетевой PostScript® лазерный принтер в офисе. Большинство пользователей хотят печатать простой текст, но несколько продвинутых пользователей хотят иметь возможность печатать PostScript® файлы напрямую. Для одного и того же принтера можно создать две записи в /etc/printcap:

```
textprinter:\
    :lp=9100@officelaser:\
    :sh:\
    :mx#0:\
    :sd=/var/spool/lpd/textprinter:\
    :if=/usr/local/libexec/enscript:\
    :lf=/var/log/lpd-errs:

psprinter:\
    :lp=9100@officelaser:\
    :sh:\
    :mx#0:\
    :sd=/var/spool/lpd/psprinter:\
    :lf=/var/log/lpd-errs:
```

Документы, отправленные на `textprinter`, будут отформатированы фильтром /usr/local/libexec/enscript, показанным в предыдущем примере. Опытные пользователи могут печатать PostScript® файлы на `psprinter`, где фильтрация не выполняется.

Этот метод с несколькими очередями может быть использован для предоставления прямого доступа к различным функциям принтера. Принтер с дуплексом может использовать две очереди: одну для обычной односторонней печати и другую с фильтром, который отправляет последовательность команд для включения двусторонней печати, а затем передаёт входящий файл.

11.5.5. Мониторинг и управление печатью

Доступно несколько утилит для мониторинга заданий печати, проверки и управления работой принтера.

lpq(1)

lpq(1) отображает состояние заданий печати пользователя. Задания печати других пользователей не показываются.

Показать ожидающие задания текущего пользователя на одном принтере:

```
% lpq -Plp
Rank   Owner      Job  Files                                 Total Size
1st    jsmith     0    (standard input)                      12792 bytes
```

Показать ожидающие задания текущего пользователя на всех принтерах:

```
% lpq -a
lp:
Rank   Owner      Job  Files                                 Total Size
1st    jsmith     1    (standard input)                      27320 bytes

laser:
Rank   Owner      Job  Files                                 Total Size
1st    jsmith     287  (standard input)                      22443 bytes
```

lprm(1)

lprm(1) используется для удаления заданий печати. Обычные пользователи могут удалять только свои собственные задания. `root` может удалять любые или все задания.

Удалить все ожидающие задания из принтера:

```
# lprm -Plp -
dfA002smithy dequeued
cfA002smithy dequeued
dfA003smithy dequeued
cfA003smithy dequeued
dfA004smithy dequeued
cfA004smithy dequeued
```

Удалить отдельное задание из принтера. Для определения номера задания используется lpq(1).

```
% lpq
Rank   Owner      Job  Files                                 Total Size
1st    jsmith     5    (standard input)                      12188 bytes

% lprm -Plp 5
dfA005smithy dequeued
cfA005smithy dequeued
```

lpc(8)

lpc(8) используется для проверки и изменения состояния принтеров. После `lpc` указывается команда и, при необходимости, имя принтера. Вместо имени конкретного принтера можно использовать `all`, и тогда команда будет применена ко всем принтерам. Обычные пользователи могут просматривать статус с помощью lpc(8). Только пользователь `root` может использовать команды, изменяющие состояние принтеров.

Показать состояние всех принтеров:

```
% lpc status all
lp:
    queuing is enabled
    printing is enabled
    1 entry in spool area
    printer idle
laser:
    queuing is enabled
    printing is enabled
    1 entry in spool area
    waiting for laser to come up
```

Запретить принтеру принимать новые задания, затем снова разрешить прием новых заданий:

```
# lpc disable lp
lp:
    queuing disabled
# lpc enable lp
lp:
    queuing enabled
```

Приостановить печать, но продолжать принимать новые задания. Затем возобновить печать:

```
# lpc stop lp
lp:
    printing disabled
# lpc start lp
lp:
    printing enabled
    daemon started
```

Перезапустить принтер после возникновения ошибки:

```
# lpc restart lp
lp:
    no daemon to abort
    printing enabled
    daemon restarted
```

Отключите очередь печати и запретите печать, добавив сообщение для пользователей с объяснением проблемы:

```
# lpc down lp Repair parts will arrive on Monday
lp:
    printer and queuing disabled
    status message is now: Repair parts will arrive on Monday
```

Включить принтер, который выключен:

```
# lpc up lp
lp:
    printing enabled
    daemon started
```

См. lpc(8) для получения дополнительных команд и параметров.

11.5.6. Общие принтеры

Принтеры часто используются совместно несколькими пользователями в офисах и учебных заведениях. Для удобства совместного использования предусмотрены дополнительные функции.

Aliases

Имя принтера задается в первой строке записи в /etc/printcap. Дополнительные имена или *псевдонимы* могут быть добавлены после этого имени. Псевдонимы отделяются от имени и друг от друга вертикальными чертами:

```
lp|repairsprinter|salesprinter:\
```

Алиасы могут использоваться вместо имени принтера. Например, сотрудники отдела продаж печатают на свой принтер с помощью

```
% lpr -Psalesprinter sales-report.txt
```

Пользователи отдела ремонта печатают на *своём* принтере с помощью

```
% lpr -Prepairsprinter repairs-report.txt
```

Все документы печатаются на этом единственном принтере. Когда отдел продаж разрастётся настолько, что ему понадобится собственный принтер, псевдоним можно удалить из записи общего принтера и использовать как имя нового принтера. Пользователи в обоих отделах продолжают использовать те же команды, но документы отдела продаж отправляются на новый принтер.

Страницы заголовков

Пользователям может быть сложно найти свои документы в стопке страниц, напечатанных на занятом общем принтере. Для решения этой проблемы были созданы *титульные страницы*. Перед каждой задачей печати выводится титульная страница с именем пользователя и названием документа. Эти страницы также иногда называют *баннерами* или *разделителями*.

Включение титульных страниц различается в зависимости от того, подключен ли принтер напрямую к компьютеру с помощью кабеля USB, кабеля параллельного или последовательного порта или удалённо через сеть.

Заголовочные страницы на непосредственно подключённых принтерах включаются путем удаления строки `:sh:\` (Suppress Header) из записи в файле /etc/printcap. Эти заголовочные страницы используют только символы перевода строки для новых строк. Некоторым принтерам может потребоваться фильтр /usr/share/examples/printing/hpif, чтобы предотвратить ступенчатый текст. Фильтр настраивает принтеры PCL на печать как возврата каретки, так и перевода строки при получении символа перевода строки.

Заглавные страницы для сетевых принтеров должны быть настроены на самом принтере. Записи заглавных страниц в /etc/printcap игнорируются. Настройки обычно доступны через панель управления принтера или веб-интерфейс конфигурации, который можно открыть в браузере.

11.5.7. Список литературы

Примеры файлов: /usr/share/examples/printing/.

Руководство по системе печати 4.3BSD, /usr/share/doc/smm/07.lpd/paper.ascii.gz.

Страницы Справочника: printcap(5), lpd(8), lpr(1), lpc(8), lprm(1), lpq(1).

11.6. Другие системы печати

Несколько других систем печати доступны в дополнение к встроенной lpd(8). Эти системы обеспечивают поддержку других протоколов или дополнительные возможности.

11.6.1. CUPS (Common UNIX® Printing System)

CUPS — это популярная система печати, доступная во многих операционных системах. Использование CUPS в FreeBSD описано в отдельной статье: CUPS

11.6.2. HPLIP

Компания Hewlett Packard предоставляет систему печати, поддерживающую многие их струйные и лазерные принтеры. Порт находится в print/hplip. Основная веб-страница расположена по адресу https://developers.hp.com/hp-linux-imaging-and-printing. Порт обрабатывает все детали установки в FreeBSD. Информация по настройке доступна на странице https://developers.hp.com/hp-linux-imaging-and-printing/install.

11.6.3. LPRng

LPRng был разработан как улучшенная альтернатива lpd(8). Порт находится в sysutils/LPRng. Подробности и документацию можно найти на https://lprng.sourceforge.net/.

Глава 12. Двоичная совместимость с Linux

12.1. Обзор

FreeBSD предоставляет **опциональную** двоичную совместимость с Linux®, часто называемую Linuxulator, что позволяет пользователям устанавливать и запускать неизменённые Linux-бинарники. Это доступно для архитектур x86 (как 32, так и 64 бит) и AArch64. Некоторые специфичные для Linux функции операционной системы пока не поддерживаются; в основном это касается функциональности, связанной с оборудованием или управлением системой, такой как cgroups или пространства имён.

Прежде чем читать эту главу, необходимо:

- Знать, как установить дополнительное стороннее программное обеспечение.

Прочтите эту главу, чтобы узнать:

- Как включить двоичную совместимость с Linux в системе FreeBSD.
- Как установить дополнительные общие библиотеки Linux.
- Как установить Linux-приложения в системе FreeBSD.
- Как реализована совместимость с Linux в FreeBSD.

12.2. Настройка бинарной совместимости с Linux

По умолчанию бинарная совместимость с linux(4) не включена.

Чтобы включить ABI Linux при загрузке, выполните следующую команду:

```
# sysrc linux_enable="YES"
```

После включения его можно запустить без перезагрузки, выполнив следующую команду:

```
# service linux start
```

Этого достаточно для работы статически связанных Linux-библиотек.

Сервис Linux загрузит необходимые модули ядра и смонтирует файловые системы, ожидаемые Linux-приложениями, в /compat/linux. Их можно запускать так же, как и родные FreeBSD-приложения; они ведут себя почти так же, как родные процессы, и их можно трассировать и отлаживать обычными способами.

Текущее содержимое /compat/linux можно проверить, выполнив следующую команду:

```
# ls -l /compat/linux/
```

Вывод должен быть похож на следующий:

```
total 1
dr-xr-xr-x  13 root  wheel  512 Apr 11 19:12 dev
dr-xr-xr-x   1 root  wheel    0 Apr 11 21:03 proc
dr-xr-xr-x   1 root  wheel    0 Apr 11 21:03 sys
```

12.3. Пользовательские окружения Linux

Программное обеспечение Linux требует не только ABI для работы. Для запуска программ Linux необходимо сначала установить пользовательское пространство Linux.

Если требуется только запустить какое-либо программное обеспечение, уже включённое в дерево портов, его можно установить через менеджер пакетов, и pkg(8) автоматически настроит необходимую пользовательскую среду Linux.

Например, чтобы установить Sublime Text 4 вместе со всеми необходимыми библиотеками Linux, выполните следующую команду:

```
# pkg install linux-sublime-text4
```

12.3.1. Базовая система Rocky Linux из пакетов FreeBSD

Для установки пользовательской среды Rocky Linux 9 выполните следующую команду:

```
# pkg install linux_base-rl9
```

Пакет emulators/linux_base-rl9 разместит базовую систему, основанную на Rocky Linux 9, в /compat/linux.

После установки пакета содержимое /compat/linux можно проверить, выполнив следующую команду, чтобы убедиться, что пользовательская среда Rocky Linux установлена:

```
# ls -l /compat/linux/
```

Вывод должен быть похож на следующий:

```
total 36
drwxr-xr-x   2 root wheel  512 Oct  9 17:28 afs
lrwxr-xr-x   1 root wheel    7 May 16  2022 bin -> usr/bin
drwxr-xr-x   3 root wheel  512 Oct  9 17:28 dev
drwxr-xr-x  24 root wheel 1536 Oct  9 17:28 etc
lrwxr-xr-x   1 root wheel    7 May 16  2022 lib -> usr/lib
```

```
lrwxr-xr-x   1 root wheel    9 May 16  2022 lib64 -> usr/lib64
drwxr-xr-x   2 root wheel  512 Oct  9 17:28 opt
drwxr-xr-x   2 root wheel  512 Oct  9 17:28 proc
lrwxr-xr-x   1 root wheel    8 Oct  1 03:11 run -> /var/run
lrwxr-xr-x   1 root wheel    8 May 16  2022 sbin -> usr/sbin
drwxr-xr-x   2 root wheel  512 Oct  9 17:28 srv
drwxr-xr-x   2 root wheel  512 Oct  9 17:28 sys
drwxr-xr-x   8 root wheel  512 Oct  9 17:28 usr
drwxr-xr-x  16 root wheel  512 Oct  9 17:28 var
```

12.3.2. Базовая система CentOS из пакетов FreeBSD

Пакет emulators/linux_base-c7 устарел после прекращения поддержки вышестоящего проекта. Это означает, что emulators/linux_base-c7 больше не будет получать обновления безопасности. Рекомендуется использовать базовую систему Rocky Linux, если не требуется совместимость с 32-битными системами.

Для установки пользовательской среды CentOS выполните следующую команду:

```
# pkg install linux_base-c7
```

Пакет emulators/linux_base-c7 разместит базовую систему, основанную на CentOS 7, в каталоге `/compat/linux`.

После установки пакета содержимое /compat/linux можно проверить, выполнив следующую команду, чтобы убедиться, что пользовательская среда CentOS установлена:

```
# ls -l /compat/linux/
```

Вывод должен быть похож на следующий:

```
total 30
lrwxr-xr-x   1 root  wheel    7 Apr 11  2018 bin -> usr/bin
drwxr-xr-x  13 root  wheel  512 Apr 11 21:10 dev
drwxr-xr-x  25 root  wheel   64 Apr 11 21:10 etc
lrwxr-xr-x   1 root  wheel    7 Apr 11  2018 lib -> usr/lib
lrwxr-xr-x   1 root  wheel    9 Apr 11  2018 lib64 -> usr/lib64
drwxr-xr-x   2 root  wheel    2 Apr 11 21:10 opt
dr-xr-xr-x   1 root  wheel    0 Apr 11 21:25 proc
lrwxr-xr-x   1 root  wheel    8 Feb 18 02:10 run -> /var/run
lrwxr-xr-x   1 root  wheel    8 Apr 11  2018 sbin -> usr/sbin
drwxr-xr-x   2 root  wheel    2 Apr 11 21:10 srv
dr-xr-xr-x   1 root  wheel    0 Apr 11 21:25 sys
drwxr-xr-x   8 root  wheel    9 Apr 11 21:10 usr
drwxr-xr-x  16 root  wheel   17 Apr 11 21:10 var
```

12.3.3. Debian / Ubuntu Базовая система с debootstrap

Альтернативный способ предоставления общих библиотек Linux — использование sysutils/debootstrap. Это имеет преимущество в виде предоставления полного дистрибутива Debian или Ubuntu.

Чтобы установить debootstrap, выполните следующую команду:

```
# pkg install debootstrap
```

Для работы debootstrap(8) требуется включённый linux(4) ABI. После его активации выполните следующую команду для установки Ubuntu или Debian в /compat/ubuntu:

```
# debootstrap focal /compat/ubuntu
```

Хотя технически возможно установить в /compat/linux, это не рекомендуется из-за возможных конфликтов с пакетами на основе CentOS. Вместо этого используйте имя каталога, производное от названия дистрибутива или версии, например, /compat/ubuntu.

Вывод должен быть похож на следующий:

```
I: Retrieving InRelease
I: Checking Release signature
I: Valid Release signature (key id F6ECB3762474EDA9D21B7022871920D1991BC93C)
I: Retrieving Packages
I: Validating Packages
I: Resolving dependencies of required packages...
I: Resolving dependencies of base packages...
I: Checking component main on http://archive.ubuntu.com/ubuntu...
[...]
I: Configuring console-setup...
I: Configuring kbd...
I: Configuring ubuntu-minimal...
I: Configuring libc-bin...
I: Configuring ca-certificates...
I: Base system installed successfully.
```

Затем настройте точки монтирования в /etc/fstab.

Если содержимое домашнего каталога должно быть общим и необходимо иметь возможность запускать приложения X11, /home и /tmp следует подключить в области совместимости linux с использованием nullfs(5) для loopback (обратной петли).

Следующий пример можно добавить в /etc/fstab:

```
# Device        Mountpoint              FStype          Options
Dump    Pass#
devfs           /compat/ubuntu/dev      devfs           rw,late
0       0
tmpfs           /compat/ubuntu/dev/shm  tmpfs
rw,late,size=1g,mode=1777    0       0
fdescfs         /compat/ubuntu/dev/fd   fdescfs
rw,late,linrdlnk             0       0
linprocfs       /compat/ubuntu/proc     linprocfs       rw,late
0       0
linsysfs        /compat/ubuntu/sys      linsysfs        rw,late
0       0
/tmp            /compat/ubuntu/tmp      nullfs          rw,late
0       0
/home           /compat/ubuntu/home     nullfs          rw,late
0       0
```

Затем выполните mount(8):

```
# mount -al
```

Для доступа к системе с использованием chroot(8) выполните следующую команду:

```
# chroot /compat/ubuntu /bin/bash
```

Тогда можно выполнить uname(1), чтобы проверить окружение Linux:

```
# uname -s -r -m
```

Вывод должен быть похож на следующий:

```
Linux 5.15.0 x86_64
```

Оказавшись в chroot, система ведёт себя как при обычной установке Ubuntu. Хотя systemd не работает, команда service(8) функционирует в обычном режиме.

Чтобы добавить отсутствующие репозитории пакетов по умолчанию, отредактируйте файл /compat/ubuntu/etc/apt/sources.list.

Для amd64 можно использовать следующий пример:

```
deb http://archive.ubuntu.com/ubuntu jammy main universe restricted
multiverse
deb http://security.ubuntu.com/ubuntu/ jammy-security universe
```

```
multiverse restricted main
deb http://archive.ubuntu.com/ubuntu jammy-backports universe
multiverse restricted main
deb http://archive.ubuntu.com/ubuntu jammy-updates universe multiverse
restricted main
```

Для arm64 можно использовать следующий пример:

```
deb http://ports.ubuntu.com/ubuntu-ports bionic main universe
restricted multiverse
```

12.4. Сложные темы

Список всех параметров sysctl(8), связанных с Linux, можно найти в linux(4).

Некоторые приложения требуют подключения определённых файловых систем.

Обычно это обрабатывается скриптом /etc/rc.d/linux, но можно отключить при загрузке, выполнив следующую команду:

```
sysrc linux_mounts_enable="NO"
```

Файловые системы, смонтированные скриптом rc, не будут работать для Linux-процессов внутри chroot или клетки. При необходимости настройте их в /etc/fstab:

```
devfs      /compat/linux/dev      devfs      rw,late                    0  0
tmpfs      /compat/linux/dev/shm  tmpfs      rw,late,size=1g,mode=1777  0  0
fdescfs    /compat/linux/dev/fd   fdescfs    rw,late,linrdlnk           0  0
linprocfs  /compat/linux/proc     linprocfs  rw,late                    0  0
linsysfs   /compat/linux/sys      linsysfs   rw,late                    0  0
```

Поскольку уровень двоичной совместимости с Linux получил поддержку выполнения как 32-, так и 64-битных бинарных файлов Linux, стало невозможно статически линковать функциональность эмуляции в пользовательское ядро.

12.4.1. Установка дополнительных библиотек вручную

Для подкаталогов базовой системы, созданных с помощью debootstrap(8), используйте приведённые выше инструкции.

Если приложение Linux жалуется на отсутствие общих библиотек после настройки совместимости с двоичными файлами Linux, определите, какие общие библиотеки нужны двоичному файлу Linux, и установите их вручную.

С системы Linux с той же архитектурой CPU можно использовать `ldd` для определения того,

какие разделяемые библиотеки необходимы приложению.

Например, чтобы проверить, какие разделяемые библиотеки требуются для `linuxdoom`, выполните следующую команду в системе Linux, где установлен Doom:

```
% ldd linuxdoom
```

Вывод должен быть похож на следующий:

```
libXt.so.3 (DLL Jump 3.1) => /usr/X11/lib/libXt.so.3.1.0
libX11.so.3 (DLL Jump 3.1) => /usr/X11/lib/libX11.so.3.1.0
libc.so.4 (DLL Jump 4.5pl26) => /lib/libc.so.4.6.29
```

Затем скопируйте все файлы из последнего столбца вывода с системы Linux в /compat/linux на системе FreeBSD. После копирования создайте символические ссылки на имена из первого столбца.

Этот пример приведёт к следующим файлам в системе FreeBSD:

```
/compat/linux/usr/X11/lib/libXt.so.3.1.0
/compat/linux/usr/X11/lib/libXt.so.3 -> libXt.so.3.1.0
/compat/linux/usr/X11/lib/libX11.so.3.1.0
/compat/linux/usr/X11/lib/libX11.so.3 -> libX11.so.3.1.0
/compat/linux/lib/libc.so.4.6.29
/compat/linux/lib/libc.so.4 -> libc.so.4.6.29
```

Если уже существует общая библиотека Linux с соответствующим номером старшей версии (первый столбец вывода `ldd`), её не нужно копировать в файл, указанный в последнем столбце, так как существующая библиотека должна работать. Однако рекомендуется скопировать общую библиотеку, если она является более новой версией. Старую версию можно удалить, при условии что символическая ссылка указывает на новую.

Например, эти библиотеки уже существуют в системе FreeBSD:

```
/compat/linux/lib/libc.so.4.6.27
/compat/linux/lib/libc.so.4 -> libc.so.4.6.27
```

и `ldd` показывает, что для бинарного файла требуется более новая версия:

```
libc.so.4 (DLL Jump 4.5pl26) -> libc.so.4.6.29
```

Поскольку существующая библиотека устарела всего на одну или две версии в последней цифре, программа должна по-прежнему работать с немного более старой версией. Однако можно безопасно заменить существующий libc.so на более новую версию:

```
/compat/linux/lib/libc.so.4.6.29
/compat/linux/lib/libc.so.4 -> libc.so.4.6.29
```

Как правило, необходимость искать разделяемые библиотеки, от которых зависят Linux-бинарники, возникает только в первые несколько раз при установке Linux-программ на FreeBSD. Через некоторое время в системе накопится достаточный набор разделяемых библиотек Linux, что позволит запускать вновь устанавливаемые Linux-бинарники без дополнительных усилий.

12.4.2. Маркировка ELF-бинарников Linux

Ядро FreeBSD использует несколько методов для определения, является ли запускаемый двоичный файл Linux-программой: оно проверяет бренд в заголовке ELF-файла, ищет известные пути к ELF-интерпретаторам и проверяет ELF-заметки; наконец, по умолчанию небрендированные ELF-исполняемые файлы в любом случае считаются Linux-программами.

Если все эти методы не сработают, попытка выполнить двоичный файл может привести к сообщению об ошибке:

```
% ./my-linux-elf-binary
```

Вывод должен быть похож на следующий:

```
ELF binary type not known
Abort
```

Чтобы помочь ядру FreeBSD отличить ELF-программу FreeBSD от Linux-программы, используйте brandelf(1):

```
% brandelf -t Linux my-linux-elf-binary
```

12.4.3. Установка приложения на основе Linux RPM

Для установки приложения на основе RPM в Linux сначала установите пакет archivers/rpm4 или порт. После установки пользователь `root` может использовать следующую команду для установки файла .rpm:

```
# cd /compat/linux
# rpm2cpio < /path/to/linux.archive.rpm | cpio -id
```

При необходимости выполните `brandelf` для установленных ELF-бинарников. Заметим, что это сделает невозможной чистую деинсталляцию.

12.4.4. Настройка преобразователя имён хостов

Если DNS не работает или появляется эта ошибка:

```
resolv+: "bind" is an invalid keyword resolv+:
"hosts" is an invalid keyword
```

Настройте файл /compat/linux/etc/host.conf следующим образом:

```
order hosts, bind
multi on
```

Это указывает, что сначала выполняется поиск в /etc/hosts, а затем в DNS. Если /compat/linux/etc/host.conf отсутствует, Linux-приложения используют /etc/host.conf основной системы, но выдают ошибку, так как этот файл отсутствует в FreeBSD. Удалите `bind`, если сервер имён не настроен в /etc/resolv.conf.

12.4.5. Разное

Дополнительная информация о работе двоичной совместимости с Linux® доступна в статье Эмуляция Linux в FreeBSD.

Глава 13. WINE

13.1. Обзор

WINE, что расшифровывается как Wine Is Not an Emulator (WINE — это не эмулятор), технически является программным уровнем трансляции. Он позволяет устанавливать и запускать программное обеспечение, написанное для Windows®, в системах FreeBSD (и других).

Он работает, перехватывая системные вызовы, или запросы от программного обеспечения к операционной системе, и преобразует их из вызовов Windows® в вызовы, которые понимает FreeBSD. Он также преобразует любые ответы, если это необходимо, в то, что ожидает программное обеспечение Windows®. Таким образом, в некотором смысле он *эмулирует* среду Windows®, предоставляя многие из ресурсов, которые ожидают приложения Windows®.

Однако это не эмулятор в традиционном смысле. Многие из этих решений работают, создавая полностью отдельный компьютер с использованием программных процессов вместо аппаратного обеспечения. Виртуализация (например, предоставляемая портом `emulators/qemu`) работает именно так. Одно из преимуществ этого подхода — возможность установить полную версию соответствующей ОС в эмулятор. Это означает, что среда для приложений не будет отличаться от реальной машины, и высока вероятность, что всё будет работать. Недостаток этого подхода заключается в том, что программное обеспечение, имитирующее аппаратное, по своей природе медленнее реального оборудования. Компьютер, созданный программно (называемый *гостевым*), требует ресурсов от реальной машины (называемой *хостовой*) и удерживает эти ресурсы всё время своей работы.

С другой стороны, проект WINE гораздо менее требователен к ресурсам системы. Он преобразует системные вызовы на лету, поэтому, хотя и сложно достичь скорости реального компьютера с Windows®, он может приблизиться к ней. Однако WINE вынужден успевать за движущейся целью в плане всех различных системных вызовов и другой функциональности, которую необходимо поддерживать. В результате некоторые приложения могут работать в WINE не так, как ожидается, не работать вовсе или даже не устанавливаться.

В конечном итоге, WINE предоставляет ещё один вариант для запуска определённого программного обеспечения Windows® на FreeBSD. Он всегда может служить первым вариантом, который в случае успеха обеспечивает хороший опыт работы без излишнего потребления ресурсов хостовой системы FreeBSD.

Эта глава расскажет о следующем:

- Как установить WINE в системе FreeBSD.
- Как работает WINE и чем он отличается от других альтернатив, таких как виртуализация.
- Как настроить WINE под конкретные потребности некоторых приложений.
- Как установить графические помощники для WINE.

- Общие советы и решения по использованию WINE в FreeBSD.
- Соображения по использованию WINE в FreeBSD в условиях многопользовательской среды.

Прежде чем читать эту главу, полезно:

- Понимать основы UNIX® и FreeBSD.
- Знать, как установить FreeBSD.
- Уметь настраивать сетевое соединение, как описано в разделе по расширенной настройке сети.
- Знать, как установить дополнительное стороннее программное обеспечение.

13.2. Обзор и основные концепции WINE

WINE — это сложная система, поэтому перед запуском на FreeBSD стоит разобраться, что она из себя представляет и как работает.

13.2.1. Что такое WINE?

Как упоминалось в Обзоре этой главы, WINE — это уровень совместимости, который позволяет приложениям Windows® работать в других операционных системах. Теоретически это означает, что такие программы должны работать в системах, подобных FreeBSD, macOS и Android.

Когда WINE запускает исполняемый файл Windows®, происходят две вещи:

- Во-первых, WINE реализует среду, имитирующую различные версии Windows®. Например, если приложение запрашивает доступ к ресурсу, такому как оперативная память, WINE предоставляет интерфейс памяти, который выглядит и ведёт себя (с точки зрения приложения) как Windows®.
- Затем, когда приложение использует этот интерфейс, WINE обрабатывает запрос на выделение памяти и преобразует его в формат, совместимый с хост-системой. Аналогично, когда приложение запрашивает эти данные, WINE обеспечивает их получение из хост-системы и передачу обратно в приложение Windows®.

13.2.2. WINE и система FreeBSD

Установка WINE в системе FreeBSD включает несколько различных компонентов:

- Приложения FreeBSD для таких задач, как запуск исполняемых файлов Windows®, настройка подсистемы WINE или компиляция программ с поддержкой WINE.
- Большое количество библиотек, реализующих основные функции Windows® (например, /lib/wine/api-ms-core-memory-l1-1-1.dll.so, которая является частью упомянутого интерфейса памяти).
- Ряд исполняемых файлов Windows®, которые являются (или имитируют) распространённые утилиты (например, /lib/wine/notepad.exe.so, предоставляющий

стандартный текстовый редактор Windows®).

- Дополнительные ресурсы Windows®, в частности шрифты (например, шрифт Tahoma, который хранится в share/wine/fonts/tahoma.ttf в корневом каталоге установки).

13.2.3. Графические и текстовые программы/терминальные программы в WINE

Как операционная система, где терминальные утилиты являются «приложениями первого класса», естественно предположить, что WINE будет иметь обширную поддержку текстовых программ. Однако большинство приложений для Windows®, особенно наиболее популярные, разработаны с расчётом на графический интерфейс пользователя (GUI). Поэтому утилиты WINE по умолчанию настроены на запуск графических программ.

Однако доступны три метода для запуска этих так называемых программ с интерфейсом пользователя в консоли (Console User Interface, CUI):

- Подход *Bare Streams* будет выводить данные напрямую в стандартный вывод.
- Утилита *wineconsole* может использоваться с бэкендом *user* или *curses* для использования некоторых улучшений, которые система WINE предоставляет для CUI-приложений.

Эти методы подробно описаны на вики WINE.

13.2.4. Производные проекты WINE

WINE сам по себе является зрелым проектом с открытым исходным кодом, поэтому неудивительно, что он используется в качестве основы для более сложных решений.

Коммерческие реализации WINE

Некоторые компании взяли WINE и сделали его основой своих собственных проприетарных продуктов (лицензия LGPL WINE позволяет это). Два самых известных из них следующие:

- Codeweavers CrossOver

Это решение предлагает упрощённую установку WINE "в один клик", включающую дополнительные улучшения и оптимизации (хотя компания вносит многие из них обратно в основной проект WINE). Codeweavers сосредоточены на том, чтобы популярные приложения легко устанавливались и работали без проблем.

Хотя компания когда-то выпускала нативную версию своего решения CrossOver для FreeBSD, похоже, что она давно заброшена. Хотя некоторые ресурсы (например, выделенный форум) всё ещё доступны, на них также давно не наблюдается активности.

- Steam Proton

Игровая компания Steam также использует WINE, чтобы позволить играм для Windows® устанавливаться и запускаться на других системах. Основной целью являются системы на базе Linux, но также существует некоторая поддержка macOS.

Хотя Steam не предлагает нативный клиент для FreeBSD, существует несколько вариантов использования клиента для Linux® с помощью слоя совместимости с Linux в FreeBSD.

Сопутствующие программы WINE

В дополнение к проприетарным решениям, другие проекты выпустили приложения, предназначенные для работы в связке со стандартной, открытой версией WINE. Их цели могут варьироваться от упрощения установки до предоставления удобных способов установки популярного программного обеспечения.

Эти решения подробно рассмотрены в разделе Графические интерфейсы управления и включают следующее:

- winetricks
- Mizutamari

13.2.5. Альтернативы WINE

Для пользователей FreeBSD существуют следующие альтернативы использованию WINE:

- Двойная загрузка: Простой вариант — запускать нужные приложения Windows® нативно в этой ОС. Конечно, это означает выход из FreeBSD для загрузки Windows®, поэтому такой метод не подходит, если требуется одновременный доступ к программам в обеих системах.
- Виртуальные машины: Виртуальные машины (ВМ), как упоминалось ранее в этой главе, представляют собой программные процессы, эмулирующие полные наборы аппаратного обеспечения, на которых могут быть установлены и запущены дополнительные операционные системы (включая Windows®). Современные инструменты упрощают создание и управление ВМ, но этот метод имеет свою цену. Значительная часть ресурсов хост-системы должна выделяться каждой ВМ, и эти ресурсы не могут быть возвращены хост-системой, пока ВМ работает. Некоторые примеры менеджеров ВМ включают открытые решения qemu, bhyve и VirtualBox. Подробнее см. в главе Виртуализация.
- Удалённый доступ: Как и многие другие UNIX®-подобные системы, FreeBSD может запускать различные приложения, позволяющие пользователям удалённо подключаться к компьютерам с Windows® и использовать их программы или данные. Помимо клиентов, таких как xrdp, которые подключаются к стандартному протоколу Windows® Remote Desktop, также могут использоваться другие открытые стандарты, например, vnc (при условии наличия совместимого сервера на другой стороне).

13.3. Установка WINE на FreeBSD

WINE можно установить с помощью утилиты pkg или скомпилировав порт(ы).

13.3.1. Предварительные требования для WINE

Прежде чем устанавливать WINE, рекомендуется установить следующие программы.

- Графический пользовательский интерфейс

Большинство программ Windows® ожидают наличия графического пользовательского интерфейса. Если WINE устанавливается без него, его зависимости будут включать композитор Wayland, и тогда графический интерфейс будет установлен вместе с WINE. Однако полезно установить, настроить и проверить работу предпочитаемого графического интерфейса до установки WINE.

- wine-gecko

Операционная система Windows® уже долгое время поставляется с предустановленным веб-браузером по умолчанию — Internet Explorer. В результате некоторые приложения работают в предположении, что всегда будет доступно средство для отображения веб-страниц. Чтобы обеспечить эту функциональность, слой WINE включает компонент веб-браузера, использующий движок Gecko проекта Mozilla. При первом запуске WINE предложит загрузить и установить его, и у пользователей могут быть причины согласиться на это (эти причины будут рассмотрены в одной из следующих глав). Однако пользователи также могут установить этот компонент до установки WINE или параллельно с основной установкой WINE.

Установите этот пакет с помощью следующей команды:

```
# pkg install wine-gecko
```

Или соберите порт с помощью следующей команды:

```
# cd /usr/ports/emulators/wine-gecko
# make install
```

- wine-mono

Этот порт устанавливает фреймворк MONO, открытую реализацию .NET от Microsoft. Включение его в установку WINE увеличивает вероятность того, что приложения, написанные на .NET, будут устанавливаться и работать в системе.

Для установки пакета:

```
# pkg install wine-mono
```

Для компиляции из коллекции портов:

```
# cd /usr/ports/emulators/wine-mono
# make install
```

13.3.2. Установка WINE через репозитории пакетов FreeBSD

Имея выполненные предварительные условия, установите WINE через пакет следующей

командой:

```
# pkg install wine
```

Или можно собрать подсистему WINE из исходного кода следующим образом:

```
# cd /usr/ports/emulators/wine
# make install
```

13.3.3. Проблемы 32- и 64-битных версий в установках WINE

Как и большинство программного обеспечения, приложения Windows® перешли с устаревшей 32-битной архитектуры на 64-битную. Большинство современного ПО разрабатывается для 64-битных операционных систем, хотя современные ОС иногда могут запускать и старые 32-битные программы. FreeBSD не является исключением, поддерживая 64-битную архитектуру начиная с серии 5.x.

Однако использование устаревшего программного обеспечения, которое больше не поддерживается по умолчанию, является распространённым случаем применения эмуляторов. Пользователи часто обращаются к WINE для запуска игр и других программ, которые не работают корректно на современном оборудовании. К счастью, FreeBSD поддерживает все три сценария:

- На современных 64-битных машинах, если требуется запускать 64-битное программное обеспечение Windows®, достаточно установить порты, упомянутые в вышеуказанных разделах. Система портов автоматически установит 64-битную версию.
- Или у пользователей может быть старая 32-разрядная машина, на которой они не хотят запускать оригинальное, теперь уже неподдерживаемое программное обеспечение. Они могут установить 32-разрядную (i386) версию FreeBSD, а затем установить порты из вышеуказанных разделов.

13.4. Запуск первой программы в WINE на FreeBSD

Теперь, когда WINE установлен, следующий шаг — попробовать его в работе, запустив простую программу. Простой способ сделать это — скачать автономное приложение, то есть такое, которое можно просто распаковать и запустить без сложного процесса установки.

Так называемые "портативные" версии приложений хорошо подходят для этого теста, как и программы, которые работают с одним исполняемым файлом.

13.4.1. Запуск программы из командной строки

Существует два различных способа запуска программ Windows из терминала. Первый и наиболее простой — перейти в каталог с исполняемым файлом программы (.EXE) и выполнить следующую команду:

```
% wine program.exe
```

Для приложений, принимающих аргументы командной строки, добавьте их после исполняемого файла, как обычно:

```
% wine program2.exe -file file.txt
```

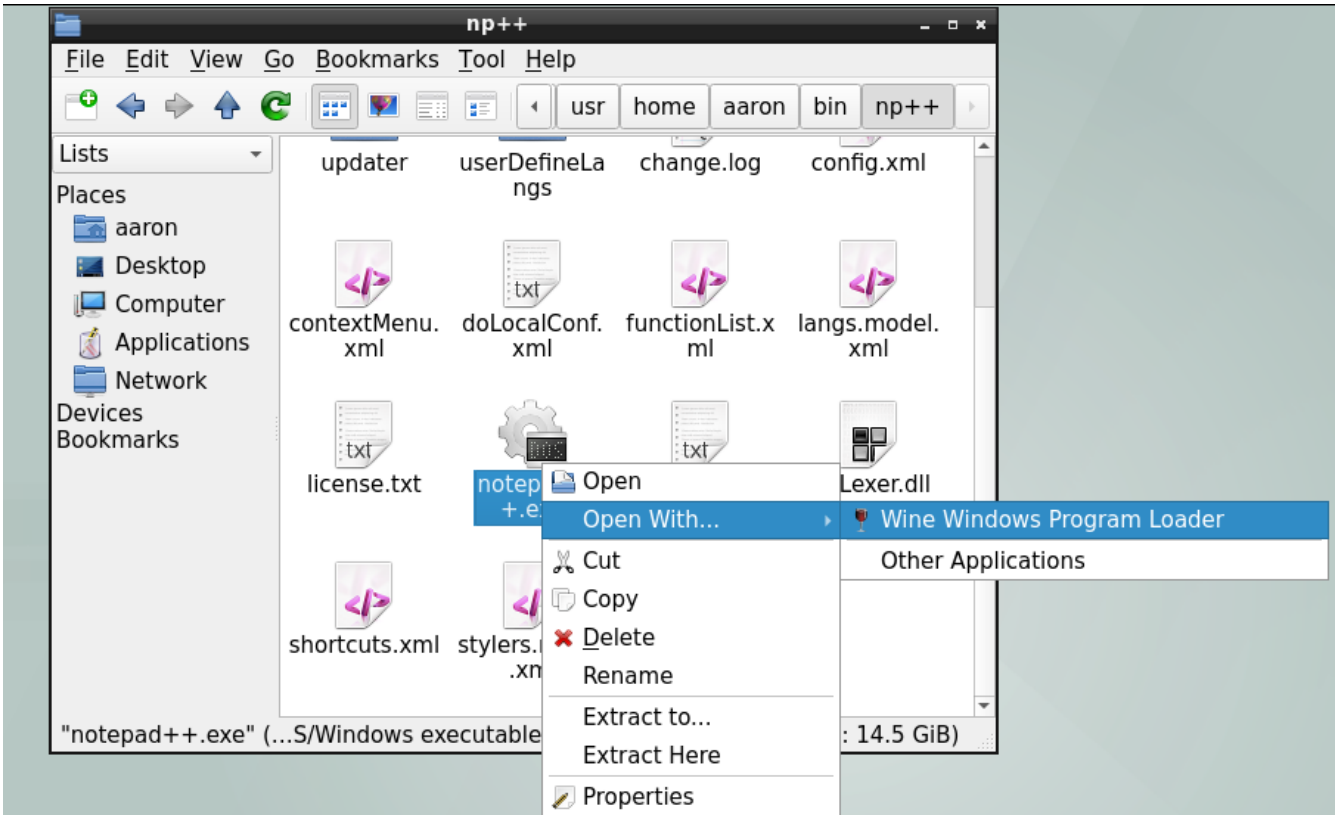
Другой способ — укажите полный путь к исполняемому файлу, чтобы использовать его в скрипте, например:

```
% wine /home/user/bin/program.exe
```

13.4.2. Запуск программы из графического интерфейса

После установки графические оболочки должны быть обновлены с новыми ассоциациями для исполняемых файлов Windows (.EXE). Теперь можно будет просматривать систему с помощью файлового менеджера и запускать Windows-приложения так же, как другие файлы и программы (одинарным или двойным щелчком, в зависимости от настроек рабочего стола).

На большинстве рабочих столов проверьте правильность этой ассоциации, щёлкнув правой кнопкой мыши на файле и найдя пункт в контекстном меню для его открытия. Один из вариантов меню (будем надеяться, выбираемый по умолчанию) будет **Wine Windows Program Loader**, как показано на рисунке ниже:



В случае, если программа не работает должным образом, попробуйте запустить её из командной строки и просмотрите сообщения, отображаемые в терминале, для устранения неполадок.

В случае, если WINE не является приложением по умолчанию для файлов .EXE после установки, проверьте ассоциацию MIME для этого расширения в текущем окружении рабочего стола, графической оболочке или файловом менеджере.

13.5. Настройка установленного WINE

Понимая, что такое WINE и как он работает на высоком уровне, следующий шаг к эффективному использованию на FreeBSD — это знакомство с его настройкой. В следующих разделах будет описано ключевое понятие *префикса WINE* и показано, как он используется для управления поведением приложений, запускаемых через WINE.

13.5.1. Префиксы WINE

Префикс WINE — это каталог, обычно расположенный в стандартном месте $HOME/.wine, хотя может находиться и в другом месте. Префикс представляет собой набор конфигураций и вспомогательных файлов, используемых WINE для настройки и запуска среды Windows®, необходимой для работы конкретного приложения. По умолчанию при первом запуске WINE пользователем создаётся следующая структура:

- .update-timestamp: содержит дату последнего изменения файла /usr/share/wine/wine.inf. Используется WINE для определения устаревания префикса и его автоматического обновления при необходимости.
- dosdevices/: содержит информацию о сопоставлениях ресурсов Windows® с ресурсами в основной системе FreeBSD. Например, после новой установки WINE здесь должны быть как минимум две записи, обеспечивающие доступ к файловой системе FreeBSD с использованием букв дисков в стиле Windows®:
  - c:@: Ссылка на drive_c, описанный ниже.
  - z:@: Ссылка на корневой каталог системы.
- drive_c/: эмулирует основной диск (то есть C:) системы Windows®. Он содержит структуру каталогов и связанные файлы, отражающие стандартную структуру Windows®. В новом префиксе WINE будут присутствовать каталоги Windows® 10, такие как *Users* и *Windows*, в котором находится сама ОС. Кроме того, приложения, установленные в префиксе, будут располагаться в *Program Files* или *Program Files (x86)* в зависимости от их архитектуры.
- system.reg: Этот файл реестра содержит информацию об установке Windows®, которая в случае WINE представляет собой окружение в drive_c.
- user.reg: Этот файл реестра содержит персональные настройки текущего пользователя, созданные различным программным обеспечением или через использование редактора реестра.
- userdef.reg: Этот файл реестра содержит набор конфигураций по умолчанию для вновь создаваемых пользователей.

13.5.2. Создание и использование префиксов WINE

Хотя WINE создаст префикс по умолчанию в каталоге $HOME/.wine/ пользователя, можно настроить несколько префиксов. Для этого есть несколько причин:

- Наиболее распространённая причина — эмуляция различных версий Windows® в соответствии с требованиями совместимости используемого программного обеспечения.
- Кроме того, часто встречается программное обеспечение, которое некорректно работает в стандартном окружении и требует специальной настройки. Полезно изолировать такие программы в собственных пользовательских префиксах, чтобы изменения не влияли на другие приложения.
- Аналогично, копирование стандартного или "основного" префикса в отдельный "тестовый" для проверки совместимости приложения может снизить вероятность повреждения.

Создание префикса из терминала требует выполнения следующей команды:

```
% WINEPREFIX="/home/username/.wine-new" winecfg
```

Это запустит программу `winecfg`, которую можно использовать для настройки префиксов wine (подробнее об этом в следующем разделе). Однако, указав путь к каталогу в переменной окружения `WINEPREFIX`, можно создать новый префикс по этому пути, если он ещё не существует.

Предоставление той же переменной программе wine аналогично приведет к запуску выбранной программы с указанным префиксом:

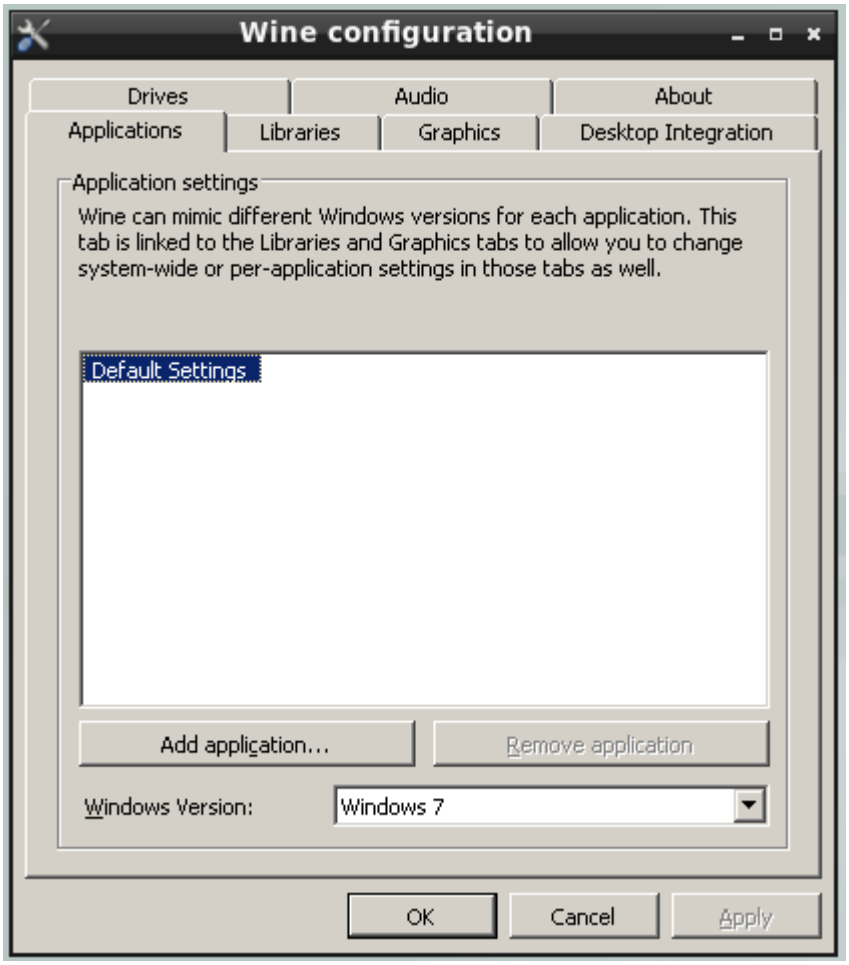
```
% WINEPREFIX="/home/username/.wine-new" wine program.exe
```

13.5.3. Настройка префиксов WINE с помощью winecfg

Как упоминалось выше, WINE включает инструмент под названием `winecfg` для настройки префиксов через графический интерфейс. Он содержит множество функций, которые подробно описаны в следующих разделах. Когда `winecfg` запускается внутри префикса или ему указывается расположение префикса через переменную `WINEPREFIX`, он позволяет настраивать выбранный префикс, как описано в разделах ниже.

Выбор на вкладке *Приложения (Applications)* влияет на область изменений, вносимых на вкладках *Библиотеки (Libraries)* и *Графика (Graphics)*, которые будут ограничены выбранным приложением. Подробнее см. раздел Использование Winecfg в вики WINE.

Приложения (Applications)

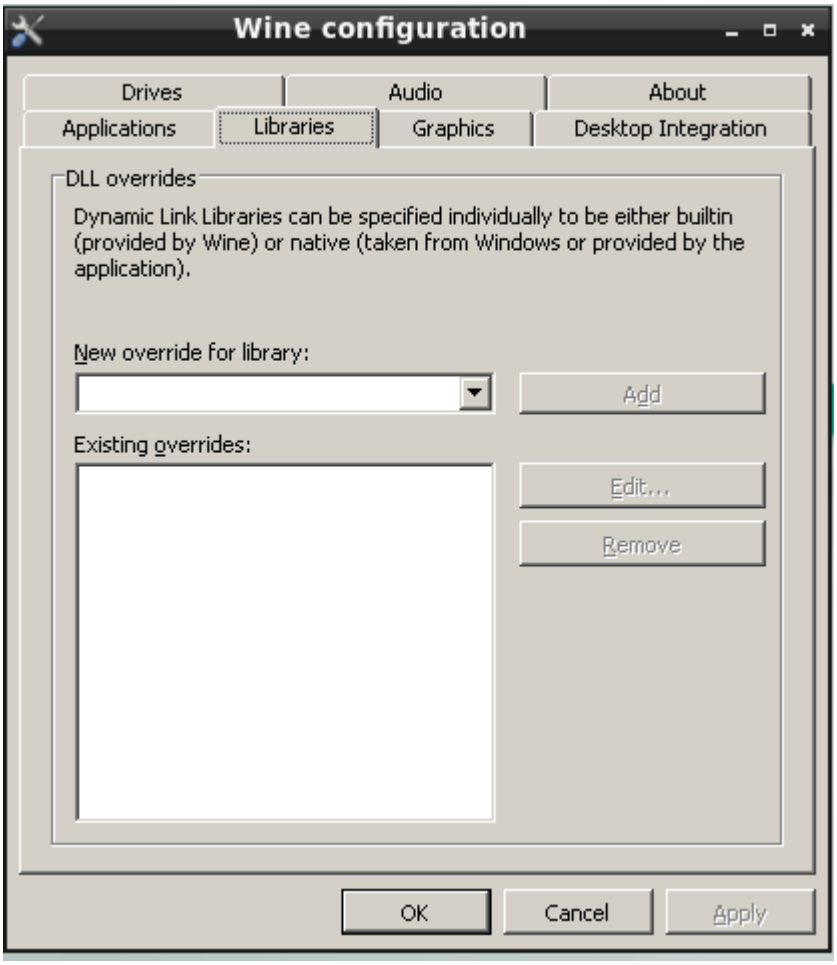


В разделе *Приложения (Applications)* находятся элементы управления, позволяющие связать программы с определённой версией Windows®. При первом запуске в разделе *Настройки приложений (Application settings)* будет только одна запись: *Настройки по умолчанию (Default Settings)*. Она соответствует всем конфигурациям префикса по умолчанию, которые (как подразумевает неактивная кнопка *Удалить приложение (Remove application)*) нельзя удалить.

Однако дополнительные приложения можно добавить с помощью следующего процесса:

1. Нажмите кнопку *Добавить приложение (Add application)*.
2. Используйте предоставленное диалоговое окно для выбора исполняемого файла нужной программы.
3. Выберите версию Windows® для использования с выбранной программой.

Библиотеки (Libraries)



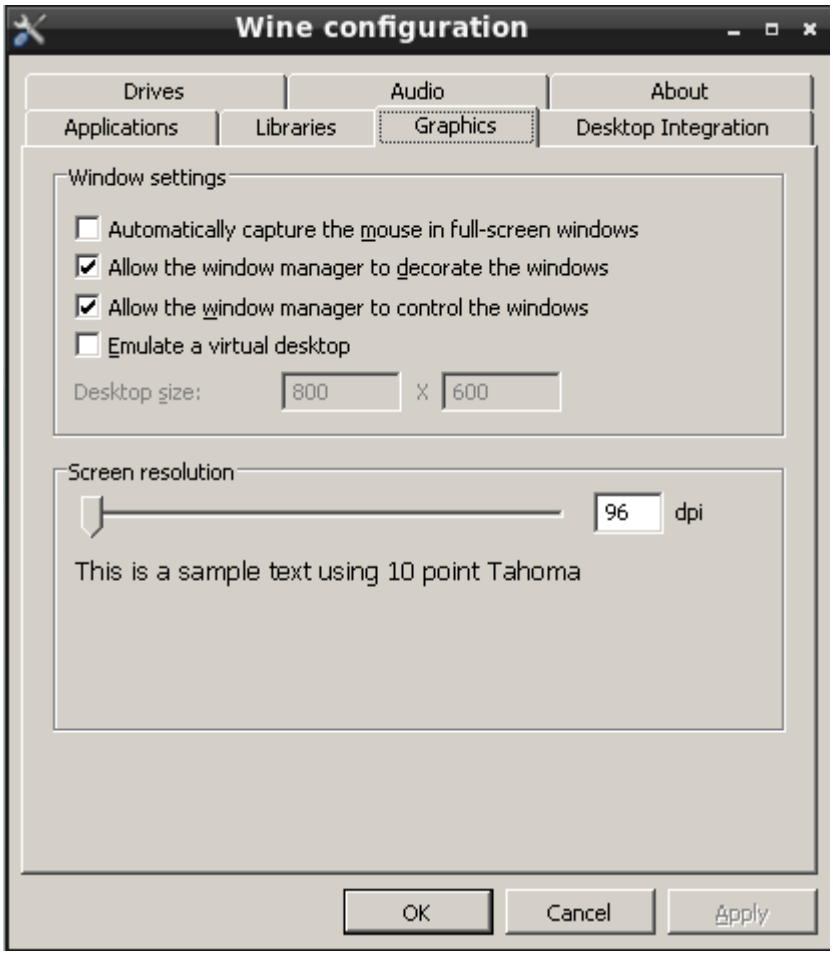
WINE предоставляет набор библиотечных файлов с открытым исходным кодом в составе своей дистрибуции, которые выполняют те же функции, что и их аналоги в Windows®. Однако, как уже отмечалось ранее в этой главе, проект WINE постоянно стремится идти в ногу с новыми обновлениями этих библиотек. В результате, версии, поставляемые с WINE, могут не иметь функциональности, которую ожидают последние программы для Windows®.

Однако `winecfg` позволяет указать переопределения для встроенных библиотек, особенно если на той же машине, где установлена FreeBSD, доступна версия Windows®. Для каждой библиотеки, которую нужно переопределить, выполните следующее:

1. Откройте выпадающий список *Новое переопределение для библиотеки (New override for library)* и выберите библиотеку, которую нужно заменить.
2. Нажмите кнопку *Добавить (Add)*.
3. Новое переопределение появится в списке *Существующие переопределения (Existing overrides)*, обратите внимание на обозначение *native, builtin* в скобках.
4. Нажмите, чтобы выбрать библиотеку.
5. Нажмите кнопку *Правка (Edit)*.
6. Используйте предоставленное диалоговое окно для выбора соответствующей библиотеки, которая будет использоваться вместо встроенной.

Убедитесь, что выбранный файл действительно соответствует версии встроенного, иначе возможно непредвиденное поведение.

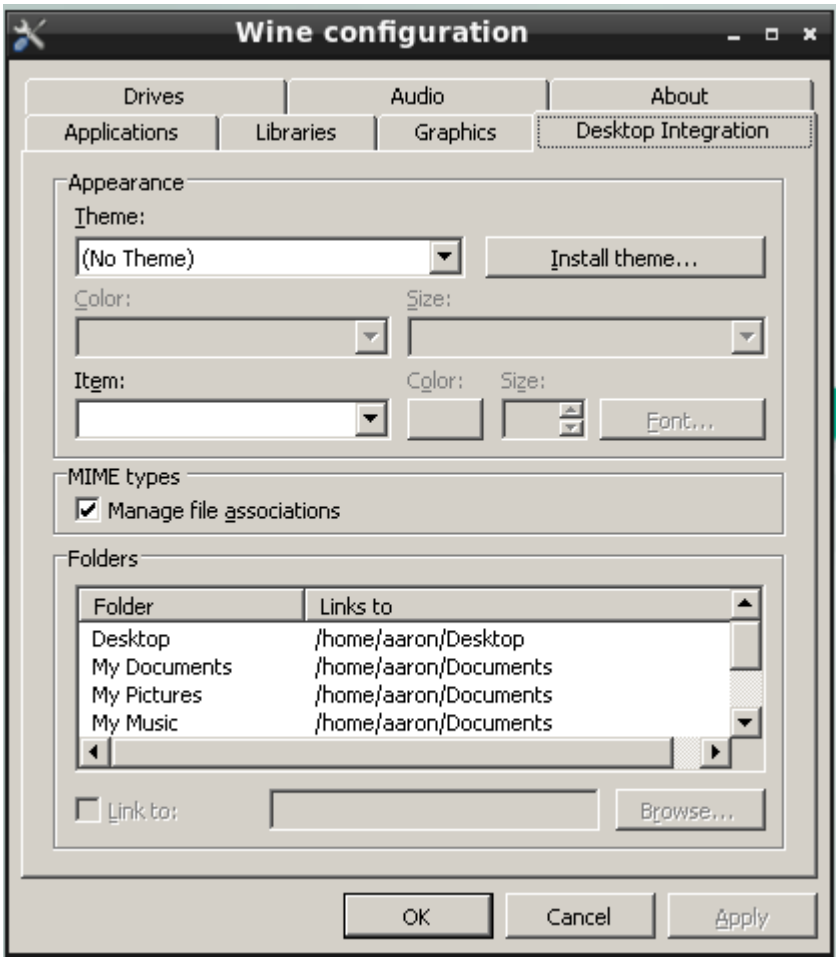
Графика (Graphics)



Вкладка *Графика (Graphics)* предоставляет несколько опций для обеспечения плавной работы окон программ, запущенных через WINE, в FreeBSD:

- Автоматический захват мыши при полноэкранном режиме окон.
- Разрешение оконному менеджеру FreeBSD оформлять окна, например, их заголовки, для программ, работающих через WINE.
- Разрешение оконному менеджеру управлять окнами программ, работающих через WINE, например, выполнение функций изменения их размера.
- Создать эмулированный виртуальный рабочий стол, в пределах которого будут запускаться все программы WINE. Если этот пункт выбран, размер виртуального рабочего стола можно указать с помощью полей ввода *Размер рабоче (Desktop size)го стола*.
- Установка разрешения экрана для программ, работающих через WINE.

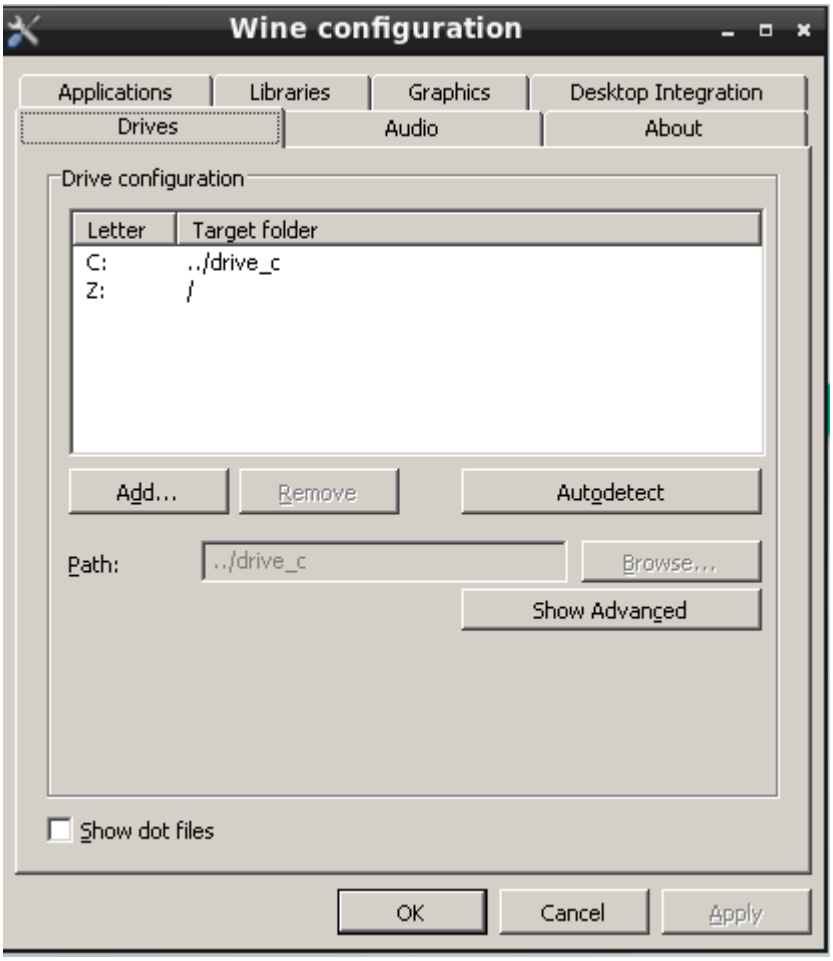
Интеграция с рабочим столом (Desktop Integration)



Эта вкладка позволяет настроить следующие элементы:

- Тему и связанные визуальные настройки, используемые для программ, работающих через WINE.
- Должна ли подсистема WINE управлять типами MIME (используемыми для определения, какое приложение открывает файлы определённого типа) внутри себя.
- Сопоставление каталогов в основной системе FreeBSD с полезными папками в среде Windows®. Чтобы изменить существующую ассоциацию, выберите нужный элемент и нажмите *Просмотреть (Browse)*, затем используйте предоставленный диалог для выбора каталога.

Накопители (Drives)



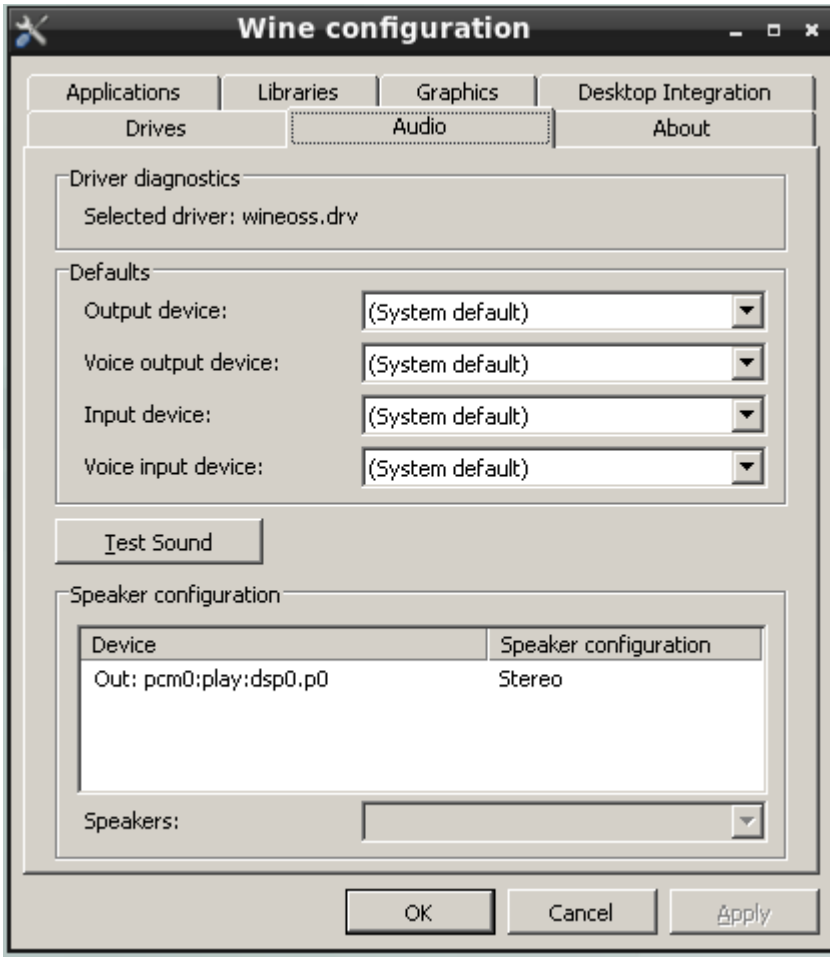
Вкладка *Накопители (Drives)* позволяет связывать каталоги в основной системе FreeBSD с буквами дисков в среде Windows®. Значения по умолчанию на этой вкладке должны выглядеть знакомо, так как они отображают содержимое dosdevices/ в текущем префиксе WINE. Изменения, сделанные через это диалоговое окно, отобразятся в dosdevices, а правильно оформленные ссылки, созданные в этом каталоге, будут показаны на этой вкладке.

Чтобы создать новую запись, например, для CD-ROM (монтированного в /mnt/cdrom), выполните следующие действия:

1. Нажмите кнопку *Добавить (Add)*.
2. В предоставленном диалоговом окне выберите свободную букву диска.
3. Нажмите *OK*.
4. Заполните поле ввода *Путь (Path)*, введя путь к ресурсу вручную или нажав *Просмотр (Browse)* и выбрав его в открывшемся диалоговом окне.

По умолчанию WINE автоматически определяет тип связанного ресурса, но это можно переопределить вручную. Дополнительные сведения о расширенных настройках см. в соответствующем разделе вики WINE.

Звук (Audio)



Эта вкладка содержит некоторые настраиваемые параметры для маршрутизации звука из программ Windows® в родную звуковую систему FreeBSD, включая:

- Выбор драйвера
- Выбор устройства по умолчанию
- Звуковой тест

О программе (About)

Последняя вкладка содержит информацию о проекте WINE, включая ссылку на веб-сайт. Также она позволяет ввести (совершенно необязательные) пользовательские данные, хотя они никуда не отправляются, в отличие от других операционных систем.

13.6. Графические программы для управления WINE

Хотя базовая установка WINE включает графический инструмент настройки `winecfg`, его основное назначение строго ограничивается конфигурацией существующего префикса WINE. Однако существуют более продвинутые приложения, которые помогают не только в первоначальной установке программ, но и в оптимизации их окружения WINE. В следующих разделах представлен выбор наиболее популярных из них.

13.6.1. Winetricks

Инструмент `winetricks` — это кроссплатформенная вспомогательная программа общего назначения для WINE. Он разрабатывается не самим проектом WINE, а поддерживается группой участников на GitHub. `winetricks` содержит автоматизированные "рецепты" для запуска распространённых приложений в WINE, включая оптимизацию настроек и автоматическую загрузку некоторых DLL-библиотек.

Установка winetricks

Для установки `winetricks` в FreeBSD с использованием бинарных пакетов выполните

следующие команды (обратите внимание, что `winetricks` требует наличия пакета i386-wine или i386-wine-devel и поэтому не устанавливается автоматически с другими зависимостями):

```
# pkg install i386-wine winetricks
```

Чтобы скомпилировать его из исходного кода, выполните следующую команду в терминале:

```
# cd /usr/ports/emulators/i386-wine
# make install
# cd /usr/ports/emulators/winetricks
# make install
```

Если требуется ручная установка, обратитесь к аккаунту на Github для получения инструкций.

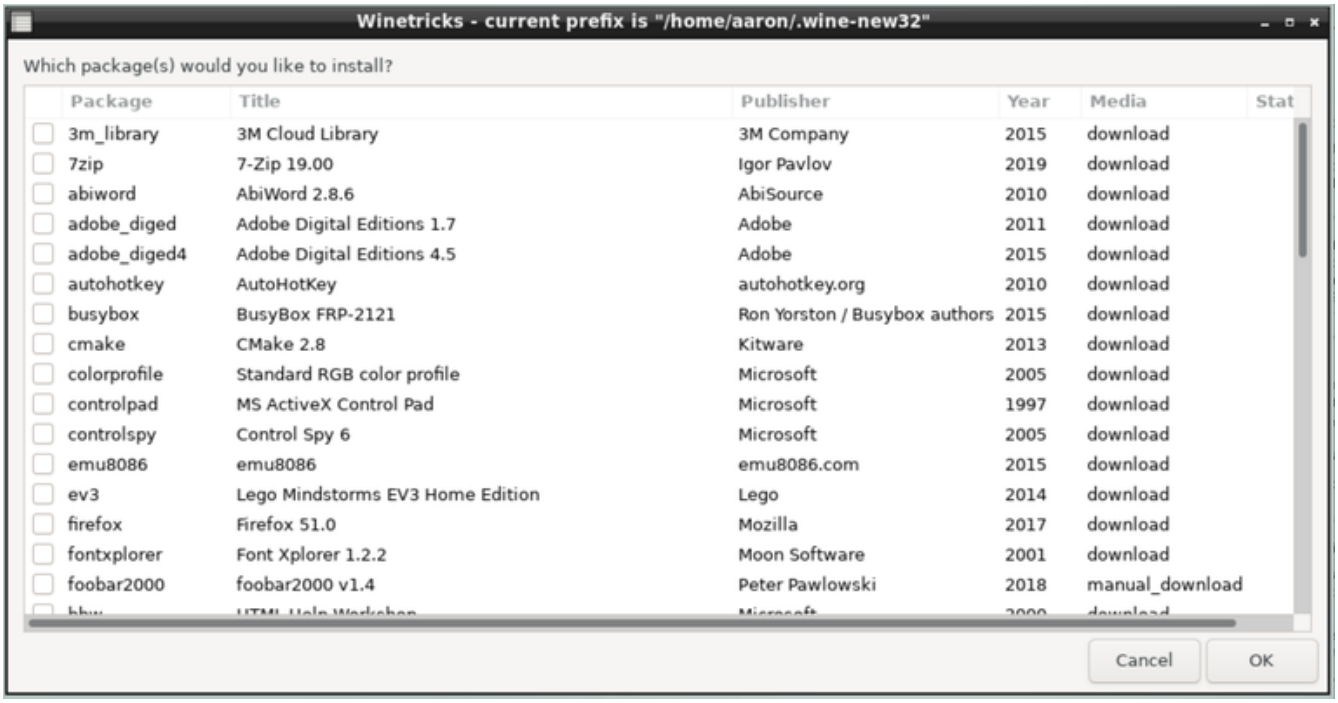
Использование winetricks

Выполните `winetricks` следующей командой:

```
% winetricks
```

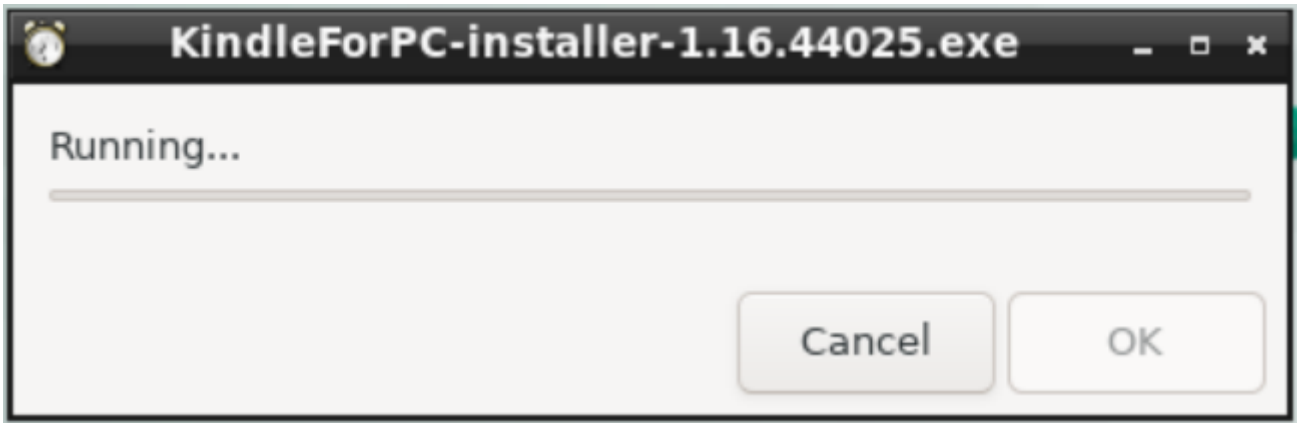
Примечание: для запуска `winetricks` требуется 32-битный префикс. При запуске `winetricks` отображается окно с несколькими вариантами выбора, как показано ниже:

Выбор любого из пунктов *Установить приложение*, *Установить тест производительности* или *Установить игру* отображает список доступных вариантов, например, такой как показанный ниже список приложений:

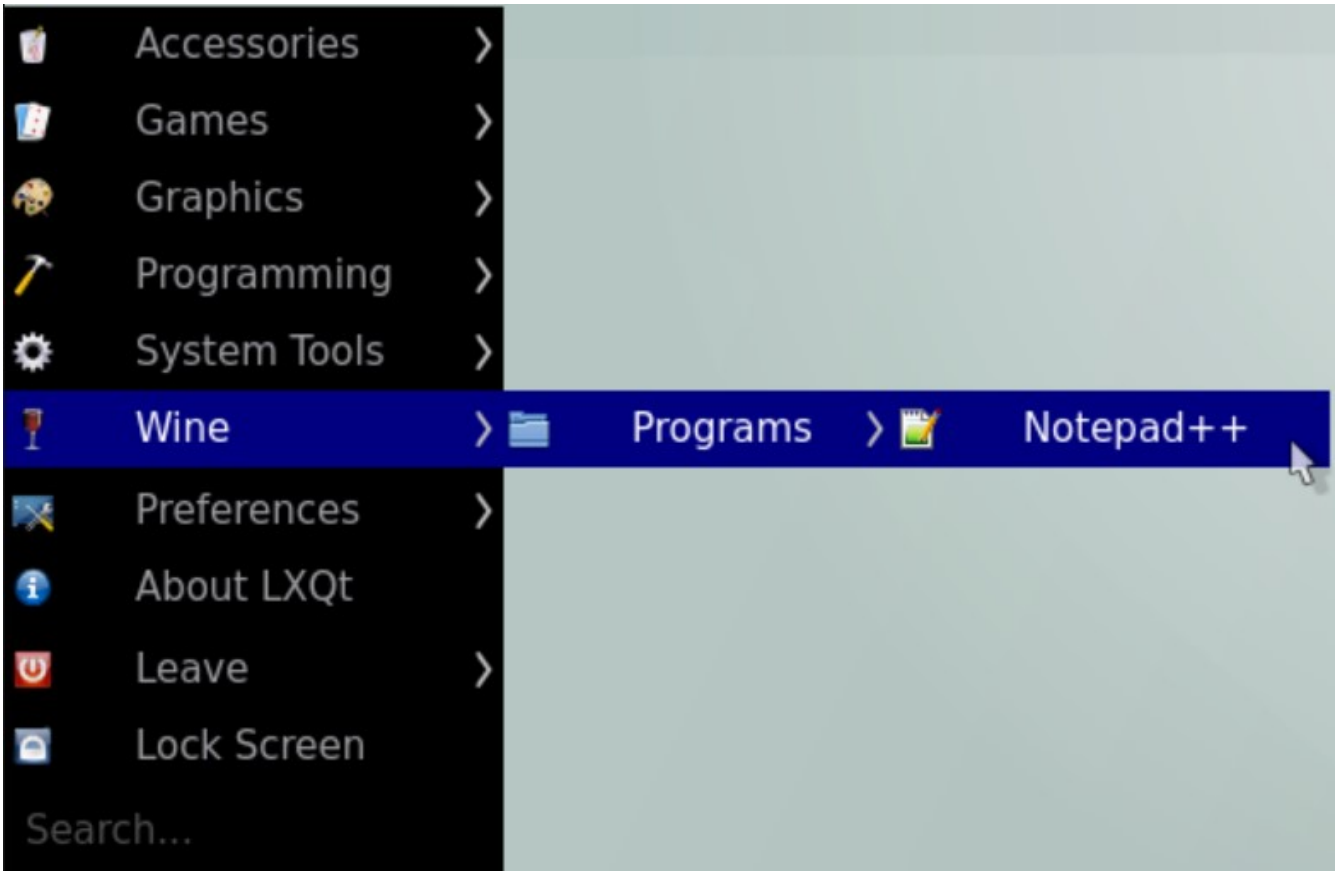


Выбор одного или нескольких элементов и нажатие *OK* запустит процесс(ы) их установки. Вначале могут появиться сообщения, которые выглядят как ошибки, но на самом деле это информационные предупреждения, так как `winetricks` настраивает среду WINE для обхода известных проблем приложения:

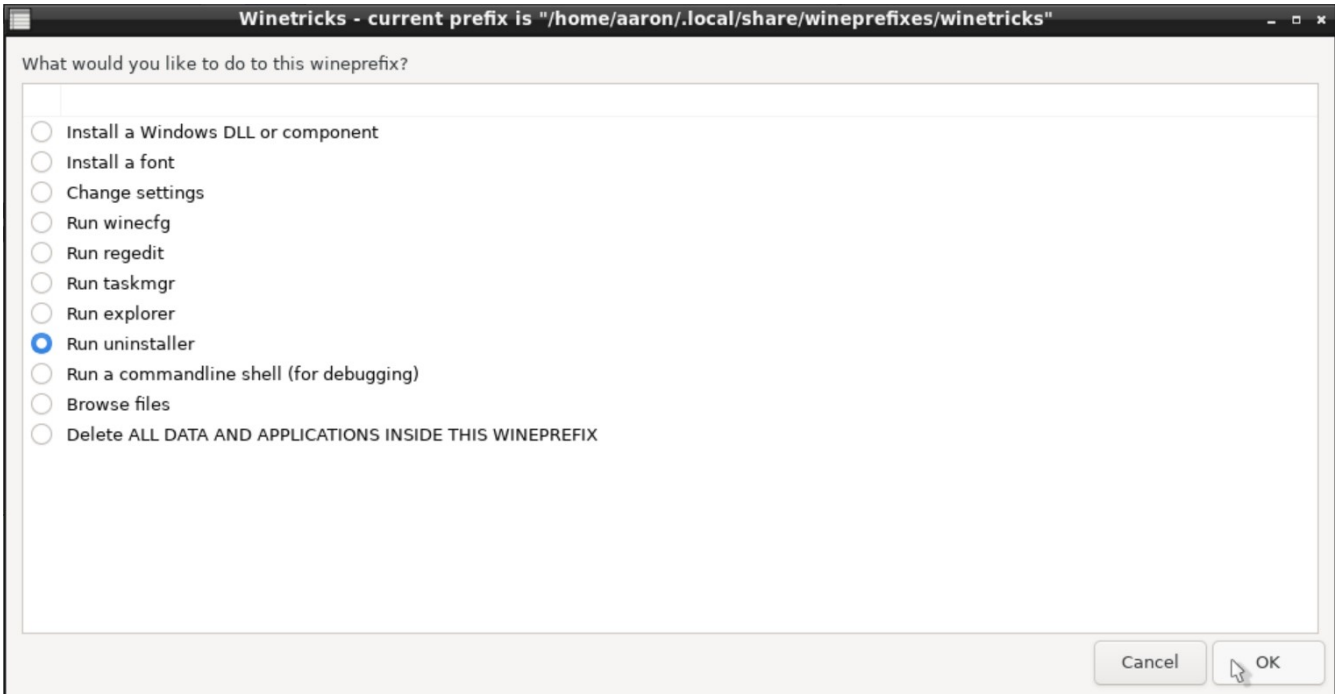
После обхода этих препятствий будет запущен настоящий установщик приложения:



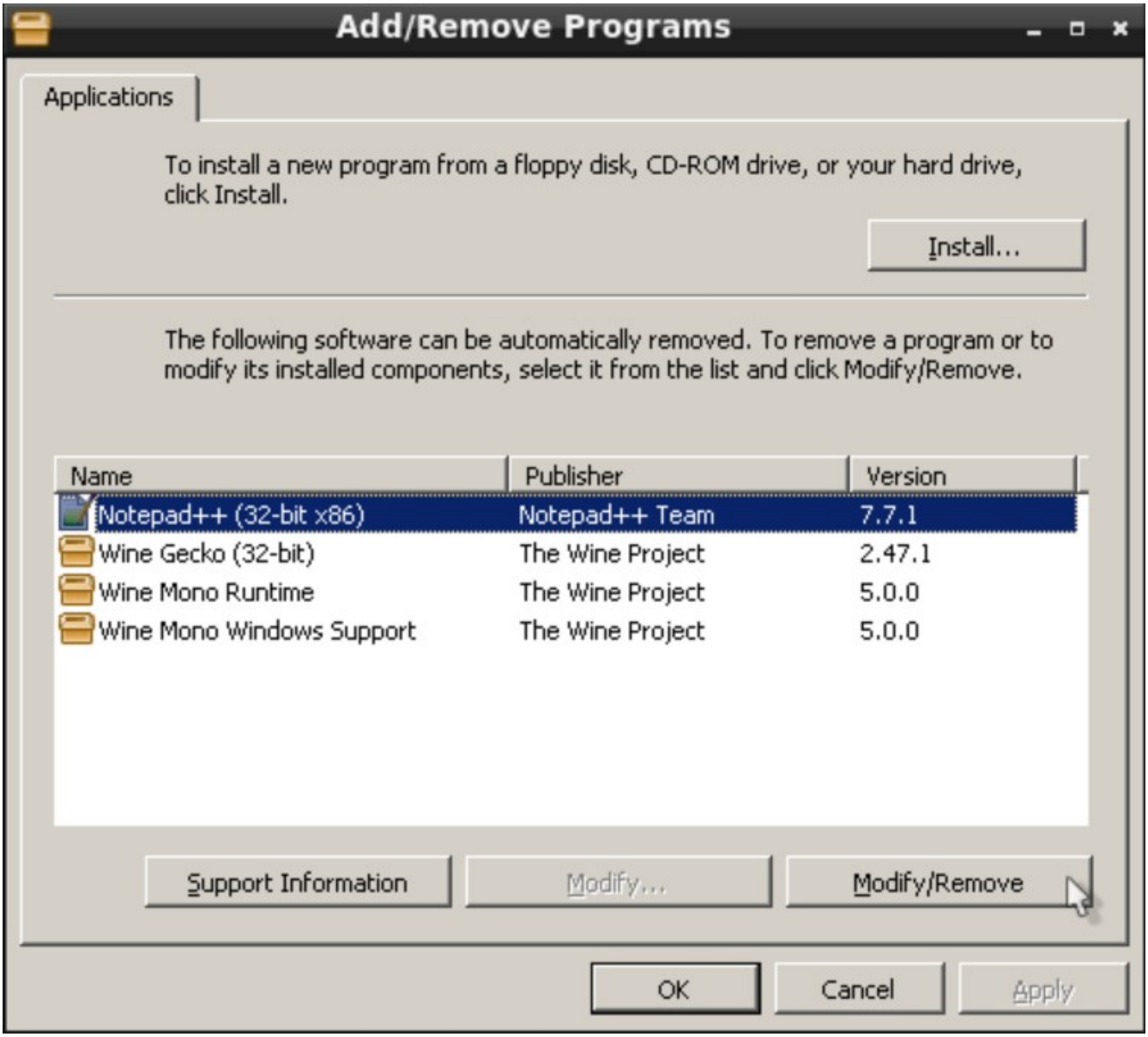
После завершения установки новое приложение Windows должно быть доступно в стандартном меню среды рабочего стола (как показано на скриншоте ниже для среды рабочего стола LXQT):



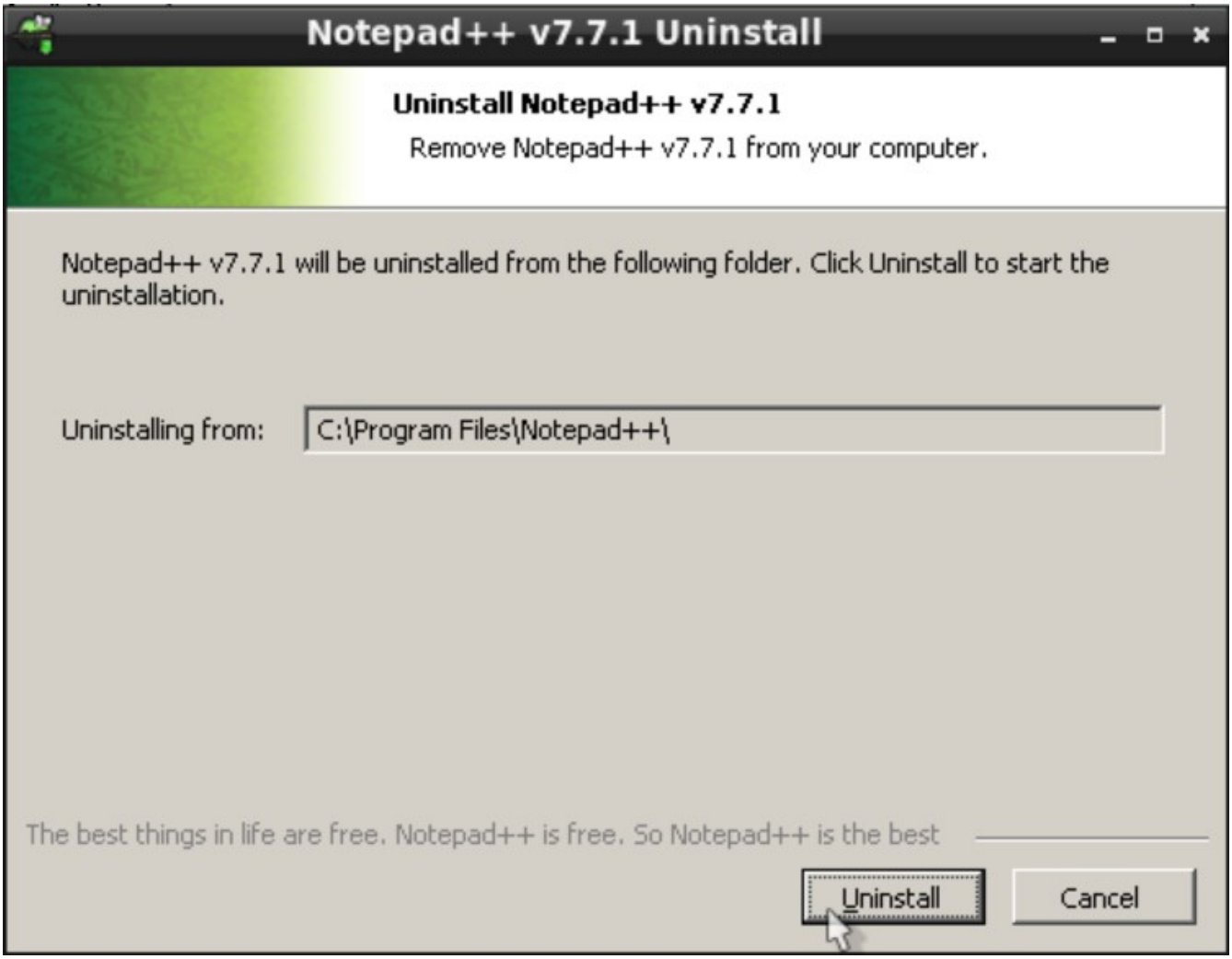
Чтобы удалить приложение, снова запустите `winetricks` и выберите *Запустить удаление инсталляции (Run an uninstaller)*.



Появится диалоговое окно в стиле Windows® со списком установленных программ и компонентов. Выберите приложение для удаления и нажмите кнопку *Изменить/Удалить (Modify/Remove)*.



Это запустит встроенный установщик приложения, который также должен иметь возможность удаления.



13.6.2. Mizutamari

Mizutamari — это приложение, похожее на `winetricks`, хотя вдохновлено игровой системой Lutris для Linux. Однако, в отличие от неё, Mizutamari ориентировано не только на игры: через него также можно установить и приложения неигрового характера.

Установка Mizutamari

Для установки бинарного пакета Mizutamari выполните следующую команду:

```
# pkg install mizuma
```

Mizutamari доступен в системе портов FreeBSD. Однако вместо поиска в разделе *emulators* портов или двоичных пакетов ищите его в разделе *games*.

```
# cd /usr/ports/games/mizuma
# make install
```

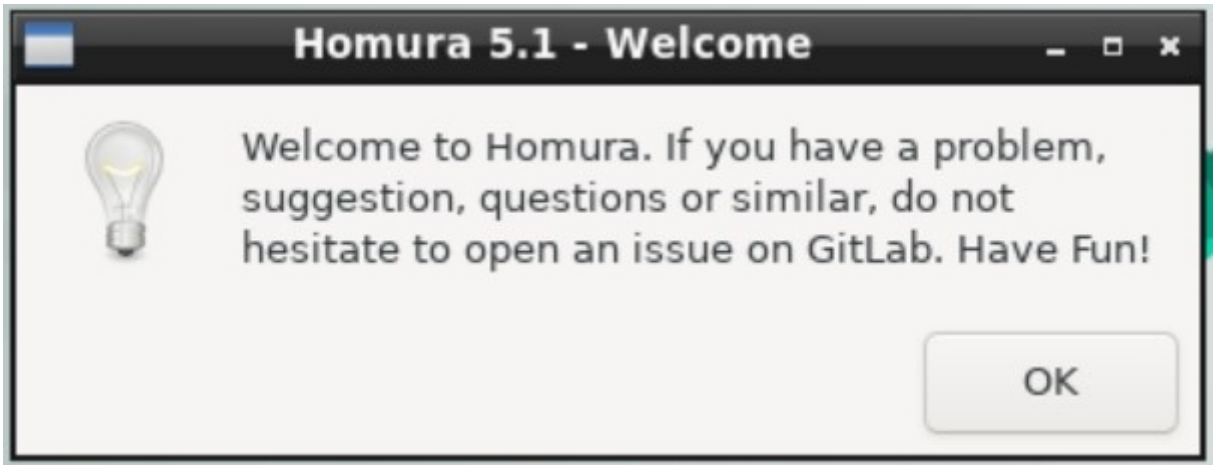
Использование Mizutamari

Использование Mizutamari очень похоже на использование `winetricks`. При первом запуске

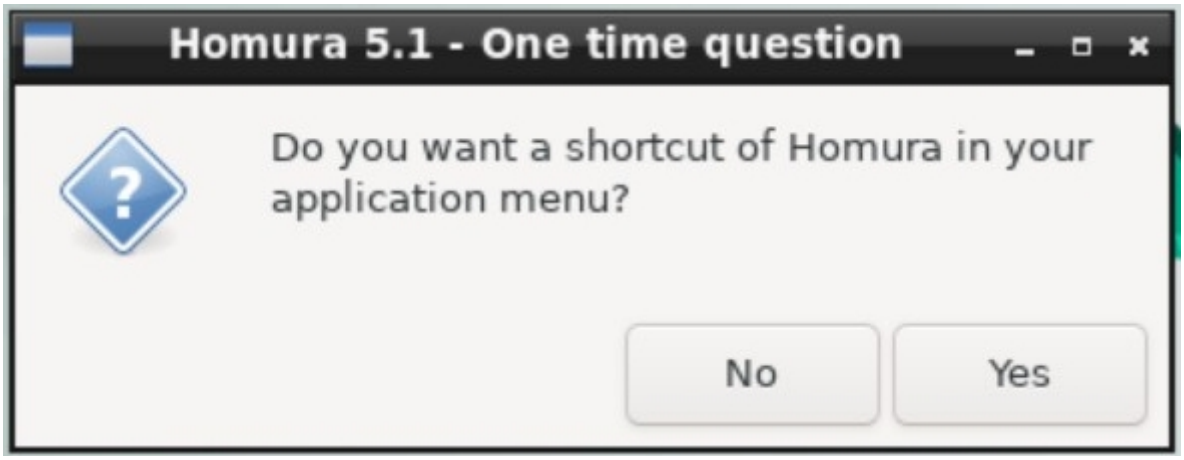
выполните его из командной строки (или с помощью апплета запуска окружения рабочего стола) командой:

```
% Mizuma
```

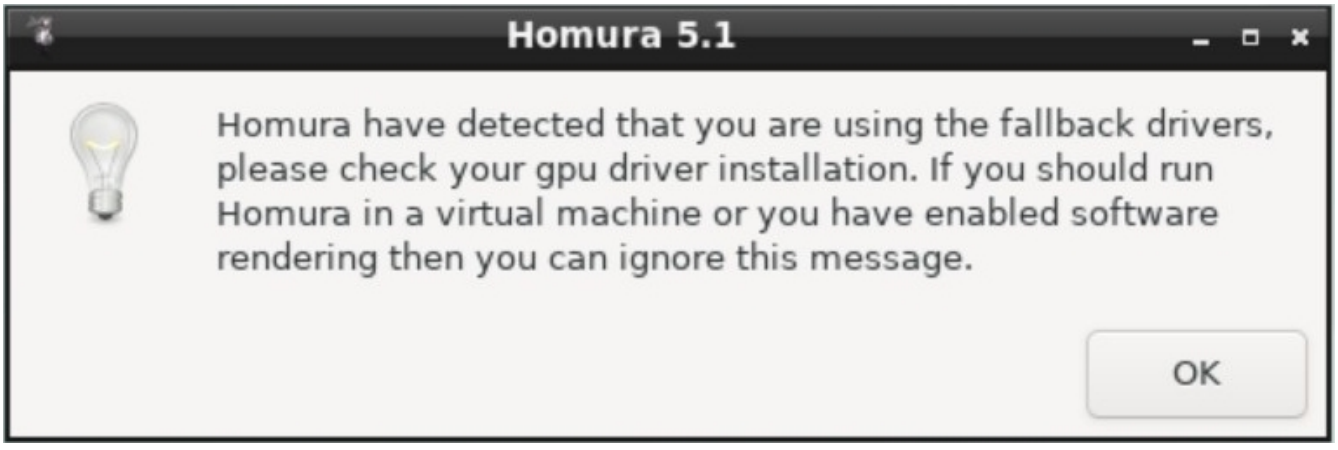
Должно появиться дружественное приветственное сообщение. Нажмите *OK* для продолжения.



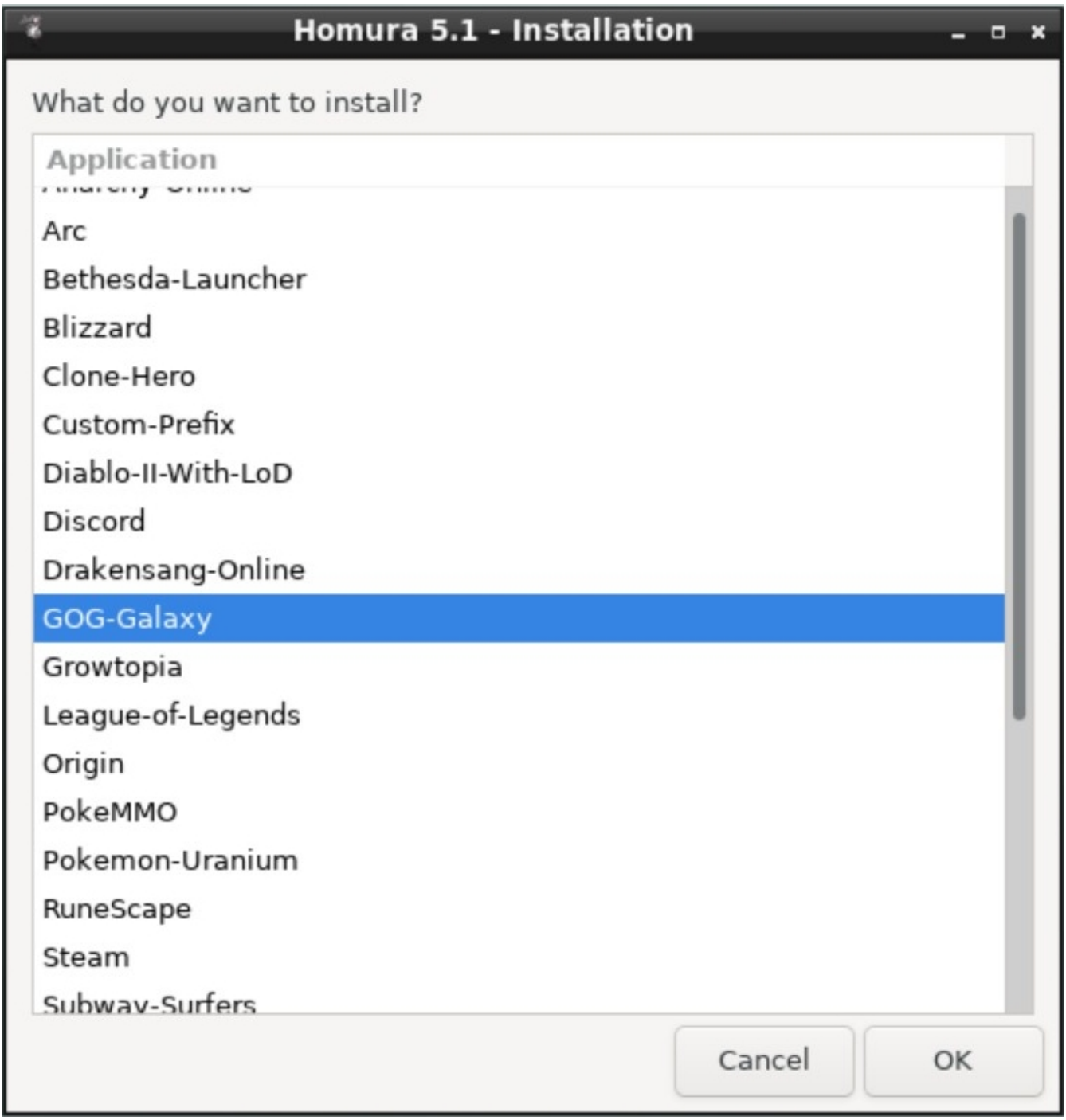
Программа также предложит разместить ссылку в меню приложений совместимых сред:



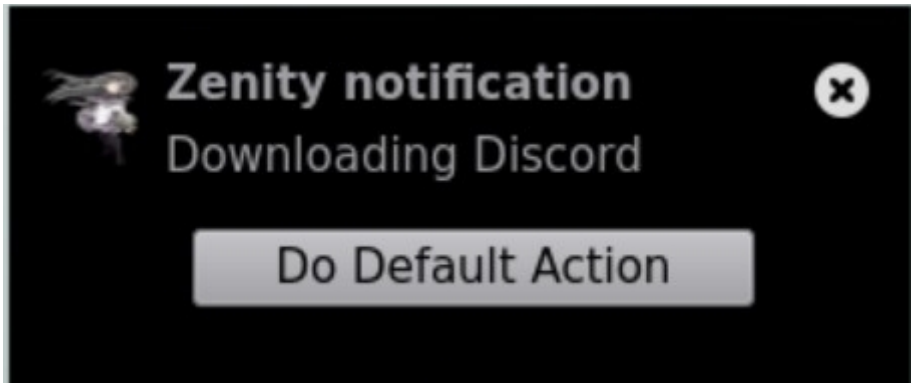
В зависимости от настройки машины FreeBSD, Mizutamari может отображать сообщение с рекомендацией установить родные графические драйверы.



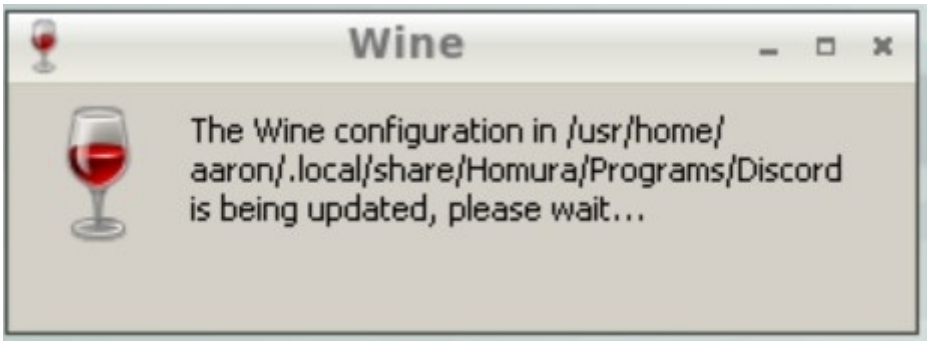
Затем должно появиться окно приложения, которое представляет собой «главное меню» со всеми его опциями. Многие пункты совпадают с `winetricks`, хотя Mizutamari предлагает некоторые дополнительные полезные опции, такие как открытие папки с данными (*Open Mizutamari Folder*) или запуск указанной программы (*Run a executable in prefix*).



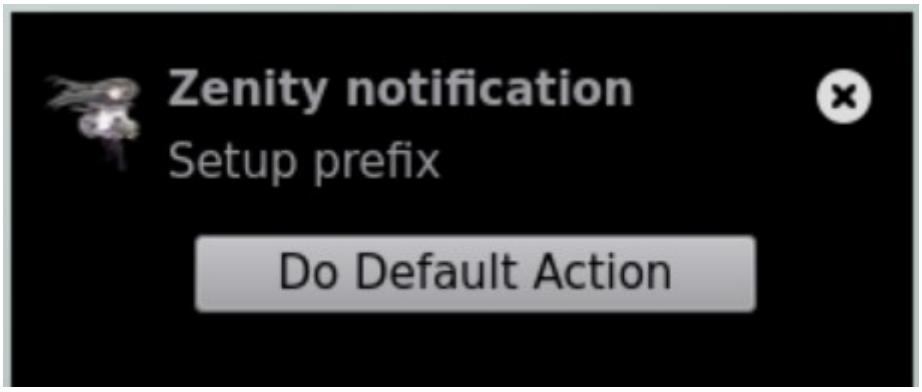
Чтобы выбрать одно из поддерживаемых приложений Mizutamari для установки, выберите *Установка* и нажмите *OK*. Это отобразит список приложений, которые Homura может установить автоматически. Выберите нужное приложение и нажмите *OK*, чтобы начать процесс.



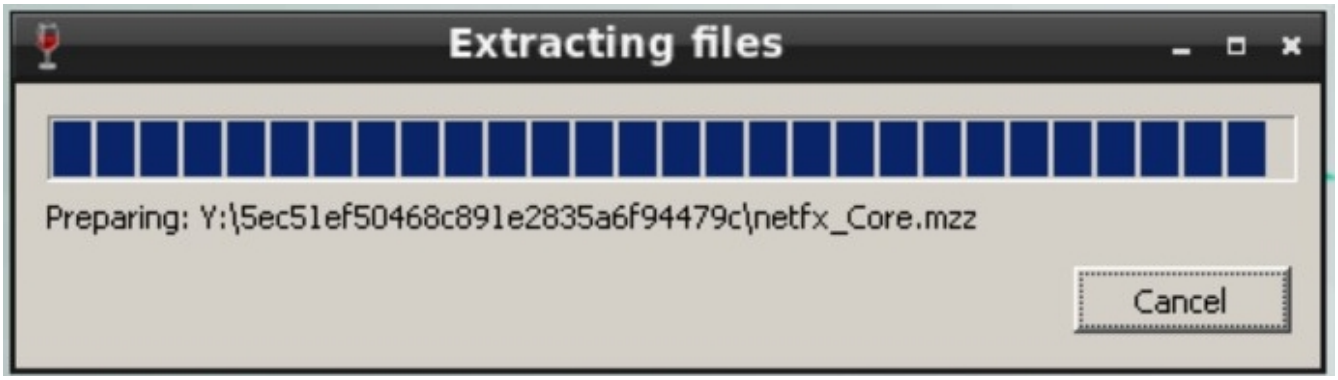
В качестве первого шага Mizutamari загрузит выбранную программу. В поддерживаемых рабочих средах может появиться уведомление.



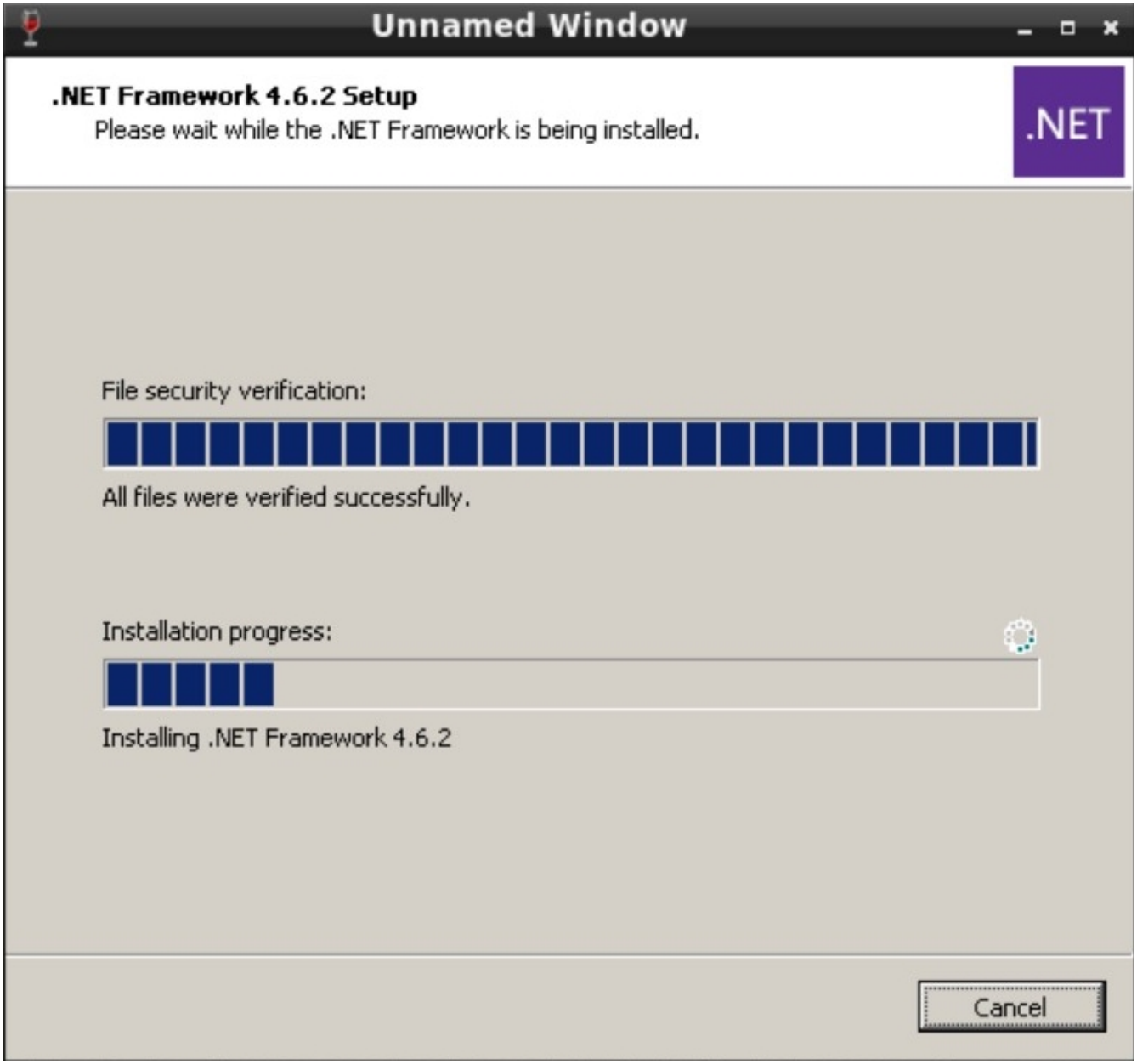
Программа также создаст новый префикс для приложения. Появится стандартное диалоговое окно WINE с этим сообщением.



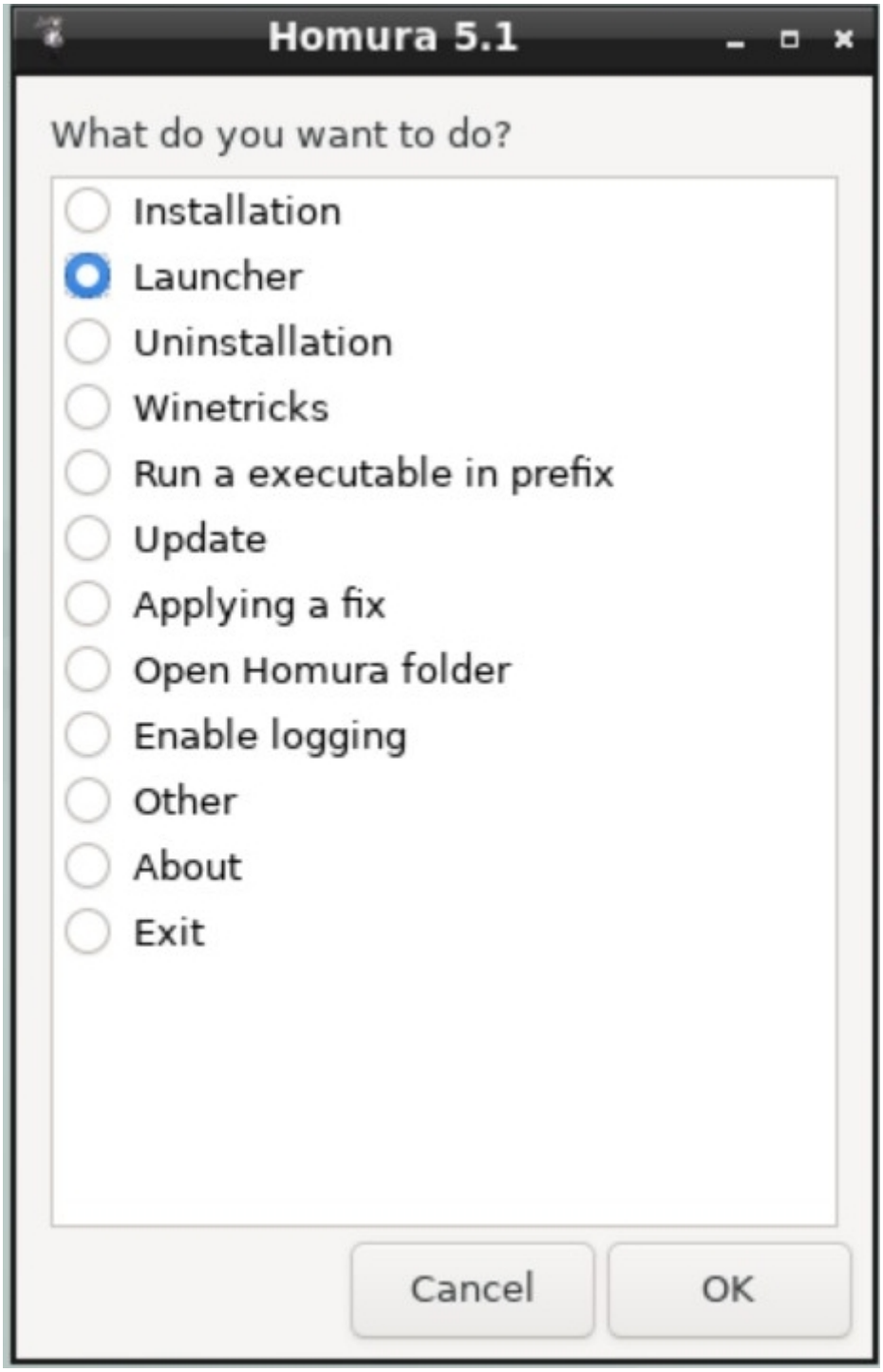
Затем Mizutamari установит все необходимые зависимости для выбранной программы. Это может включать загрузку и распаковку значительного количества файлов, подробности чего будут отображаться в диалоговых окнах.



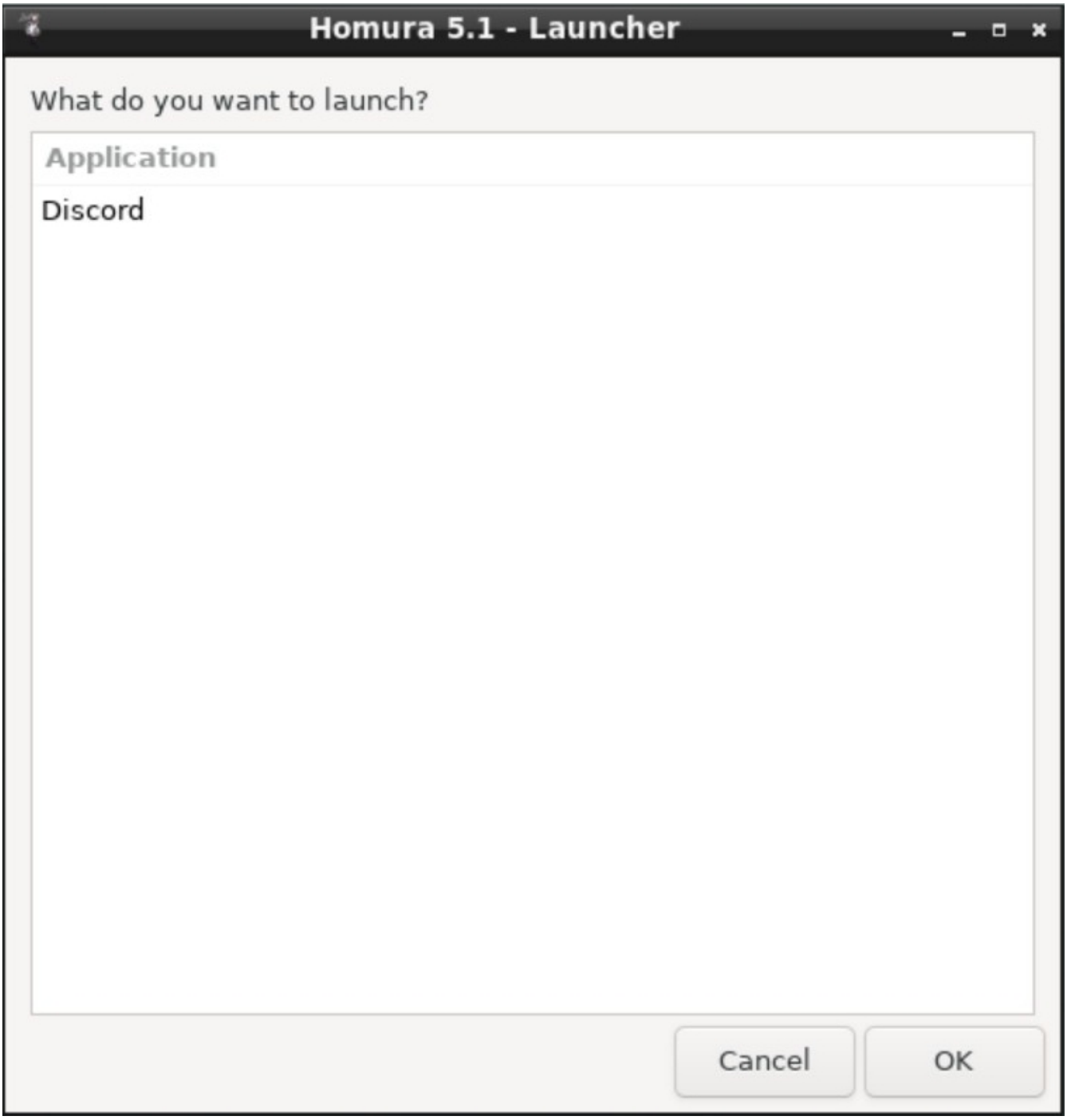
Загруженные пакеты автоматически открываются и запускаются по мере необходимости.



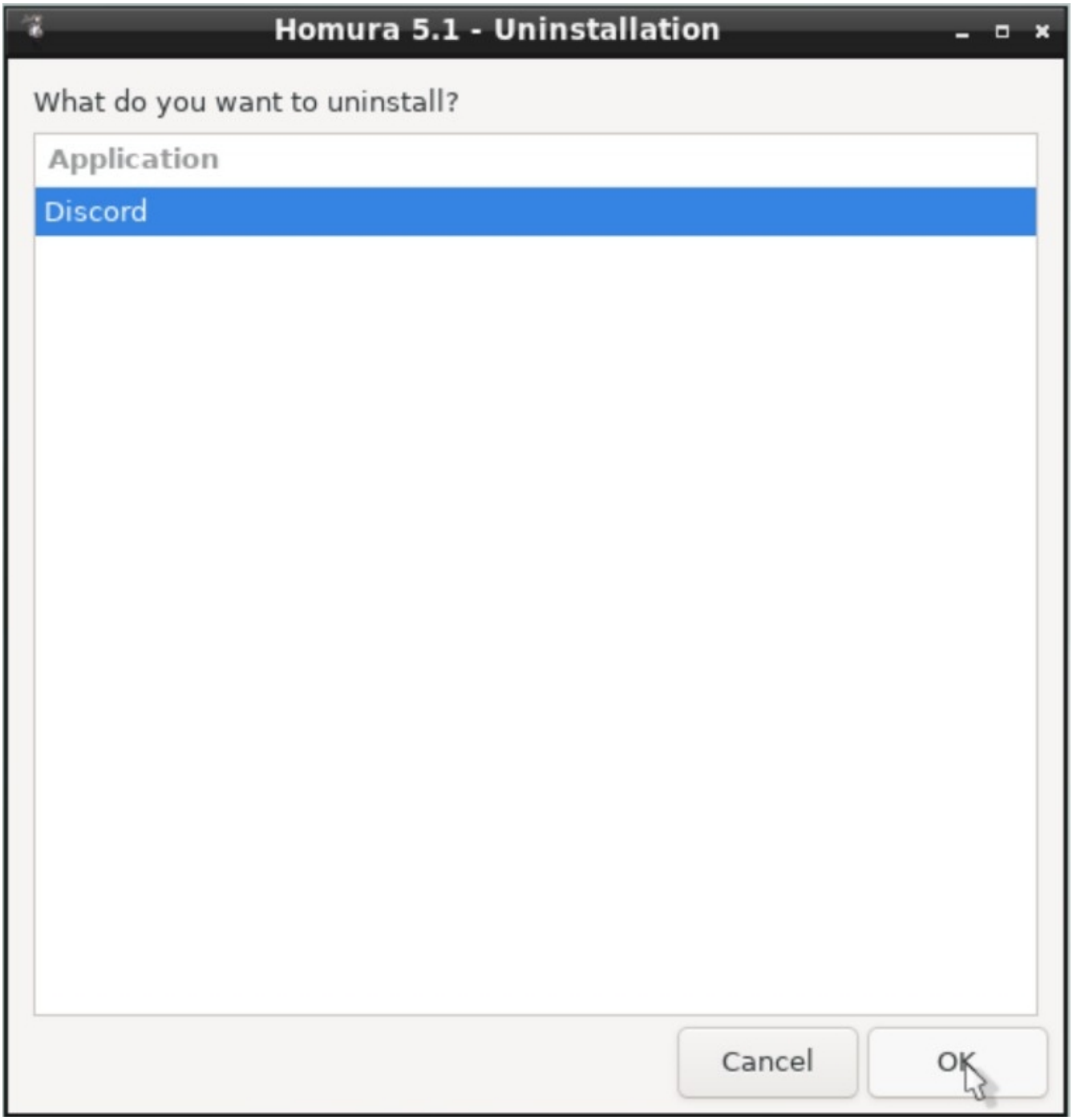
Установка может завершиться простым уведомлением на рабочем столе или сообщением в терминале, в зависимости от того, как был запущен Mizutamari. Однако в любом случае Mizutamari должен вернуться на главный экран. Чтобы подтвердить успешность установки, выберите *Launcher* и нажмите *OK*.



Это выведет список установленных приложений.



Чтобы запустить новую программу, выберите её из списка и нажмите *OK*. Для удаления приложения выберите пункт *Удаление* на главном экране, после чего отобразится аналогичный список. Выберите программу для удаления и нажмите *OK*.



13.6.3. Запуск нескольких графических интерфейсов управления

Стоит отметить, что приведённые выше решения не являются взаимоисключающими. Вполне допустимо и даже выгодно иметь оба пакета установленными одновременно, так как они поддерживают разные наборы программ.

Однако разумно убедиться, что они не обращаются к одним и тем же префиксам WINE. Каждое из этих решений применяет обходные пути и вносит изменения в реестры на основе известных исправлений существующих проблем WINE, чтобы обеспечить бесперебойную работу конкретного приложения. Если разрешить и `winetricks`, и Homura доступ к одному и тому же префиксу, это может привести к перезаписи некоторых из этих изменений, в результате чего некоторые или все приложения не будут работать должным образом.

13.7. WINE в многопользовательских установках FreeBSD

13.7.1. Проблемы при использовании общего префикса WINE

Как и большинство UNIX®-подобных операционных систем, FreeBSD разработана для одновременной работы нескольких пользователей. С другой стороны, Windows® является многопользовательской в том смысле, что в системе может быть создано несколько учётных записей. Однако предполагается, что физический компьютер (настольный или портативный ПК) в любой момент времени будет использовать только один пользователь.

Более новые несерверные версии Windows® предприняли некоторые шаги для улучшения ОС в сценариях с несколькими пользователями. Однако система по-прежнему в основном ориентирована на однопользовательскую работу. Более того, меры, принятые проектом WINE для создания совместимой среды, означают, что, в отличие от приложений FreeBSD (включая сам WINE), она будет напоминать эту однопользовательскую среду.

Следовательно, каждый пользователь должен будет поддерживать свой собственный набор конфигураций, что потенциально хорошо. Однако выгодно устанавливать приложения, особенно крупные, такие как офисные пакеты или игры, только один раз. Две причины для этого: обслуживание (обновления ПО нужно применять только один раз) и эффективность использования хранилища (отсутствие дублирования файлов).

Существует две стратегии для минимизации влияния множества пользователей WINE в системе.

13.7.2. Установка приложений на общий диск

Как показано в разделе о настройке WINE, WINE предоставляет возможность подключения дополнительных дисков к заданному префиксу. Таким образом, приложения можно устанавливать в общее место, в то время как у каждого пользователя останется префикс, где могут храниться индивидуальные настройки (в зависимости от программы). Это хорошая конфигурация, если нужно использовать совместно между пользователями относительно немного приложений, и это программы, которые требуют минимальных изменений в префиксе для работы.

Необходимо сделать следующие шаги для установки приложений таким способом:

1. Сначала настройте общее место в системе, где будут храниться файлы, например, /mnt/windows-drive_d/. Создание новых каталогов описано в руководстве mkdir(1).
2. Далее установите разрешения для этого нового каталога, чтобы разрешить доступ только нужным пользователям. Один из подходов — создать новую группу, например, "windows", добавить нужных пользователей в эту группу (см. подраздел о группах в разделе Пользователи и основы управления учётными записями) и установить разрешения для каталога в `770` (раздел Разрешения описывает этот процесс).
3. Наконец, добавьте выбранное расположение как диск в префикс пользователя с помощью `winecfg`, как описано в разделе выше о настройке WINE в этой главе.

По завершении приложения могут быть установлены в этом расположении и затем запущены с использованием назначенной буквы диска (или стандартного пути к каталогу в стиле UNIX®). Однако, как упоминалось выше, только один пользователь должен запускать эти приложения (которые могут обращаться к файлам в их каталоге установки) одновременно. Некоторые приложения также могут демонстрировать неожиданное поведение при запуске пользователем, который не является владельцем, несмотря на то что он входит в группу, которая должна иметь полные права "чтения/записи/исполнения" для всего каталога.

13.7.3. Использование общей установки WINE

С другой стороны, если требуется совместное использование множества приложений или они нуждаются в особой настройке для корректной работы, может потребоваться другой подход. В этом методе создаётся отдельный пользователь исключительно для хранения префикса WINE и всех установленных приложений. Затем отдельным пользователям предоставляется право запускать программы от имени этого пользователя с помощью команды sudo(8). В результате эти пользователи могут запускать приложения WINE как обычно, но они будут работать так, как если бы их запустил вновь созданный пользователь, и, следовательно, использовать централизованно поддерживаемый префикс, содержащий настройки и программы. Для реализации этого выполните следующие действия:

Создайте нового пользователя следующей командой (от имени `root`), которая выполнит все необходимые шаги:

```
# adduser
```

Введите имя пользователя (например, *windows*) и полное имя ("Microsoft Windows"). Затем примите значения по умолчанию для оставшихся вопросов. Далее установите утилиту `sudo` из бинарных пакетов с помощью следующей команды:

```
# pkg install sudo
```

После установки отредактируйте файл /etc/sudoers следующим образом:

```
# User alias specification

# define which users can run the wine/windows programs
User_Alias WINDOWS_USERS = user1,user2

# define which users can administrate (become root)
User_Alias ADMIN = user1

# Cmnd alias specification

# define which commands the WINDOWS_USERS may run
Cmnd_Alias WINDOWS = /usr/bin/wine,/usr/bin/winecfg
```

```
# Defaults
Defaults:WINDOWS_USERS env_reset
Defaults:WINDOWS_USERS env_keep += DISPLAY
Defaults:WINDOWS_USERS env_keep += XAUTHORITY
Defaults    !lecture,tty_tickets,!fqdn

# User privilege specification
root    ALL=(ALL) ALL

# Members of the admin user_alias, defined above, may gain root privileges
ADMIN ALL=(ALL) ALL

# The WINDOWS_USERS may run WINDOWS programs as user windows without a password
WINDOWS_USERS ALL = (windows) NOPASSWD: WINDOWS
```

Результатом этих изменений является разрешение пользователям, указанным в разделе *User_Alias*, запускать программы, перечисленные в разделе *Cmnd Alias*, используя ресурсы, указанные в разделе *Defaults* (текущий дисплей), как если бы они были пользователем, указанным в последней строке файла. Другими словами, пользователи, обозначенные как *WINDOWS_USERS*, могут запускать приложения WINE и `winecfg` от имени пользователя *windows*. В качестве бонуса данная конфигурация означает, что им не потребуется вводить пароль пользователя *windows*.

Далее предоставьте доступ к дисплею пользователю *windows*, от имени которого будут запускаться программы WINE:

```
% xhost +local:windows
```

Это следует добавить в список команд, выполняемых либо при входе в систему, либо при запуске графической среды по умолчанию. После завершения всех вышеуказанных действий пользователь, настроенный как один из `WINDOW_USERS` в sudoers, может запускать программы с использованием общего префикса следующей командой:

```
% sudo -u windows wine program.exe
```

Стоит отметить, что одновременный доступ нескольких пользователей к этой общей среде всё ещё сопряжён с рисками. Однако также учтите, что сама общая среда может содержать несколько префиксов. Таким образом, администратор может создать протестированный и проверенный набор программ, каждая со своим префиксом. В то же время один пользователь может играть в игру, а другой — работать с офисными программами без необходимости избыточной установки программного обеспечения.

13.8. Часто задаваемые вопросы о WINE на FreeBSD

Следующий раздел описывает некоторые часто задаваемые вопросы, советы, хитрости или распространённые проблемы при запуске WINE в FreeBSD, а также соответствующие ответы

на них.

13.8.1. Базовая установка и использование

Как установить 32-битный и 64-битный WINE в одной системе?

Как упоминалось ранее в этом разделе, пакеты wine и i386-wine конфликтуют друг с другом, поэтому их нельзя установить на одну систему обычным способом. Однако можно выполнить несколько установок, используя такие механизмы, как chroots/клетки, или собрав WINE из исходных кодов (обратите внимание, что это *не* означает сборку порта).

Можно ли запускать программы DOS в WINE?

Они могут работать как приложения с "Console User Interface", как упоминалось ранее в этом разделе. Однако существует, возможно, лучший способ запуска ПО для DOS: emulators/dosbox. С другой стороны, нет причин хотя бы не попробовать. Просто создайте новый префикс, установите программное обеспечение, и если оно не заработает — удалите префикс.

Нужно ли устанавливать пакет emulators/wine-devel для использования версии WINE для разработчиков вместо стабильной?

Да, установка этой версии приведет к установке "разрабатываемой" версии WINE. Как и в случае с 32- и 64-разрядными версиями, их нельзя установить вместе со стабильными версиями без принятия дополнительных мер.

Обратите внимание, что у WINE также есть "Staging" версия, которая содержит самые последние обновления. Ранее она была доступна как порт FreeBSD, но затем была удалена. Тем не менее, её можно скомпилировать непосредственно из исходного кода.

13.8.2. Оптимизации установленного WINE

Что делать с драйверами Windows® (например, графическими)?

Драйверы операционной системы передают команды между приложениями и оборудованием. WINE эмулирует среду Windows®, включая драйверы, которые, в свою очередь, используют собственные драйверы FreeBSD для этой передачи. Устанавливать драйверы Windows® не рекомендуется, так как система WINE предназначена для использования драйверов хостовой системы. Если, например, видеокарта требует специализированных драйверов, их следует устанавливать стандартными методами FreeBSD, а не установщиками Windows®.

Есть ли способ улучшить отображение шрифтов Windows®?

Пользователь на форумах FreeBSD предлагает следующую конфигурацию для исправления стандартного вида шрифтов в WINE, которые могут быть слегка пикселизированы.

Согласно сообщению на FreeBSD Forums, добавление следующего в .config/fontconfig/fonts.conf включит сглаживание и улучшит читаемость текста.

```
<?xml version="1.0"?>
<!DOCTYPE fontconfig SYSTEM "fonts.dtd>"

<fontconfig>

  <!-- antialias all fonts -->
  <match target="font">
    <edit name="antialias" mode="assign"><bool>true</bool></edit>>
    <edit name="hinting" mode="assign"><bool>true</bool></edit>>
    <edit name="hintstyle" mode="assign"><const>hintslight</const></edit>>
    <edit name="rgba" mode="assign"><const>rgb</const></edit>>
  </match>
</fontconfig>
```

Помогает ли наличие Windows®, установленной в другом месте системы, работе WINE?

В зависимости от запускаемого приложения это возможно. Как упоминалось в разделе, описывающем `winecfg`, некоторые встроенные DLL WINE и другие библиотеки могут быть заменены указанием пути к альтернативной версии. При условии, что раздел или диск Windows® подключен к системе FreeBSD и доступен пользователю, настройка некоторых из этих переопределений будет использовать родные библиотеки Windows®, что может снизить вероятность неожиданного поведения.

13.8.3. Зависимые от программы

Где лучше всего проверить, работает ли приложение X в WINE?

Первым шагом в определении совместимости должен быть WINE AppDB. Это сборник отчётов о работе (или неработоспособности) программ на всех поддерживаемых платформах, хотя (как упоминалось ранее) решения для одной платформы часто применимы и к другим.

Есть ли что-то, что поможет играм работать лучше?

Возможно. Многие игры для Windows® зависят от DirectX, проприетарного графического слоя Microsoft. Однако в сообществе открытого исходного кода существуют проекты, направленные на реализацию поддержки этой технологии.

Проект *dxvk*, представляющий собой попытку реализации DirectX с использованием совместимой с FreeBSD графической подсистемы Vulkan, является одним из таких. Хотя его основная цель — WINE на Linux, некоторые пользователи FreeBSD сообщают о компиляции и использовании dxvk.

В дополнение, ведётся работа над портом wine-proton. Это позволит использовать наработки Valve, разработчика игровой платформы Steam, в FreeBSD. Proton — это дистрибутив WINE, предназначенный для запуска многих игр под Windows® на других операционных системах с минимальными настройками.

Есть ли место, где пользователи FreeBSD WINE собираются для обмена советами и хитростями?

Существует множество мест, где пользователи FreeBSD обсуждают вопросы, связанные с WINE, и где можно поискать решения:

- Форумы FreeBSD, в частности разделы *Установка и обслуживание портов или пакетов* и *Эмуляция и виртуализация*.
- Каналы FreeBSD в IRC, включая #freebsd (для общих вопросов поддержки), #freebsd-games и другие.
- Discord-сервер BSD World с каналами, такими как *bsd-desktop*, *bsd-gaming*, *bsd-wine* и другими.

13.8.4. Ресурсы других ОС

Существует множество ресурсов, посвящённых другим операционным системам, которые могут быть полезны пользователям FreeBSD:

- Вики WINE содержит множество информации об использовании WINE, большая часть которой применима во многих поддерживаемых WINE операционных системах.
- Аналогично, документация других проектов операционных систем также может быть полезной. Страница о WINE в Arch Linux Wiki является особенно хорошим примером, хотя некоторые из «Сторонних приложений» (т.е. «вспомогательных приложений») очевидно недоступны в FreeBSD.
- Наконец, Codeweavers (разработчик коммерческой версии WINE) активно участвует в разработке основной версии. Часто ответы на вопросы в их форуме поддержки могут помочь в решении проблем с открытой версией WINE.

Часть III: Системное администрирование

Оставшиеся главы охватывают все аспекты администрирования системы FreeBSD. Каждая глава начинается с описания того, что будет изучено в результате прочтения, а также подробно указывает, что читатель должен знать перед изучением материала.

Эти главы предназначены для чтения по мере необходимости. Их не обязательно читать в каком-то определённом порядке, а также не требуется прочесть все перед началом работы с FreeBSD.

Глава 14. Конфигурация, сервисы, журналирование и управление питанием

14.1. Обзор

Одним из важных аспектов FreeBSD является правильная настройка системы. В этой главе описывается процесс настройки FreeBSD, включая некоторые параметры, которые можно задать для тонкой настройки системы FreeBSD.

Прежде чем читать эту главу, необходимо:

- Понимать основы UNIX® и FreeBSD (Основы FreeBSD).

Прочтите эту главу, чтобы узнать:

- Как использовать различные конфигурационные файлы в /etc.
- Основы настройки rc.conf и скриптов запуска в /usr/local/etc/rc.d.
- Как настроить FreeBSD с помощью переменных sysctl(8).
- Как настроить управление питанием в FreeBSD.

14.2. Файлы конфигурации

FreeBSD поддерживает чёткое разделение между базовой системой и сторонними приложениями, что влияет на расположение их конфигурационных файлов.

Конфигурация базовой системы FreeBSD находится в каталоге /etc, а в каталоге /usr/local/etc содержатся все конфигурационные файлы приложений, установленных в систему через коллекцию портов и пакеты.

Конфигурация состояния ядра находится в файле /etc/sysctl.conf. В разделе Утилита sysctl работа sysctl(8) будет рассмотрена более подробно.

Для получения дополнительной информации о структуре файловой системы FreeBSD обратитесь к hier(7).

Как правило, конфигурационные файлы не придерживаются единого стандарта в отношении синтаксиса. Хотя верно, что символ # обычно используется для комментирования строки и что каждая строка содержит переменную конфигурации.

Некоторые приложения, такие как pkg(8), начинают использовать Universal Configuration Language (UCL).

14.2.1. Каталог /etc

Каталог /etc содержит все файлы конфигурации базовой системы FreeBSD, которые отвечают за её настройку.

Крайнюю осторожность следует соблюдать при изменении файлов в каталоге /etc; неправильная настройка может сделать FreeBSD незагружаемой или неработоспособной.

/etc	Системные конфигурационные файлы и скрипты.
/etc/defaults	Файлы конфигурации системы по умолчанию, подробности см. в rc(8).
/etc/fstab	fstab(5) содержит описательную информацию о различных файловых системах.
/etc/mail	Дополнительная конфигурация sendmail(8) и другие файлы конфигурации MTA.
/etc/mtree	Файлы конфигурации mtree, подробнее см. mtree(8).
/etc/pam.d	Файлы конфигурации библиотеки Pluggable Authentication Modules (PAM).
/etc/periodic	Скрипты, выполняемые ежедневно, еженедельно и ежемесячно через cron(8). Подробнее см. periodic(8).
/etc/rc.d	Системные и демон-скрипты запуска/управления, подробнее см. в rc(8).
/etc/rc.conf	Содержит описательную информацию о локальном имени хоста, настройках сетевых интерфейсов и службах, которые должны запускаться при начальной загрузке системы. Подробнее в разделе Управление системными настройками
/etc/security	Файлы конфигурации аудита OpenBSM, дополнительную информацию смотрите в audit(8).
/etc/ppp	Файлы конфигурации ppp, подробности смотрите в ppp(8).
/etc/ssh	Файлы конфигурации OpenSSH, подробности см. в ssh(1).
/etc/ssl	Конфигурационные файлы OpenSSL.
/etc/sysctl.conf	Содержит настройки ядра. Дополнительная информация в Утилита sysctl

14.2.2. Утилита sysctl

Утилита sysctl(8) используется для внесения изменений в работающую систему FreeBSD.

Утилита sysctl(8) позволяет получать состояние ядра и, при наличии соответствующих привилегий, изменять его. Состояние, которое требуется получить или установить, описывается с использованием имени в стиле "Базы управляющей информации" ("MIB"), представленного в виде набора компонентов, разделённых точками.

Таблица 24. База управляющей информации

sysctl	«Волшебные» числа
kern	Ядро: функции и возможности
vm	Виртуальная память
vfs	Файловая система
net	Сеть
debug	Параметры отладки
hw	Оборудование
machdep	Зависимые от аппаратного обеспечения
user	Пользовательское пространство
p1003_1b	POSIX 1003.1B

В основе своей sysctl(8) выполняет две функции: чтение и изменение системных настроек.

Для просмотра всех доступных для чтения переменных:

```
% sysctl -a
```

Вывод должен быть похож на следующий:

```
kern.ostype: FreeBSD
...
vm.swap_enabled: 1
vm.overcommit: 0
vm.domain.0.pidctrl.kdd: 8
vm.domain.0.pidctrl.kid: 4
vm.domain.0.pidctrl.kpd: 3
...
vfs.zfs.sync_pass_rewrite: 2
vfs.zfs.sync_pass_dont_compress: 8
vfs.zfs.sync_pass_deferred_free: 2
```

Чтобы прочитать конкретную переменную, укажите её имя:

```
% sysctl kern.maxproc
```

Вывод должен быть похож на следующий:

```
kern.maxproc: 1044
```

База управляющей информации (MIB) иерархична, поэтому указание префикса выводит все узлы, зависящие от него:

```
% sysctl net
```

Вывод должен быть похож на следующий:

```
net.local.stream.recvspace: 8192
net.local.stream.sendspace: 8192
net.local.dgram.recvspace: 16384
net.local.dgram.maxdgram: 2048
net.local.seqpacket.recvspace: 8192
net.local.seqpacket.maxseqpacket: 8192
net.local.sockcount: 60
net.local.taskcount: 25
net.local.recycled: 0
net.local.deferred: 0
net.local.inflight: 0
net.inet.ip.portrange.randomtime: 1
net.inet.ip.portrange.randomcps: 9999
[...]
```

Чтобы установить конкретную переменную, используйте синтаксис *переменная=значение*:

```
# sysctl kern.maxfiles=5000
```

Вывод должен быть похож на следующий:

```
kern.maxfiles: 2088 -> 5000
```

Чтобы сохранить настройки после перезагрузки, необходимо добавить эти переменные в файл /etc/sysctl.conf, как описано ниже.

14.2.3. Файл /etc/sysctl.conf

Файл конфигурации для sysctl(8), /etc/sysctl.conf, выглядит очень похоже на /etc/rc.conf.

Значения задаются с использованием синтаксиса `переменная=значение`.

Указанные значения устанавливаются после перехода системы в многопользовательский режим. Не все переменные могут быть установлены в этом режиме.

Например, чтобы отключить запись завершений по фатальным сигналам и запретить пользователям видеть процессы, запущенные другими пользователями, в файле /etc/sysctl.conf можно установить следующие параметры:

```
# Do not log fatal signal exits (e.g., sig 11)
kern.logsigexit=0

# Prevent users from seeing information about processes that
# are being run under another UID.
security.bsd.see_other_uids=0
```

Чтобы получить дополнительную информацию о функции конкретного sysctl, можно выполнить следующую команду:

```
% sysctl -d kern.dfldsiz
```

Вывод должен быть похож на следующий:

```
kern.dfldsiz: Initial data size limit
```

14.2.4. Управление конфигурацией, специфичной для системы

Основное расположение информации о конфигурации системы — это /etc/rc.conf.

Этот файл содержит обширную информацию о конфигурации и читается при запуске системы для её настройки. Он предоставляет конфигурационную информацию для файлов rc*.

Записи в /etc/rc.conf переопределяют настройки по умолчанию из /etc/defaults/rc.conf.

Файл /etc/defaults/rc.conf, содержащий настройки по умолчанию, не следует редактировать. Вместо этого все специфичные для системы изменения должны вноситься в /etc/rc.conf.

В кластеризованных приложениях может быть применён ряд стратегий для разделения общесайтовой конфигурации и системно-специфичной конфигурации с целью снижения административной нагрузки.

Рекомендуемый подход заключается в размещении специфичной для системы конфигурации в /etc/rc.conf.local.

Например, эти записи в /etc/rc.conf применяются ко всем системам:

```
sshd_enable="YES"
keyrate="fast"
```

```
defaultrouter="10.1.1.254"
```

В то время как эти записи в /etc/rc.conf.local применяются только к этой системе:

```
hostname="node1.example.org"
ifconfig_fxp0="inet 10.1.1.1/8"
```

Распределите /etc/rc.conf на каждую систему с помощью таких приложений, как rsync или puppet, в то время как /etc/rc.conf.local остаётся уникальным.

Обновление системы не перезапишет /etc/rc.conf, поэтому системные настройки не будут потеряны.

Оба файла /etc/rc.conf и /etc/rc.conf.local обрабатываются sh(1). Это позволяет системным операторам создавать сложные сценарии конфигурации. Дополнительную информацию по этой теме можно найти в rc.conf(5).

14.3. Управление службами в FreeBSD

FreeBSD использует систему стартовых сценариев rc(8) для инициализации системы и управления службами.

Скрипты, перечисленные в /etc/rc.d, предоставляют базовые сервисы, которыми можно управлять с помощью опций `start`, `stop` и `restart` команды service(8).

Базовый скрипт может выглядеть следующим образом:

```
#!/bin/sh
#
# PROVIDE: utility
# REQUIRE: DAEMON
# KEYWORD: shutdown

. /etc/rc.subr

name=utility
rcvar=utility_enable

command="/usr/local/sbin/utility"

load_rc_config $name

#
# DO NOT CHANGE THESE DEFAULT VALUES HERE
# SET THEM IN THE /etc/rc.conf FILE
#
utility_enable=${utility_enable-"NO"}
pidfile=${utility_pidfile-"/var/run/utility.pid"}
```

```
run_rc_command "$1"
```

Обратитесь к этой статье для получения инструкций по созданию пользовательских сценариев rc(8).

14.3.1. Запуск служб

Многие пользователи устанавливают стороннее программное обеспечение на FreeBSD из Коллекции портов и требуют, чтобы установленные службы запускались при инициализации системы.

Службы, такие как security/openssh-portable или www/nginx, являются лишь двумя из множества программных пакетов, которые могут запускаться при инициализации системы. В этом разделе описаны доступные процедуры для запуска служб.

Поскольку система rc(8) в первую очередь предназначена для запуска и остановки служб во время загрузки и выключения системы, опции `start`, `stop` и `restart` выполнят соответствующие действия только в том случае, если установлена соответствующая переменная в /etc/rc.conf.

Итак, первый шаг для запуска службы, например www/nginx, — это добавить её в /etc/rc.conf, выполнив следующую команду:

```
# sysrc nginx_enable="YES"
```

Затем nginx можно запустить, выполнив следующую команду:

```
# service nginx start
```

Для запуска (`start`), остановки (`stop`) или перезапуска (`restart`) службы независимо от настроек в /etc/rc.conf, перед этими командами следует добавить "one". Например, чтобы запустить www/nginx независимо от текущих настроек в /etc/rc.conf, выполните следующую команду:

```
# service nginx onestart
```

Также возможно автоматическое размещение службы в jail, см. соответствующее пояснение в разделе Сервисные клетки.

14.3.2. Состояние службы

Чтобы определить, запущена ли служба, используйте подкоманду `status`.

Например, чтобы проверить, что www/nginx работает:

```
# service nginx status
```

Вывод должен быть похож на следующий:

```
nginx is running as pid 27871.
```

14.3.3. Перезагрузить службу

В некоторых случаях также можно `перезагрузить` службу. Это попытка отправить сигнал отдельной службе, заставляя её перезагрузить свои конфигурационные файлы.

В большинстве случаев это означает отправку службе сигнала `SIGHUP`.

Не все службы поддерживают эту возможность.

Система rc(8) используется для сетевых сервисов, а также играет важную роль в большинстве процессов инициализации системы. Например, при выполнении скрипта /etc/rc.d/bgfsck выводится следующее сообщение:

```
Starting background file system checks in 60 seconds.
```

Этот скрипт используется для фоновой проверки файловых систем, которая выполняется только во время инициализации системы.

Многие системные службы зависят от других служб для корректной работы. Например, yp(8) и другие службы на основе RPC могут не запуститься, пока не будет запущена служба rpcbind(8).

Дополнительную информацию можно найти в rc(8) и rc.subr(8).

14.3.4. Использование служб для запуска сервисов

Другие службы могут быть запущены с помощью inetd(8). Работа с inetd(8) и его настройка подробно описаны в «Суперсервер inetd».

В некоторых случаях может быть целесообразнее использовать cron(8) для запуска системных служб. Такой подход имеет ряд преимуществ, поскольку cron(8) запускает эти процессы от имени владельца crontab(5). Это позволяет обычным пользователям запускать и поддерживать свои собственные приложения.

Функция `@reboot` в cron(8) может быть использована вместо указания времени. Это приводит к запуску задачи при старте cron(8), обычно во время инициализации системы.

14.4. Cron и Periodic

Планирование задач на определённый день или время — очень распространённая задача в

FreeBSD. Инструмент, отвечающий за выполнение этой задачи, — это cron(8).

В дополнение к задачам, которые пользователь может запланировать с помощью cron(8), FreeBSD выполняет фоновые задачи, управляемые periodic(8).

14.4.1. Cron

Утилита cron(8) работает в фоновом режиме и периодически проверяет файл /etc/crontab на наличие задач для выполнения, а также ищет пользовательские файлы crontab в каталоге /var/cron/tabs.

Эти файлы используются для планирования задач, которые cron запускает в указанное время.

Каждая запись в crontab определяет задачу для выполнения и называется *cron job*.

Используются два типа конфигурационных файлов: системный crontab, который не следует изменять, и пользовательские crontabs, которые можно создавать и редактировать по необходимости. Формат этих файлов описан в crontab(5). Формат системного crontab, /etc/crontab, включает столбец `who`, который отсутствует в пользовательских crontabs. В системном crontab команды выполняются от имени пользователя, указанного в этом столбце. В пользовательском crontab все команды выполняются от имени пользователя, создавшего crontab.

Пользовательские crontab-файлы позволяют отдельным пользователям планировать свои собственные задачи. У пользователя `root` также может быть пользовательский crontab, который можно использовать для планирования задач, отсутствующих в системном crontab.

Вот пример записи из системного crontab, /etc/crontab:

```
# /etc/crontab - root's crontab for FreeBSD
#
①
#
SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin ②
#
#minute hour    mday    month   wday    who     command ③
#
# Save some entropy so that /dev/random can re-seed on boot.
*/11    *       *       *       *       operator /usr/libexec/save-entropy ④
#
# Rotate log files every hour, if necessary.
0       *       *       *       *       root    newsyslog
#
# Perform daily/weekly/monthly maintenance.
1       3       *       *       *       root    periodic daily
15      4       *       *       6       root    periodic weekly
30      5       1       *       *       root    periodic monthly
```

```
#
# Adjust the time zone if the CMOS clock keeps local time, as opposed to
# UTC time.  See adjkerntz(8) for details.
1,31    0-5     *       *       *       root    adjkerntz -a
```

① Строки, начинающиеся с символа `#`, являются комментариями. Комментарий можно добавить в файл как напоминание о том, что и зачем выполняется. Комментарии не могут находиться на одной строке с командой, иначе они будут восприняты как часть команды; они должны быть на отдельной строке. Пустые строки игнорируются.

② Символ равенства (`=`) используется для определения параметров окружения. В данном примере он применяется для задания переменных `SHELL` и `PATH`. Если `SHELL` не указана, cron будет использовать оболочку POSIX® по умолчанию. Если `PATH` не указан, необходимо указывать полный путь к команде или скрипту, который требуется выполнить.

③ Эта строка определяет семь полей, используемых в системном crontab: `minute`, `hour`, `mday`, `month`, `wday`, `who` и `command`. Поле `minute` указывает время в минутах, когда должна выполняться указанная команда, `hour` — час, `mday` — день месяца, `month` — месяц, а `wday` — день недели. Эти поля должны содержать числовые значения в 24-часовом формате или символ `*`, обозначающий все возможные значения для данного поля. Поле `who` существует только в системном crontab и указывает, от имени какого пользователя должна выполняться команда. Последнее поле — это команда, которую нужно выполнить.

④ Эта запись определяет значения для этого задания cron. Комбинация `*/11`, за которой следует несколько символов `*`, указывает, что `/usr/libexec/save-entropy` запускается от имени пользователя `operator` каждые одиннадцать минут каждого часа, каждого дня и дня недели, каждого месяца. Команды могут включать любое количество параметров. Однако команды, занимающие несколько строк, должны быть разделены символом продолжения обратной косой черты "\".

14.4.2. Создание пользовательского Crontab

Чтобы создать пользовательский crontab, вызовите `crontab` в режиме редактора:

```
% crontab -e
```

Это откроет crontab пользователя в текстовом редакторе по умолчанию. При первом запуске этой команды откроется пустой файл. После создания crontab эта команда будет открывать его для редактирования.

Полезно добавить следующие строки в начало файла crontab, чтобы установить переменные окружения и запомнить назначение полей в crontab:

```
SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin
# Order of crontab fields
# minute hour mday month wday command
```

Затем добавьте строку для каждой команды или скрипта, указав время их выполнения. В этом примере указанный пользовательский скрипт оболочки будет запускаться каждый день в два часа дня. Поскольку путь к скрипту не указан в `PATH`, указывается полный путь к скрипту:

```
0 14 * * * /home/user/bin/mycustomscript.sh
```

Перед использованием пользовательского скрипта убедитесь, что он исполняемый, и протестируйте его с ограниченным набором переменных окружения, установленных cron. Чтобы воспроизвести окружение, которое будет использоваться для выполнения вышеуказанной записи cron, используйте:

```
env -i SHELL=/bin/sh PATH=/etc:/bin:/sbin:/usr/bin:/usr/sbin
HOME=/home/user LOGNAME=user /home/user/bin/mycustomscript.sh
```

Окружение, устанавливаемое cron, описано в crontab(5). Особенно важно проверять корректность работы скриптов в окружении cron, если они содержат команды удаления файлов с использованием шаблонов.

После завершения редактирования файла crontab сохраните его. Он будет автоматически установлен, и cron прочитает crontab и запустит задания по расписанию. Для просмотра списка заданий в crontab используйте следующую команду:

```
% crontab -l
```

Вывод должен быть похож на следующий:

```
0 14 * * * /home/user/bin/mycustomscript.sh
```

Удалить все задания cron из пользовательского crontab:

```
% crontab -r
```

Вывод должен быть похож на следующий:

```
remove crontab for user? y
```

14.4.3. Периодические задачи (Periodic)

FreeBSD предоставляет набор скриптов для управления системой, которые проверяют состояние различных подсистем, выполняют проверки, связанные с безопасностью,

осуществляют ротацию файлов журналов и т.д. Эти скрипты запускаются периодически: ежедневно, еженедельно или ежемесячно. Управление этими задачами осуществляется с помощью periodic(8), а его конфигурация находится в periodic.conf(5). Периодические задачи инициируются записями в системном crontab, как показано выше.

Скрипты, выполняемые periodic(8), расположены в /etc/periodic/ для базовых утилит и в /usr/local/etc/periodic/ для стороннего программного обеспечения.

Они организованы в 4 подкаталога: daily, weekly, monthly и security.

14.4.4. Включение или отключение периодических задач

FreeBSD имеет некоторые скрипты, включённые по умолчанию для периодического выполнения.

Чтобы включить или отключить задачу, первым шагом необходимо отредактировать файл /etc/periodic.conf, выполнив следующую команду:

```
# ee /etc/periodic.conf
```

И затем, чтобы включить, например, `daily_status_zfs_enable`, поместите следующее содержимое в файл:

```
daily_status_zfs_enable="YES"
```

Чтобы отключить задание, которое активно по умолчанию, достаточно изменить `YES` на `NO`.

14.4.5. Настройка вывода периодических задач

В файле /etc/periodic.conf переменные `daily_output`, `weekly_output` и `monthly_output` определяют, куда отправлять результаты выполнения скриптов.

По умолчанию результаты работы периодических скриптов отправляются по электронной почте пользователю root, поэтому рекомендуется либо читать почту root, либо настроить пересылку писем root на почтовый ящик, который регулярно проверяется.

Чтобы отправить результаты на другой адрес электронной почты или на несколько адресов, добавьте email-адреса через пробел в файл /etc/periodic.conf:

```
daily_output="email1@example.com email2@example.com"
weekly_output="email1@example.com email2@example.com"
monthly_output="email1@example.com email2@example.com"
```

Для записи периодического вывода в журнал вместо получения его по электронной почте добавьте следующие строки в /etc/periodic.conf. Утилита newsyslog(8) будет выполнять ротацию этих файлов в соответствующее время:

```
daily_output=/var/log/daily.log
weekly_output=/var/log/weekly.log
monthly_output=/var/log/monthly.log
```

14.5. Настройка системного журналирования

Генерация и чтение системных журналов — важный аспект администрирования системы. Информация в системных журналах может использоваться для выявления проблем с оборудованием и программным обеспечением, а также ошибок в конфигурации приложений и системы. Эти данные также играют важную роль в аудите безопасности и реагировании на инциденты. Большинство системных демонов и приложений создают записи в журналах.

FreeBSD предоставляет системный модуль журналирования syslogd(8) для управления журналированием. По умолчанию syslogd включен и запускается при загрузке системы.

В этом разделе описывается, как настроить системный модуль журналирования FreeBSD для локального и удалённого ведения журналов, а также как выполнять ротацию и управление журналами.

14.5.1. Настройка локального журналирования

Файл конфигурации /etc/syslog.conf определяет, как syslogd обрабатывает поступающие записи журналов. Существует несколько параметров для управления обработкой входящих событий. *Facility* (источник) указывает, какая подсистема сгенерировала сообщение, например, ядро или демон, а *level* (уровень) описывает степень серьезности произошедшего события. Это позволяет настроить, будет ли сообщение зарегистрировано и куда, в зависимости от категории и уровня. Также можно выполнять действия в зависимости от приложения, отправившего сообщение, а в случае удалённого журналирования — от имени хоста машины, генерирующей событие.

Этот файл конфигурации содержит по одной строке для каждого действия, где синтаксис каждой строки представляет собой поле селектора, за которым следует поле действия. Синтаксис поля селектора — *facility.level*, что соответствует журнальным сообщениям от *facility* уровня *level* и выше. Также можно добавить необязательный флаг сравнения перед уровнем, чтобы точнее указать, что регистрируется. Для одного действия можно использовать несколько полей селектора, разделённых точкой с запятой (`;`). Использование `*` соответствует всем сообщениям. Поле действия указывает, куда отправлять журнальное сообщение, например, в файл или на удалённый хост для журналирования.

В качестве примера приведён стандартный файл /etc/syslog.conf из FreeBSD:

```
#       Spaces ARE valid field separators in this file. However,
#       other *nix-like systems still insist on using tabs as field
#       separators. If you are sharing this file between systems, you
#       may want to use only tabs as field separators here.
#       Consult the syslog.conf(5) manpage.
```

```
*.err;kern.warning;auth.notice;mail.crit                /dev/console ①
*.notice;authpriv.none;kern.debug;lpr.info;mail.crit;news.err   /var/log/messages
security.*                                              /var/log/security
auth.info;authpriv.info                                 /var/log/auth.log
mail.info                                               /var/log/maillog ②
cron.*                                                  /var/log/cron
!-devd
*.=debug                                                /var/log/debug.log ③
*.emerg                                                 *
daemon.info                                             /var/log/daemon.log
# uncomment this to log all writes to /dev/console to /var/log/console.log
# touch /var/log/console.log and chmod it to mode 600 before it will work
#console.info                                           /var/log/console.log
# uncomment this to enable logging of all log messages to /var/log/all.log
# touch /var/log/all.log and chmod it to mode 600 before it will work
#*.*                                                    /var/log/all.log
# uncomment this to enable logging to a remote loghost named loghost
#*.*                                                    @loghost
# uncomment these if you're running inn
# news.crit                                             /var/log/news/news.crit
# news.err                                              /var/log/news/news.err
# news.notice                                           /var/log/news/news.notice
# Uncomment this if you wish to see messages produced by devd
# !devd
# *.>=notice                                            /var/log/devd.log ④
!*
include                                                 /etc/syslog.d
include                                                 /usr/local/etc/syslog.d
```

① Соответствует всем сообщениям с уровнем `err` или выше, а также `kern.warning`, `auth.notice` и `mail.crit`, и отправляет эти сообщения журнала на консоль (/dev/console).

② Соответствует всем сообщениям из источника `mail` уровня `info` и выше и записывает их в /var/log/maillog.

③ Использует флаг сравнения (`=`) для совпадения только с сообщениями уровня `debug` и записывает их в /var/log/debug.log.

④ Пример использования спецификации программы. Это делает следующие правила действительными только для указанной программы. В данном случае только сообщения, генерируемые devd(8), записываются в /var/log/devd.log.

Для получения дополнительной информации о /etc/syslog.conf, его синтаксисе и более сложных примерах использования см. syslog.conf(5).

14.5.2. Источники системы журналирования

Источник описывает часть системы, генерирующую сообщение. Источники — это способ разделения различных сообщений, чтобы пользователю было проще анализировать журналы.

Таблица 25. Источники системных журналов (syslog)

Имя	Описание
auth	Система авторизации: login(1), su(1), getty(8) и т.д.
authpriv	То же, что и auth, но записывается в файл, доступный только для чтения root.
console	Сообщения, записываемые драйвером вывода консоли ядра в /dev/console.
cron	Сообщения, записываемые демоном cron(8).
daemon	Системные демоны, такие как routed(8), которые не предусмотрены явно другими средствами.
ftp	Демоны протокола передачи файлов: ftpd(8), tftpd(8).
kern	Сообщения, генерируемые ядром. Они не могут быть созданы какими-либо пользовательскими процессами.
lpr	Система буферизации принтеров: lpr(1), lpc(8), lpd(8) и т.д.
mail	Почтовая система.
mark	Этот механизм добавляет запись каждые 20 минут.
news	Система сетевых новостей.
ntp	Система протокола сетевого времени.
security	Подсистемы безопасности, такие как ipfw(4).
syslog	Сообщения, генерируемые внутренне syslogd(8).
user	Сообщения, генерируемые случайными пользовательскими процессами. **Это идентификатор источника по умолчанию, если не указан другой**.
uucp	Система Unix-to-Unix Copy. Устаревший протокол. Очень странно видеть сообщения от этого сервиса.
от `local0` до `local7`	Зарезервировано для локального использования.

14.5.3. Уровни журналирования

Уровень описывает степень важности сообщения и является ключевым словом из следующего упорядоченного списка (от высшего к низшему):

Таблица 26. Уровни в syslog

Имя	Описание
emerg	Паника. Обычно это сообщение рассылается всем пользователям.
alert	Условие, которое должно быть немедленно исправлено, например, повреждение системной базы данных.
crit	Критические условия, например, аппаратные ошибки устройств.
err	Ошибки.
warning	Предупреждающие сообщения.
notice	Условия, которые не являются ошибочными, но, возможно, требуют специальной обработки.
info	Информационные сообщения.
debug	Сообщения, содержащие информацию, которая обычно полезна только при отладке программы.
none	Этот специальный уровень отключает определённую функцию.

14.5.4. Как читать сообщения журналов

По умолчанию журнальные файлы FreeBSD используют формат rfc3164, также известный как протокол BSD syslog. Подробнее о других форматах и их использовании можно узнать в syslog(8).

Обычно журналы имеют следующий синтаксис:

```
date time hostname program[pid]: the message
```

В качестве примера будет использоваться вывод файла /var/log/cron:

```
[...]
Jul 16 12:40:00 FreeBSD /usr/sbin/cron[81519]: (root) CMD (/usr/libexec/atrun)
Jul 16 12:44:00 FreeBSD /usr/sbin/cron[83072]: (operator) CMD (/usr/libexec/save-
entropy)
[...]
```

Подробное журналирование, при котором к каждому сообщению добавляются данные о facility и level, можно включить в syslog(8), выполнив следующую команду:

```
# sysrc syslogd_flags="-vv"
```

После активации функции источник и уровень будут отображаться в журнале, как показано в следующем примере:

```
[...]
Jul 16 17:40:00 <cron.info> FreeBSD /usr/sbin/cron[1016]: (root) CMD
(/usr/libexec/atrun)
Jul 16 17:44:00 <cron.info> FreeBSD /usr/sbin/cron[1030]: (operator) CMD
(/usr/libexec/save-entropy)
[...]
```

14.5.5. Управление журналами и их ротация

Файлы журналов могут быстро увеличиваться в размере, занимая место на диске и затрудняя поиск полезной информации.

В FreeBSD для управления файлами журналов и предотвращения этой проблемы используется newsyslog(8).

Эта встроенная программа периодически выполняет ротацию и сжатие файлов журналов, а также при необходимости создаёт отсутствующие файлы журналов и отправляет сигналы программам при перемещении файлов журналов.

Поскольку newsyslog запускается через cron(8), он не может выполнять ротацию файлов чаще, чем запланировано в cron(8). В стандартной конфигурации он запускается каждый час.

Вот стандартная конфигурация в FreeBSD, подробнее можно узнать в newsyslog.conf(5):

```
# configuration file for newsyslog
#
# Entries which do not specify the '/pid_file' field will cause the
# syslogd process to be signalled when that log file is rotated.  This
# action is only appropriate for log files which are written to by the
# syslogd process (ie, files listed in /etc/syslog.conf).  If there
# is no process which needs to be signalled when a given log file is
# rotated, then the entry for that file should include the 'N' flag.
#
# Note: some sites will want to select more restrictive protections than the
# defaults.  In particular, it may be desirable to switch many of the 644
# entries to 640 or 600.  For example, some sites will consider the
# contents of maillog, messages, and lpd-errs to be confidential.  In the
# future, these defaults may change to more conservative ones.
#
# logfilename          [owner:group]    mode count size when  flags [/pid_file]
[sig_num]
```

```
/var/log/all.log                        600  7     *    @T00  J
/var/log/auth.log                       600  7     1000 @0101T JC
/var/log/console.log                    600  5     1000 *     J
/var/log/cron                           600  3     1000 *     JC
/var/log/daily.log                      640  7     *    @T00  JN
/var/log/debug.log                      600  7     1000 *     JC
/var/log/init.log                       644  3     1000 *     J
/var/log/kerberos.log                   600  7     1000 *     J
/var/log/maillog                        640  7     *    @T00  JC
/var/log/messages                       644  5     1000 @0101T JC
/var/log/monthly.log                    640  12    *    $M1D0 JN
/var/log/devd.log                       644  3     1000 *     JC
/var/log/security                       600  10    1000 *     JC
/var/log/utx.log                        644  3     *    @01T05 B
/var/log/weekly.log                     640  5     *    $W6D0 JN
/var/log/daemon.log                     644  5     1000 @0101T JC

<include> /etc/newsyslog.conf.d/[!.]*.conf
<include> /usr/local/etc/newsyslog.conf.d/[!.]*.conf
```

1. `logfilename` - Имя файла системного журнала, предназначенного для архивирования.
2. `[owner:group]` - Это необязательное поле указывает владельца и группу для архивируемого файла.
3. `mode` - Указывает режим доступа к файлу журнала и архивам. Допустимые биты режима - 0666. (То есть, права на чтение и запись для ротированного журнала могут быть заданы для владельца, группы и остальных пользователей.)
4. `count` - Указывает максимальное количество архивных файлов, которые могут существовать.
5. `size` — Когда размер файла журнала достигает указанного размера в килобайтах, файл журнала будет обрезан, как описано выше. Если в этом поле указана звёздочка ('*'), файл журнала не будет обрезаться по размеру.
6. `when` - Может состоять из интервала, конкретного времени или обоих вариантов. Поддерживаемые опции описаны в newsyslog.conf(5).
7. `flags` - Указывает флаги, которые принимает newsyslog, поддерживаемые опции описаны в newsyslog.conf(5).
8. `[/pid_file]` - Это необязательное поле указывает имя файла, содержащего идентификатор процесса (PID) демона или идентификатор группы процессов.
9. `[sig_num]` - Это необязательное поле указывает сигнал, который будет отправлен процессу демона.

Последние два поля необязательны и указывают имя файла ID процесса (PID) и номер сигнала, который будет отправлен этому процессу при ротации файла.

14.5.6. Настройка удалённого журналирования

Мониторинг журналов событий на нескольких хостах может стать громоздким по мере увеличения количества систем. Настройка централизованного журналирования позволяет снизить нагрузку на администратора при управлении журналами.

В FreeBSD централизованная агрегация, объединение и ротация файлов журналов могут быть настроены с помощью `syslogd` и `newsyslog`.

В этом разделе приведён пример конфигурации, в которой хост `A` с именем `logserv.example.com` будет собирать информацию журналирования для локальной сети.

Узел `B` с именем `logclient.example.com` будет настроен для передачи информации журналирования на сервер журналирования.

Конфигурация сервера журналов

Сервер журналов — это система, настроенная для приёма информации журналирования с других узлов.

Перед настройкой сервера журналов проверьте следующее:

- Если между сервером журналирования и клиентами журналирования есть межсетевой экран, убедитесь, что набор правил межсетевого экрана разрешает UDP-порт 514 как для клиентов, так и для сервера.
- Сервер журналирования и все клиентские машины должны иметь прямые и обратные записи в локальной DNS. Если в сети нет DNS-сервера, создайте записи в файле /etc/hosts каждой системы. Корректное разрешение имён необходимо для того, чтобы записи журналов не отклонялись сервером журналирования.

На сервере журналов отредактируйте файл /etc/syslog.conf, указав имя клиента, от которого будут приниматься записи журнала, используемое средство ведения журнала и имя файла для хранения записей журнала данного хоста. В этом примере добавляется имя хоста `B`, регистрируются все средства и записи журнала сохраняются в файле /var/log/logclient.log.

Пример 22. Пример конфигурации сервера журналов

```
+logclient.example.com
*.*     /var/log/logclient.log
```

При добавлении нескольких клиентов для журналирования добавьте аналогичную запись из двух строк для каждого клиента. Дополнительную информацию о доступных средствах можно найти в syslog.conf(5).

Далее выполните следующие команды:

```
# sysrc syslogd_enable="YES"
```

```
# sysrc syslogd_flags="-a logclient.example.com -v -v"
```

Первый параметр запускает syslogd при загрузке системы. Второй параметр разрешает записи журналов от указанного клиента. Опция `-v -v` увеличивает подробность записываемых сообщений. Это полезно для настройки категорий, так как администраторы могут видеть, какие типы сообщений записываются в каждой категории.

Можно указать несколько параметров `-a`, чтобы разрешить ведение журнала от нескольких клиентов. Также можно указывать IP-адреса и целые сетевые блоки. Полный список возможных параметров приведён в syslogd(8).

Наконец, создайте файл журнала:

```
# touch /var/log/logclient.log
```

На этом этапе следует перезапустить и проверить syslogd:

```
# service syslogd restart
# pgrep syslog
```

Если возвращен PID, сервер успешно перезапущен, и можно приступать к настройке клиента. Если сервер не перезапустился, проверьте ошибку в /var/log/messages.

Конфигурация клиента журналирования

Клиент системы журналирования отправляет записи журналов на сервер журналирования в сети. Клиент также сохраняет локальную копию своих собственных журналов.

После настройки сервера журналирования выполните следующие команды на клиенте журналирования:

```
# sysrc syslogd_enable="YES"
# sysrc syslogd_flags="-s -v -v"
```

Первый параметр включает запуск syslogd при загрузке системы. Второй параметр запрещает получение журналов от других хостов (`-s`) и увеличивает детализацию записываемых сообщений.

Затем определите сервер журналирования в файле /etc/syslog.conf клиента. В этом примере все журналируемые источники отправляются на удалённую систему, обозначенную символом `@`, с указанным именем хоста:

```
*.*  @logserv.example.com
```

После сохранения изменений перезапустите syslogd, чтобы изменения вступили в силу:

```
# service syslogd restart
```

Для проверки отправки журнальных сообщений по сети используйте logger(1) на клиенте, чтобы отправить сообщение в syslogd:

```
# logger "Test message from logclient"
```

Это сообщение теперь должно присутствовать как в /var/log/messages на клиенте, так и в /var/log/logclient.log на сервере журналов.

Отладка серверов журналов

Если на сервере журналов не поступают сообщения, скорее всего, причина в проблеме с сетевым подключением, разрешении имён или опечатке в конфигурационном файле. Чтобы выявить причину, убедитесь, что и сервер журналов, и клиент могут `ping` друг друга, используя имя хоста, указанное в их /etc/rc.conf. Если это не удаётся, проверьте сетевые кабели, набор правил межсетевого экрана и записи имён хостов на DNS-сервере или в /etc/hosts как на сервере журналов, так и на клиентах. Повторяйте проверки, пока `ping` не будет успешным с обоих хостов.

Если команда `ping` выполняется успешно на обоих хостах, но сообщения журнала по-прежнему не поступают, временно увеличьте уровень детализации журналирования, чтобы сузить круг поиска проблемы в конфигурации. В следующем примере файл /var/log/logclient.log на сервере журналирования пуст, а файл /var/log/messages на клиенте журналирования не указывает на причину сбоя.

Для увеличения уровня отладочной информации отредактируйте параметр `syslogd_flags` на сервере журналирования и выполните перезапуск:

```
sysrc syslogd_flags="-d -a logclient.example.com -v -v"
```

```
# service syslogd restart
```

На консоли сразу после перезагрузки появится отладочная информация, подобная следующей:

```
logmsg: pri 56, flags 4, from logserv.example.com, msg syslogd: restart
syslogd: restarted
logmsg: pri 6, flags 4, from logserv.example.com, msg syslogd: kernel boot file is
/boot/kernel/kernel
Logging to FILE /var/log/messages
syslogd: kernel boot file is /boot/kernel/kernel
cvthname(192.168.1.10)
validate: dgram from IP 192.168.1.10, port 514, name logclient.example.com;
```

```
rejected in rule 0 due to name mismatch.
```

В этом примере сообщения журнала отклоняются из-за опечатки, которая приводит к несоответствию имени хоста. Имя хоста клиента должно быть `logclient`, а не `logclien`. Исправьте опечатку, выполните перезапуск и проверьте результат:

```
# service syslogd restart
```

Вывод должен быть похож на следующий:

```
logmsg: pri 56, flags 4, from logserv.example.com, msg syslogd: restart
syslogd: restarted
logmsg: pri 6, flags 4, from logserv.example.com, msg syslogd: kernel boot file is
/boot/kernel/kernel
syslogd: kernel boot file is /boot/kernel/kernel
logmsg: pri 166, flags 17, from logserv.example.com,
msg Dec 10 20:55:02 <syslog.err> logserv.example.com syslogd: exiting on signal 2
cvthname(192.168.1.10)
validate: dgram from IP 192.168.1.10, port 514, name logclient.example.com;
accepted in rule 0.
logmsg: pri 15, flags 0, from logclient.example.com, msg Dec 11 02:01:28 trhodes: Test
message 2
Logging to FILE /var/log/logclient.log
Logging to FILE /var/log/messages
```

На данном этапе сообщения правильно принимаются и помещаются в нужный файл.

Безопасность

Как и в случае с любой сетевой службой, перед внедрением сервера журналирования следует учитывать требования безопасности. Журналы могут содержать конфиденциальные данные о службах, включённых на локальном хосте, учётных записях пользователей и конфигурации. Данные, передаваемые по сети от клиента к серверу, не шифруются и не защищаются паролем. Если требуется шифрование, можно использовать security/stunnel, который передаёт данные журналирования через зашифрованный туннель.

Локальная безопасность также является важным вопросом. Журналы не шифруются ни во время использования, ни после ротации. Локальные пользователи могут получить доступ к файлам журналов для сбора дополнительной информации о конфигурации системы. Установка правильных разрешений для файлов журналов крайне важна. Встроенный ротатор журналов newsyslog поддерживает установку разрешений для вновь создаваемых и ротируемых файлов журналов. Установка режима `600` для файлов журналов должна предотвратить нежелательный доступ со стороны локальных пользователей. Дополнительную информацию можно найти в newsyslog.conf(5).

14.6. Управление питанием и ресурсами

Важно эффективно использовать аппаратные ресурсы. Управление питанием и ресурсами позволяет операционной системе отслеживать системные ограничения и, возможно, выполнять некоторые действия, вызванные событиями, связанными с этими ограничениями.

14.6.1. Конфигурация ACPI

На FreeBSD управление этими ресурсами осуществляется с помощью устройства ядра acpi(4).

В FreeBSD драйвер acpi(4) загружается по умолчанию при загрузке системы.

Этот драйвер **нельзя выгрузить после загрузки**, так как системная шина использует его для различных взаимодействий с оборудованием.

В дополнение к acpi(4), FreeBSD имеет несколько специализированных модулей ядра для различных подсистем ACPI от производителей. Эти модули добавляют дополнительную функциональность, такую как управление скоростью вентилятора, подсветкой клавиатуры или яркостью экрана.

Список можно получить, выполнив следующую команду:

```
% ls /boot/kernel | grep acpi
```

Вывод должен быть похож на следующий:

```
acpi_asus.ko
acpi_asus_wmi.ko
acpi_dock.ko
acpi_fujitsu.ko
acpi_hp.ko
acpi_ibm.ko
acpi_panasonic.ko
acpi_sony.ko
acpi_toshiba.ko
acpi_video.ko
acpi_wmi.ko
sdhci_acpi.ko
uacpi.ko
```

В случае, если используется ноутбук IBM/Lenovo, необходимо загрузить модуль acpi_ibm(4), выполнив следующую команду:

```
# kldload acpi_ibm
```

И добавьте эту строку в /boot/loader.conf, чтобы загружать модуль при запуске:

```
acpi_ibm_load="YES"
```

Альтернативой модулю acpi_video(4) является драйвер backlight(9). Он предоставляет универсальный способ управления подсветкой экрана. Этот драйвер включён в стандартное ядро GENERIC. Утилита backlight(8) позволяет запрашивать и изменять яркость подсветки экрана. В этом примере яркость уменьшается на 10%:

```
% backlight decr 10
```

14.6.2. Управление питанием процессора

CPU — это наиболее энергозатратная часть системы. Знание того, как повысить эффективность использования CPU, является важной частью нашей системы для экономии энергии.

Для правильного и эффективного использования ресурсов системы FreeBSD поддерживает такие технологии, как Intel Turbo Boost, AMD Turbo Core, Intel Speed Shift и другие, с помощью powerd(8) и cpufreq(4).

Первым шагом будет получение информации о процессоре с помощью выполнения следующей команды:

```
% sysctl dev.cpu.0 ①
```

① В этом случае цифра 0 обозначает первое ядро процессора.

Вывод должен быть похож на следующий:

```
dev.cpu.0.cx_method: C1/mwait/hwc C2/mwait/hwc C3/mwait/hwc/bma
dev.cpu.0.cx_usage_counters: 3507294 0 0
dev.cpu.0.cx_usage: 100.00% 0.00% 0.00% last 3804us
dev.cpu.0.cx_lowest: C3 ①
dev.cpu.0.cx_supported: C1/1/1 C2/2/1 C3/3/57 ②
dev.cpu.0.freq_levels: 2267/35000 2266/35000 1600/15000 800/12000 ③
dev.cpu.0.freq: 1600 ④
dev.cpu.0.temperature: 40.0C ⑤
dev.cpu.0.coretemp.throttle_log: 0
dev.cpu.0.coretemp.tjmax: 105.0C
dev.cpu.0.coretemp.resolution: 1
dev.cpu.0.coretemp.delta: 65
dev.cpu.0.%parent: acpi0
dev.cpu.0.%pnpinfo: _HID=none _UID=0 _CID=none
dev.cpu.0.%location: handle=\_PR_.CPU0
dev.cpu.0.%driver: cpu
```

```
dev.cpu.0.%desc: ACPI CPU
```

① Наименьшее состояние Cx, используемое для бездействия процессора.

② Поддерживаемые процессором состояния Cx.

③ Доступные в настоящее время уровни для CPU (частота/потребление энергии).

④ Текущая активная частота CPU в МГц.

⑤ Текущая температура процессора.

Если информация о температуре не отображается, загрузите модуль coretemp(4). В случае использования процессора AMD загрузите модуль amdtemp(4).

Как только информация о CPU станет доступной, самый простой способ настроить энергосбережение — позволить powerd(8) взять управление на себя.

Включите службу powerd(8) в /etc/rc.conf для запуска при загрузке системы:

```
# sysrc powerd_enable=YES
```

Также необходимо указать определённые параметры для powerd(8), чтобы сообщить ему, как управлять состоянием CPU, выполнив следующую команду:

```
# sysrc powerd_flags="-a hiadaptive -i 25 -r 85 -N"
```

1. `-a`: Выбирает режим, используемый при работе от сети переменного тока.
2. `hiadaptive`: Режим работы. Подробнее см. в powerd(8).
3. `-i`: Указывает уровень загрузки процессора в процентах, при котором адаптивный режим должен начать снижать производительность для экономии энергии.
4. `-r`: Указывает уровень загрузки процессора в процентах, при котором адаптивный режим считает, что процессор работает, и повышает производительность.
5. `-N`: Рассматривать время выполнения "nice"-процессов как время простоя при расчете нагрузки; т.е. не увеличивать частоту CPU, если он занят только "nice"-процессами.

И затем включите службу, выполнив следующую команду:

```
# service powerd start
```

14.6.3. Управление частотой процессора

FreeBSD включает универсальный драйвер cpufreq(4), который позволяет администратору или программам, таким как powerd(8) и sysutils/powerdxx, управлять частотой CPU для достижения желаемого баланса между производительностью и энергопотреблением. Более

низкая настройка сэкономит энергию, уменьшая нагрев CPU. Более высокая настройка увеличит производительность за счёт большего энергопотребления и усиленного нагрева.

14.6.4. Intel® Enhanced SpeedStep™

Драйвер Intel® Enhanced Speed Step™, est(4), заменяет универсальный драйвер cpufreq(4) для процессоров, поддерживающих эту функцию. Частота процессора может быть статически настроена с помощью sysctl(8) или скрипта запуска `/etc/rc.d/power_profile`. Дополнительное программное обеспечение, такое как powerd(8) или sysutils/powerdxx, может использоваться для автоматической регулировки частоты процессора на основе его загрузки.

Все поддерживаемые частоты, а также ожидаемое энергопотребление, можно узнать, изучив дерева sysctl(3):

```
# sysctl dev.cpufreq.0.freq_driver dev.cpu.0.freq_levels dev.cpu.0.freq
```

Вывод должен быть похож на следующий:

```
dev.cpufreq.0.freq_driver: est0
dev.cpu.0.freq_levels: 3001/53000 3000/53000 2900/50301 2700/46082 2600/43525
2400/39557 2300/37137 2100/33398 2000/31112 1800/27610 1700/25455 1500/22171
1400/20144 1200/17084 1100/15181 900/12329 800/10550
dev.cpu.0.freq: 800
```

Частота на 1 МГц выше максимальной частоты процессора указывает на наличие технологии Intel® Turbo Boost™.

14.6.5. Intel Speed Shift™

Пользователи, работающие на новых процессорах Intel®, могут заметить некоторые различия в динамическом управлении частотой при обновлении до FreeBSD 13. Новый драйвер для технологии Intel® Speed Shift™, доступной в определённых моделях, предоставляет возможность аппаратного изменения частоты ядер, в том числе для каждого ядра отдельно. FreeBSD 13 включает драйвер hwpstate_intel(4) для автоматического включения управления Speed Shift™ на поддерживаемых процессорах, заменяя старый драйвер Enhanced Speed Step™ est(4). Значение `dev.cpufreq.%d.freq_driver` в sysctl(8) покажет, использует ли система Speed Shift.

Чтобы определить, какой драйвер управления частотой используется, изучите OID `dev.cpufreq.0.freq_driver`.

```
# sysctl dev.cpufreq.0.freq_driver
```

Вывод должен быть похож на следующий:

```
dev.cpufreq.0.freq_driver: hwpstate_intel0
```

Это указывает на использование нового драйвера hwpstate_intel(4). В таких системах OID `dev.cpu.%d.freq_levels` будет показывать только максимальную частоту CPU и указывать уровень энергопотребления `-1`.

Текущая частота процессора может быть определена путем проверки OID `dev.cpu.%d.freq`.

```
# sysctl dev.cpu.0.freq_levels dev.cpu.0.freq
```

Вывод должен быть похож на следующий:

```
dev.cpu.0.freq_levels: 3696/-1
dev.cpu.0.freq: 898
```

Для получения дополнительной информации, включая сведения о балансировке производительности и энергопотребления, а также о том, как отключить этот драйвер, обратитесь к справочной странице hwpstate_intel(4).

Пользователи, привыкшие к использованию powerd(8) или sysutils/powerdxx, обнаружат, что эти утилиты были заменены драйвером hwpstate_intel(4) и больше не работают как ожидалось.

14.6.6. Управление энергопотреблением графической карты

Графические карты стали неотъемлемой частью современных компьютеров. Некоторые графические карты могут потреблять чрезмерное количество энергии. FreeBSD позволяет настраивать определённые параметры для снижения энергопотребления.

В случае использования видеокарты Intel® с драйвером graphics/drm-kmod следующие параметры можно добавить в /boot/loader.conf:

```
compat.linuxkpi.fastboot=1 ①
compat.linuxkpi.enable_dc=2 ②
compat.linuxkpi.enable_fbc=1 ③
```

① Попытайтесь пропустить ненужные установки режимов во время загрузки.

② Включить энергосберегающие состояния C для дисплея.

③ Включить сжатие кадрового буфера для экономии энергии

14.6.7. Приостановка/возобновление

Функция приостановки/возобновления позволяет перевести машину в состояние с низким энергопотреблением и возобновить работу системы без потери состояния запущенных

программ.

Для корректной работы функции приостановки/возобновления в системе должны быть загружены графические драйверы. В видеокартах без поддержки KMS следует использовать sc(4), чтобы не нарушить функциональность приостановки/возобновления.

Дополнительная информация о том, какой драйвер использовать и как его настроить, доступна в главе Система X Window.

acpi(4) поддерживает следующий список состояний сна:

Таблица 27. Поддерживаемые состояния сна

S1	**Быстрый переход в режим ожидания с сохранением в оперативной памяти. ЦП переходит в состояние пониженного энергопотребления, но большинство периферийных устройств остаются активными.**
S2	Состояние с меньшим энергопотреблением, чем S1, но с теми же основными характеристиками. Не поддерживается многими системами.
S3 (Режим сна)	Режим приостановки с сохранением в ОЗУ. Большинство устройств отключается, и система перестает работать, за исключением обновления памяти.
S4 (Режим гибернации)	Приостановка с сохранением состояния на диск. Все устройства отключаются, и система прекращает работу. При возобновлении система запускается, как после холодного включения. **Пока не поддерживается в FreeBSD**.
S5	Система корректно завершает работу и выключается.

Настройка приостановки/возобновления

Первым шагом будет определение типов состояний сна, которые поддерживает используемое оборудование, выполнив следующую команду:

```
% sysctl hw.acpi.supported_sleep_state
```

Вывод должен быть похож на следующий:

```
hw.acpi.supported_sleep_state: S3 S4 S5
```

Как упоминалось выше, FreeBSD **пока** не поддерживает состояние S4.

acpiconf(8) можно использовать для проверки корректности работы состояния S3, выполнив следующую команду. Если команда выполнится успешно, экран погаснет и компьютер выключится:

```
# acpiconf -s 3
```

В подавляющем большинстве случаев функциональность Suspend/Resume предназначена для использования на ноутбуке.

FreeBSD можно настроить для перехода в состояние S3 при закрытии крышки, добавив следующую строку в файл /etc/sysctl.conf.

```
hw.acpi.lid_switch_state=S3
```

Устранение неполадок при приостановке/возобновлении работы

Много усилий было приложено, чтобы функции приостановки (Suspend) и возобновления (Resume) работали корректно и наилучшим образом в FreeBSD. Однако в настоящее время эти функции работают правильно только на некоторых определённых моделях ноутбуков.

Некоторые проверки можно выполнить, если система работает некорректно.

В некоторых случаях достаточно выключить bluetooth. В других — загрузить правильный драйвер для видеокарты и т.д.

В случае, если это работает некорректно, некоторые рекомендации можно найти на FreeBSD Wiki в разделе Приостановка/возобновление.

14.7. Добавление swap-пространства

Иногда системе FreeBSD требуется больше места в подкачке. В этом разделе описаны два способа увеличения пространства подкачки: добавление раздела подкачки к существующему разделу или новому жёсткому диску и создание файла подкачки в существующей файловой системе.

Для получения информации о том, как зашифровать пространство подкачки, какие существуют варианты и почему это следует делать, обратитесь к разделу Шифрование раздела подкачки.

14.7.1. Раздел подкачки на новом жёстком диске или существующем разделе

Добавление нового диска для раздела подкачки обеспечивает лучшую производительность по сравнению с использованием раздела на существующем диске. Настройка разделов и дисков описана в Добавление дисков, а Проектирование разметки разделов рассматривает варианты разметки разделов и вопросы выбора размера раздела подкачки.

Можно использовать любой раздел, который в данный момент не смонтирован, даже если он уже содержит данные. Использование `swapon` на разделе с данными приведёт к их перезаписи и уничтожению. Перед выполнением `swapon` убедитесь, что выбранный раздел действительно предназначен для добавления в swap.

swapon(8) может использоваться для добавления раздела подкачки в систему выполнением следующей команды:

```
# swapon /dev/ada1p2
```

Для автоматического добавления этого раздела подкачки при загрузке добавьте запись в /etc/fstab:

```
/dev/ada1p2 none swap sw 0 0
```

См. fstab(5) для объяснения записей в /etc/fstab.

14.7.2. Создание файла подкачки

Эти примеры создают файл подкачки размером 512 МБ с именем /usr/swap0.

Файлы подкачки на файловых системах ZFS крайне не рекомендуются, так как подкачка может привести к зависанию системы.

Первым шагом является создание файла подкачки:

```
# dd if=/dev/zero of=/usr/swap0 bs=1m count=512
```

Второй шаг — установить соответствующие разрешения для нового файла:

```
# chmod 0600 /usr/swap0
```

Третий шаг — сообщить системе о файле подкачки, добавив строку в /etc/fstab:

```
md none swap sw,file=/usr/swap0,late 0 0
```

Раздел подкачки будет добавлен при запуске системы. Чтобы добавить раздел подкачки немедленно, используйте swapon(8):

```
# swapon -aL
```

Глава 15. Процесс загрузки FreeBSD

15.1. Обзор

Процесс запуска компьютера и загрузки операционной системы называется "процессом начальной загрузки" или "загрузкой". Процесс загрузки FreeBSD предоставляет большую гибкость в настройке действий при старте системы, включая возможность выбора между различными операционными системами, установленными на одном компьютере, разными версиями одной операционной системы или другим установленным ядром.

Эта глава подробно описывает доступные параметры конфигурации. В ней показано, как настроить процесс загрузки FreeBSD, включая все этапы до момента, когда ядро FreeBSD запустится, определит устройства и запустит init(8). Это происходит, когда цвет текста загрузочных сообщений меняется с ярко-белого на серый.

Прочтите эту главу, чтобы узнать:

- Компоненты загрузочной системы FreeBSD и их взаимодействие.
- Параметры, которые можно передать компонентам в процессе загрузки FreeBSD для управления процессом загрузки.
- Основы настройки подсказок устройств.
- Как загрузиться в однопользовательском и многопользовательском режимах и как правильно завершить работу системы FreeBSD.

Эта глава описывает только процесс загрузки FreeBSD на системах с архитектурой x86 и amd64.

15.2. Процесс загрузки FreeBSD

Включение компьютера и запуск операционной системы представляет собой интересную дилемму. По определению, компьютер не умеет ничего делать до запуска операционной системы. Это включает в себя запуск программ с диска. Если компьютер не может запустить программу с диска без операционной системы, а программы операционной системы находятся на диске, то как же тогда запускается операционная система?

Эта проблема аналогична описанной в книге «Приключения барона Мюнхгаузена». Персонаж частично провалился в люк, но смог выбраться, ухватившись за шнурки своих ботинок и подтянув себя вверх. В первые дни развития вычислительной техники термин ***bootstrap*** применялся к механизму загрузки операционной системы. Со временем он сократился до «**загрузки**» (англ. "booting").

На аппаратном обеспечении x86 загрузка операционной системы выполняется базовой системой ввода/вывода (BIOS). BIOS ищет на жёстком диске главную загрузочную запись (MBR), которая должна находиться в определённом месте диска. BIOS обладает достаточными знаниями для загрузки и запуска MBR и предполагает, что MBR затем сможет выполнить остальные задачи по загрузке операционной системы, возможно, с помощью

BIOS.

FreeBSD поддерживает загрузку как по старому стандарту MBR, так и по новому стандарту GUID Partition Table (GPT). Разметка GPT часто встречается на компьютерах с Unified Extensible Firmware Interface (UEFI). Однако FreeBSD может загружаться с разделов GPT даже на машинах с устаревшим BIOS, используя gptboot(8). Ведутся работы по обеспечению прямой загрузки через UEFI.

Код в MBR обычно называют *загрузчиком*, особенно если он взаимодействует с пользователем. Загрузчик обычно содержит больше кода в первом треке диска или в файловой системе. Примерами загрузчиков являются стандартный загрузчик FreeBSD boot0, также называемый Boot Easy, и GNU GRUB, который используется во многих дистрибутивах Linux®.

Пользователи GRUB могут обратиться к документации от GNU.

Если на компьютере установлена только одна операционная система, MBR ищет первый загрузочный (активный) раздел на диске, а затем запускает код в этом разделе для загрузки остальной части операционной системы. При наличии нескольких операционных систем можно установить другой загрузчик, который отобразит список операционных систем, позволяя пользователю выбрать нужную для загрузки.

Оставшаяся часть загрузочной системы FreeBSD разделена на три этапа. Первый этап знает ровно столько, чтобы перевести компьютер в определённое состояние и запустить второй этап. Второй этап может выполнить немного больше, прежде чем запустить третий этап. Третий этап завершает загрузку операционной системы. Работа разделена на три этапа, потому что MBR накладывает ограничения на размер программ, которые могут быть выполнены на первом и втором этапах. Объединение задач в цепочку позволяет FreeBSD предоставить более гибкий загрузчик.

Затем запускается ядро, которое начинает поиск и инициализацию устройств для их использования. После завершения процесса загрузки ядра управление передаётся пользовательскому процессу init(8), который проверяет готовность дисков к работе, запускает пользовательскую настройку ресурсов, монтирует файловые системы, настраивает сетевые карты для работы в сети и запускает процессы, запланированные для выполнения при старте системы.

Этот раздел подробно описывает эти этапы и показывает, как взаимодействовать с процессом загрузки FreeBSD.

15.2.1. Загрузчик

Код загрузчика в MBR иногда называют *нулевой стадией* процесса загрузки. По умолчанию FreeBSD использует загрузчик boot0.

MBR, записываемый установщиком FreeBSD, основан на /boot/boot0. Размер и возможности boot0 ограничены 446 байтами из-за таблицы разделов и идентификатора `0x55AA` в конце MBR. Если установлены boot0 и несколько операционных систем, при загрузке будет

отображаться сообщение, похожее на этот пример:

Пример 23. Экран boot0

```
F1 Win
F2 FreeBSD

Default: F2
```

Другие операционные системы перезапишут существующую MBR, если они установлены после FreeBSD. Если это произошло или требуется заменить существующую MBR на MBR FreeBSD, используйте следующую команду:

```
# fdisk -B -b /boot/boot0 device
```

где *device* — это загрузочный диск, например, ad0 для первого IDE-диска, ad2 для первого IDE-диска на втором IDE-контроллере или da0 для первого SCSI-диска. Для создания пользовательской конфигурации MBR обратитесь к boot0cfg(8).

15.2.2. Этап первый и Этап второй

Концептуально первая и вторая стадии являются частью одной программы и расположены в одной области диска. Из-за ограничений по объёму они были разделены на две части, но всегда устанавливаются вместе. Они копируются из объединённого файла /boot/boot с помощью установщика FreeBSD или `bsdlabel`.

Эти два этапа расположены за пределами файловых систем, в первом треке загрузочного раздела, начиная с первого сектора. Именно здесь boot0 или любой другой загрузчик ожидает найти программу для запуска, которая продолжит процесс загрузки.

Первая стадия, boot1, очень проста, так как её размер может составлять только 512 байт. Она знает ровно столько о *bsdlabel* FreeBSD, в котором хранится информация о слайсе, чтобы найти и выполнить boot2.

Этап два, boot2, немного сложнее и понимает файловую систему FreeBSD достаточно хорошо, чтобы находить файлы. Он может предоставить простой интерфейс для выбора ядра или загрузчика для запуска. Он запускает loader, который гораздо более продвинут и предоставляет файл конфигурации загрузки. Если процесс загрузки прерывается на втором этапе, отображается следующий интерактивный экран:

Пример 24. Экран boot2

```
>> FreeBSD/i386 BOOT
Default: 0:ad(0,a)/boot/loader
boot:
```

Для замены установленных boot1 и boot2 используйте `bsdlabel`, где *diskslice* — это диск и раздел, с которого производится загрузка, например ad0s1 для первого раздела на первом IDE-диске:

```
# bsdlabel -B diskslice
```

Если указано только имя диска, например ad0, `bsdlabel` создаст диск в «опасном выделенном режиме» (dangerously dedicated mode) без разделов. Вероятно, это нежелательное действие, поэтому дважды проверьте *дисковый раздел* перед нажатием `Enter`.

15.2.3. Этап три

Загрузчик является завершающим этапом трёхэтапного процесса начальной загрузки. Он располагается в файловой системе, обычно как /boot/loader.

Загрузчик предназначен для интерактивной настройки с использованием встроенного набора команд, а также поддерживается более мощным интерпретатором с расширенным набором команд.

Во время инициализации загрузчик определит консоль и диски, а также выяснит, с какого диска происходит загрузка. Он установит соответствующие переменные и запустит интерпретатор, в котором пользовательские команды могут передаваться из скрипта или вводиться в интерактивном режиме.

Затем загрузчик прочитает /boot/loader.rc, который по умолчанию загружает /boot/defaults/loader.conf, где задаются разумные значения по умолчанию для переменных, а также читает /boot/loader.conf для локальных изменений этих переменных. После этого loader.rc действует в соответствии с этими переменными, загружая выбранные модули и ядро.

Наконец, по умолчанию загрузчик ожидает нажатия клавиш в течение 10 секунд и загружает ядро, если не было прерывания. Если прерывание произошло, пользователю предоставляется командная строка, поддерживающая набор команд, где можно изменить переменные, выгрузить все модули, загрузить модули и, наконец, загрузить или перезагрузить систему. Встроенные команды загрузчика перечисляет наиболее часто используемые команды загрузчика. Для полного описания всех доступных команд обратитесь к loader(8).

Таблица 28. Встроенные команды загрузчика

Переменная	Описание
autoboot *секунды*	Продолжает загрузку ядра, если не прервано в течение указанного времени в секундах. Отображает обратный отсчет, а время по умолчанию составляет 10 секунд.

Переменная	Описание
boot [-options] [kernelname]	Немедленно переходит к загрузке ядра с любыми указанными параметрами или именем ядра. Указание имени ядра в командной строке применимо только после выполнения команды unload. В противном случае будет использовано ранее загруженное ядро. Если *kernelname* не является полным путём к файлу ядра, поиск будет выполнен в */boot/kernel* и */boot/modules*.
boot-conf	Проходит ту же автоматическую настройку модулей на основе указанных переменных, чаще всего kernel. Имеет смысл только при использовании unload перед изменением некоторых переменных.
help [тема]	Показывает справочные сообщения, прочитанные из файла /boot/loader.help. Если указана тема index, отображается список доступных тем.
include имяфайла ...	Читает указанный файл и обрабатывает его построчно. Ошибка немедленно останавливает include.
load [-t тип] имяфайла	Загружает ядро, модуль ядра или файл указанного типа с заданным именем. Все аргументы после *имяфайла* передаются файлу. Если *имяфайла* не указано полностью, файл будет искаться в */boot/kernel* и */boot/modules*.
ls [-l] [путь]	Отображает список файлов в указанном пути или в корневом каталоге, если путь не указан. Если указан параметр -l, также будут показаны размеры файлов.
lsdev [-v]	Перечисляет все устройства, с которых возможно загрузить модули. Если указан параметр -v, выводится более подробная информация.
lsmod [-v]	Отображает загруженные модули. Если указан параметр -v, выводится более подробная информация.
more имяфайла	Отображает указанные файлы с паузой после каждых LINES выведенных строк.
reboot	Немедленно перезагружает систему.
set variable, set variable=value	Устанавливает указанные переменные окружения.
unload	Удаляет все загруженные модули.

Вот несколько практических примеров использования загрузчика. Чтобы загрузить обычное ядро в однопользовательском режиме:

```
boot -s
```

Чтобы выгрузить обычное ядро и модули, а затем загрузить предыдущее или другое указанное ядро:

```
unload
```

```
load /путь/к/файлуядра
```

Используйте полный путь /boot/GENERIC/kernel для ссылки на стандартное ядро, поставляемое с установкой, или /boot/kernel.old/kernel для ссылки на предыдущую версию ядра до обновления системы или до настройки собственного ядра.

Используйте следующее для загрузки обычных модулей с другим ядром. Обратите внимание, что в этом случае не обязательно указывать полный путь к файлу, достаточно только имя:

```
unload
set kernel="mykernel"
boot-conf
```

Для загрузки автоматизированного сценария конфигурации ядра:

```
load -t userconfig_script /boot/kernel.conf
```

15.2.4. Последний этап

После загрузки ядра с помощью `loader` или `boot2`, который обходит `loader`, оно проверяет флаги загрузки и при необходимости корректирует свое поведение. В Взаимодействие ядра во время загрузки приведены часто используемые флаги загрузки. Дополнительную информацию о других флагах загрузки можно найти в boot(8).

Таблица 29. Взаимодействие с ядром во время загрузки

Опция	Описание
`-a`	Во время инициализации ядра запросить устройство для монтирования в качестве корневой файловой системы.
`-C`	Загрузка корневой файловой системы с CDROM.
`-s`	Загрузится в однопользовательский режим.
`-v`	Более подробный вывод во время загрузки ядра.

После завершения загрузки ядро передаёт управление пользовательскому процессу init(8), который находится в /sbin/init, или программе, указанной в переменной `init_path` в `loader`. Это последний этап процесса загрузки.

Последовательность загрузки гарантирует, что файловые системы, доступные в системе, находятся в согласованном состоянии. Если файловая система UFS не согласована и `fsck` не может исправить несоответствия, init переводит систему в однопользовательский режим, чтобы администратор системы мог устранить проблему вручную. В противном случае

система загружается в многопользовательский режим.

Однопользовательский режим

Пользователь может указать этот режим, загрузившись с ключом `-s` или установив переменную `boot_single` в загрузчике. Также можно перейти в этот режим, выполнив команду `shutdown now` из многопользовательского режима. Однопользовательский режим начинается со следующего сообщения:

```
Enter full pathname of shell or RETURN for /bin/sh:
```

Если пользователь нажмёт `Enter`, система запустит оболочку POSIX® по умолчанию. Чтобы указать другую оболочку, введите полный путь к ней.

Однопользовательский режим обычно используется для восстановления системы, которая не загружается из-за несогласованности файловой системы или ошибки в конфигурационном файле загрузки. Он также может применяться для сброса пароля `root`, если он неизвестен. Эти действия возможны, поскольку приглашение однопользовательского режима предоставляет полный локальный доступ к системе и её конфигурационным файлам. В этом режиме отсутствует сетевое взаимодействие.

Хотя однопользовательский режим полезен для восстановления системы, он представляет угрозу безопасности, если система не находится в физически защищённом месте. По умолчанию любой пользователь, имеющий физический доступ к системе, получит полный контроль над ней после загрузки в однопользовательском режиме.

Если в /etc/ttys параметр `console` изменён на `insecure`, система сначала запросит пароль `root` перед переходом в однопользовательский режим. Это добавляет уровень безопасности, но лишает возможности сбросить пароль `root`, если он неизвестен.

Пример 25. Настройка небезопасной консоли в /etc/ttys

```
# name  getty                           type    status          comments
#
# If console is marked "insecure", then init will ask for the root password
# when going to single-user mode.
console none                            unknown off insecure
```

`insecure` консоль означает, что физическая безопасность консоли считается ненадёжной, поэтому только тот, кто знает пароль `root`, может использовать однопользовательский режим.

Многопользовательский режим

Если `init` обнаруживает, что файловые системы в порядке, или после того, как пользователь завершит свои команды в однопользовательском режиме и введёт `exit` для выхода из этого режима, система переходит в многопользовательский режим, в котором начинается

настройка ресурсов системы.

Система конфигурации ресурсов загружает настройки по умолчанию из /etc/defaults/rc.conf и специфичные для системы параметры из /etc/rc.conf. Затем она монтирует файловые системы, перечисленные в /etc/fstab. После этого запускаются сетевые службы, различные системные демоны, а затем скрипты запуска локально установленных пакетов.

Чтобы узнать больше о системе настройки ресурсов, обратитесь к rc(8) и изучите скрипты в /etc/rc.d.

15.3. Подсказки устройств

Во время начальной загрузки системы загрузчик loader(8) читает файл device.hints(5). Этот файл хранит информацию о загрузке ядра, известную как переменные, иногда называемые "подсказками устройств" ("device hints"). Эти "подсказки устройств" используются драйверами устройств для их конфигурации.

Подсказки для устройств также могут быть указаны в строке загрузчика на Этапе 3, как показано в Третий Этап. Переменные можно добавить с помощью `set`, удалить с помощью `unset` и просмотреть с помощью `show`. Переменные, заданные в /boot/device.hints, также могут быть переопределены. Подсказки для устройств, введённые в загрузчике, не являются постоянными и не будут применены при следующей перезагрузке.

После загрузки системы все переменные можно вывести с помощью kenv(1).

Синтаксис для /boot/device.hints — одна переменная на строку, символ "#" используется для комментариев. Строки формируются следующим образом:

```
hint.driver.unit.keyword="value"
```

Синтаксис загрузчика Этапа 3 следующий:

```
set hint.driver.unit.keyword=value
```

где `driver` — это имя драйвера устройства, `unit` — номер устройства драйвера, а `keyword` — ключевое слово подсказки. Ключевое слово может включать следующие варианты:

- `at`: указывает шину, к которой подключено устройство.
- `port`: указывает начальный адрес ввода-вывода, который будет использоваться.
- `irq`: указывает номер запроса на прерывание, который должен использоваться.
- `drq`: указывает номер канала DMA.
- `maddr`: указывает физический адрес памяти, занятый устройством.
- `flags`: устанавливает различные биты флагов для устройства.
- `disabled`: если установлено в `1`, устройство отключено.

Поскольку драйверы устройств могут принимать или требовать дополнительные подсказки, не перечисленные здесь, рекомендуется ознакомиться с руководством по конкретному драйверу. Для получения дополнительной информации обратитесь к device.hints(5), kenv(1), loader.conf(5) и loader(8).

15.4. Последовательность выключения

При контролируемом завершении работы с помощью shutdown(8), init(8) попытается выполнить скрипт /etc/rc.shutdown, а затем отправит всем процессам сигнал `TERM`, и после этого сигнал `KILL` всем процессам, которые не завершатся своевременно.

Для выключения питания машины FreeBSD на архитектурах и системах, поддерживающих управление питанием, используйте `shutdown -p now`, чтобы немедленно отключить питание. Для перезагрузки системы FreeBSD используйте `shutdown -r now`. Для выполнения shutdown(8) необходимо быть `root` или членом группы `operator`. Также можно использовать halt(8) и reboot(8). Дополнительную информацию см. на их справочных страницах и в shutdown(8).

Измените членство в группе, обратившись к разделу «Пользователи и основы управления учётными записями».

Управление питанием требует, чтобы модуль acpi(4) был загружен как модуль или статически скомпилирован в собственное ядро.

Глава 16. Безопасность

16.1. Обзор

Сотни стандартных практик написаны о том, как защитить системы и сети, и, как пользователю FreeBSD, понимание того, как защититься от атак и злоумышленников, является обязательным.

В этой главе будут рассмотрены несколько основополагающих принципов и методов. Система FreeBSD включает в себя несколько уровней безопасности, а также множество сторонних утилит, которые могут быть добавлены для её усиления.

Эта глава охватывает:

- Основные концепции безопасности системы FreeBSD.
- Доступные в FreeBSD механизмы шифрования.
- Как настроить TCP Wrappers для использования с inetd(8).
- Как настроить Kerberos на FreeBSD.
- Как настроить и использовать OpenSSH на FreeBSD.
- Как использовать OpenSSL в FreeBSD.
- Как использовать ACL файловых систем.
- Как использовать pkg для аудита сторонних программных пакетов, установленных из Коллекции портов.
- Как использовать рекомендации по безопасности FreeBSD.
- Что такое учёт процессов и как его включить в FreeBSD.
- Как управлять ресурсами пользователей с помощью классов входа или базы данных ограничений ресурсов.
- Что такое Capsicum и базовый пример.

Некоторые темы из-за их сложности рассматриваются в отдельных главах, таких как Межсетевые экраны, Принудительное управление доступом, и статьях, например VPN через IPsec.

16.2. Введение

Безопасность — это ответственность каждого. Слабое звено в любой системе может позволить злоумышленникам получить доступ к важной информации и нанести ущерб всей сети. Один из основных принципов информационной безопасности — триада КЦД, которая означает Конфиденциальность, Целостность и Доступность информационных систем.

Триада КЦД (конфиденциальность, целостность, доступность) — это фундаментальная концепция безопасности в компьютерных системах, так как клиенты и пользователи

ожидают, что их данные будут защищены. Например, клиент ожидает, что информация о его кредитной карте хранится безопасно (конфиденциальность), что его заказы не будут изменены без его ведома (целостность) и что он в любое время сможет получить доступ к информации о своих заказах (доступность).

Для обеспечения КЦД специалисты по безопасности применяют стратегию "глубокой эшелонированной защиты". Идея глубокой эшелонированной защиты заключается в добавлении нескольких уровней безопасности, чтобы предотвратить отказ одного уровня и полный крах всей системы безопасности. Например, системный администратор не может просто включить межсетевой экран и считать сеть или систему безопасными. Необходимо также проводить аудит учётных записей, проверять целостность бинарных файлов и убеждаться, что вредоносные инструменты не установлены. Для реализации эффективной стратегии безопасности необходимо понимать угрозы и способы защиты от них.

Что такое угроза в контексте компьютерной безопасности? Угрозы не ограничиваются удалёнными злоумышленниками, которые пытаются получить доступ к системе без разрешения из удалённого местоположения. Угрозы также включают сотрудников, вредоносное программное обеспечение, несанкционированные сетевые устройства, стихийные бедствия, уязвимости безопасности и даже конкурирующие компании.

Системы и сети могут быть доступны без разрешения, иногда случайно, или удалёнными злоумышленниками, а в некоторых случаях — посредством корпоративного шпионажа или действий бывших сотрудников. Как пользователю, важно быть готовым признать, когда ошибка привела к нарушению безопасности, и сообщить о возможных проблемах команде безопасности. Как администратору, важно знать об угрозах и быть готовым к их устранению.

Применяя меры безопасности к системам, рекомендуется начать с защиты базовых учётных записей и конфигурации системы, а затем обеспечить безопасность сетевого уровня, чтобы он соответствовал политике системы и процедурам безопасности организации. Многие организации уже имеют политику безопасности, которая охватывает конфигурацию технологических устройств. Эта политика должна включать настройки безопасности рабочих станций, настольных компьютеров, мобильных устройств, телефонов, производственных серверов и серверов разработки. Во многих случаях уже существуют стандартные операционные процедуры (СОП). Если возникают сомнения, обратитесь к команде безопасности.

16.3. Защита учётных записей

Поддержание безопасных учётных записей в FreeBSD крайне важно для конфиденциальности данных, целостности системы и разделения привилегий, так как это предотвращает несанкционированный доступ, вредоносное ПО и утечки данных, обеспечивая соответствие требованиям и защищая репутацию организации.

16.3.1. Предотвращение входов в систему

При обеспечении безопасности системы хорошей отправной точкой является аудит учётных записей. Отключите все учётные записи, которым не требуется доступ для входа.

Убедитесь, что у пользователя `root` установлен надёжный пароль и что этот пароль не используется совместно.

Для запрета входа в учётные записи существуют два метода.

Первый способ — заблокировать учётную запись, в этом примере показано, как заблокировать учётную запись `imani`:

```
# pw lock imani
```

Второй способ — запретить вход в систему, изменив оболочку на /usr/sbin/nologin. Оболочка nologin(8) предотвращает назначение оболочки пользователю при попытке входа в систему.

Только суперпользователь может изменить оболочку для других пользователей:

```
# chsh -s /usr/sbin/nologin imani
```

16.3.2. Хеши паролей

Пароли — это неизбежное зло в мире технологий. Когда их использование необходимо, они должны быть сложными, а для хранения зашифрованной версии в базе данных паролей следует использовать надёжный механизм хеширования. FreeBSD поддерживает несколько алгоритмов, включая SHA256, SHA512 и Blowfish, в своей библиотеке `crypt()`. Подробности можно найти в crypt(3).

По умолчанию используется SHA512, и его не следует заменять на менее безопасный алгоритм хеширования, но можно перейти на более безопасный алгоритм Blowfish.

Blowfish не является частью AES и не соответствует Федеральным стандартам обработки информации (FIPS). Его использование может быть запрещено в некоторых средах.

Для определения того, какой алгоритм хеширования используется для шифрования пароля пользователя, суперпользователь может просмотреть хеш пользователя в базе данных паролей FreeBSD. Каждый хеш начинается с символа, который указывает на тип механизма хеширования, использованного для шифрования пароля.

Если используется DES, начальный символ отсутствует. Для MD5 символ — `$`. Для SHA256 и SHA512 символ — `$6$`. Для Blowfish символ — `$2a$`. В этом примере пароль для `imani` хеширован с использованием алгоритма SHA512 по умолчанию, так как хеш начинается с `$6$`. Обратите внимание, что в базе данных паролей хранится зашифрованный хеш, а не сам пароль:

```
# grep imani /etc/master.passwd
```

Вывод должен быть похож на следующий:

```
imani:$6$pzIjSvCAn.PBYQBA$PXpSeWPx3g5kscj3IMiM7tUEUSPmGexxta.8Lt9TGSi2lNQqYGKszsBPuGME
0:1001:1001::0:0:imani:/usr/home/imani:/bin/sh
```

Механизм хеширования задается в классе входа пользователя.

Следующая команда позволяет проверить, какой механизм хеширования используется в данный момент:

```
% grep passwd_format /etc/login.conf
```

Вывод должен быть похож на следующий:

```
:passwd_format=sha512:\
```

Например, чтобы изменить алгоритм на Blowfish, измените эту строку следующим образом:

```
:passwd_format=blf:\
```

Затем необходимо выполнить cap_mkdb(1) для обновления базы данных login.conf:

```
# cap_mkdb /etc/login.conf
```

Обратите внимание, что это изменение не затронет существующие хэши паролей. Это означает, что все пароли должны быть перехэшированы, для чего пользователям необходимо запустить `passwd`, чтобы изменить свой пароль.

16.3.3. Политика применения паролей

Обеспечение строгой политики паролей для локальных учётных записей является основополагающим аспектом безопасности системы. В FreeBSD длина пароля, его стойкость и сложность могут быть реализованы с использованием встроенных подключаемых модулей аутентификации (Pluggable Authentication Modules - PAM).

Этот раздел демонстрирует, как настроить минимальную и максимальную длину пароля, а также принудительное использование смешанных символов с помощью модуля pam_passwdqc(8). Данный модуль применяется при изменении пользователем своего пароля.

Для настройки этого модуля станьте суперпользователем и раскомментируйте строку с `pam_passwdqc.so` в /etc/pam.d/passwd.

Затем отредактируйте эту строку в соответствии с политикой паролей:

```
password        requisite       pam_passwdqc.so
```

```
min=disabled,disabled,disabled,12,10 similar=deny retry=3 enforce=users
```

Описание параметров можно найти в pam_passwdqc(8).

После сохранения этого файла пользователь, изменяющий свой пароль, увидит сообщение, похожее на следующее:

```
% passwd
```

Вывод должен быть похож на следующий:

```
Changing local password for user
Old Password:

You can now choose the new password.
A valid password should be a mix of upper and lower case letters,
digits and other characters.  You can use a 12 character long
password with characters from at least 3 of these 4 classes, or
a 10 character long password containing characters from all the
classes.  Characters that form a common pattern are discarded by
the check.
Alternatively, if no one else can see your terminal now, you can
pick this as your password: "trait-useful&knob".
Enter new password:
```

Если вводится пароль, не соответствующий политике, он будет отклонён с предупреждением, и пользователю будет предоставлена возможность попробовать снова, но не более установленного количества попыток.

Если политика вашей организации требует, чтобы пароли имели срок действия, FreeBSD поддерживает параметр `passwordtime` в классе пользователя в /etc/login.conf

Класс входа `default` содержит пример:

```
#       :passwordtime=90d:\
```

Итак, чтобы установить срок действия в 90 дней для этого класса входа, удалите символ комментария (#), сохраните изменения и выполните следующую команду:

```
# cap_mkdb /etc/login.conf
```

Чтобы установить срок действия для отдельных пользователей, передайте дату истечения срока или количество дней до истечения и имя пользователя в `pw`:

```
# pw usermod -p 30-apr-2025 -n user
```

Как показано здесь, срок действия устанавливается в виде дня, месяца и года. Для получения дополнительной информации см. pw(8).

16.3.4. Совместное администрирование с помощью sudo

Системные администраторы часто сталкиваются с необходимостью предоставления пользователям расширенных прав для выполнения привилегированных задач. Идея о том, что члены команды получают доступ к системе FreeBSD для выполнения своих конкретных задач, создаёт уникальные проблемы для каждого администратора. Этим сотрудникам требуется лишь ограниченный набор прав, выходящий за рамки обычного уровня пользователя; однако они почти всегда заявляют руководству, что не могут выполнять свои задачи без прав суперпользователя. К счастью, нет необходимости предоставлять такие права конечным пользователям, так как существуют инструменты для управления этим требованием.

Даже администраторы должны ограничивать свои привилегии, когда в них нет необходимости.

До этого момента в главе о безопасности рассматривались вопросы предоставления доступа авторизованным пользователям и попытки предотвратить несанкционированный доступ. Однако возникает другая проблема, когда авторизованные пользователи получают доступ к системным ресурсам. Во многих случаях некоторым пользователям может потребоваться доступ к скриптам запуска приложений, или команде администраторов необходимо обслуживать систему.. Традиционно для управления таким доступом использовались стандартные пользователи и группы, права доступа к файлам и даже команда su(1). Однако по мере того, как приложениям требовалось больше прав, а большему числу пользователей нужно было использовать системные ресурсы, потребовалось лучшее решение. В настоящее время наиболее часто используемым приложением для этих целей является Sudo.

Sudo позволяет администраторам настраивать более строгий доступ к системным командам и обеспечивать дополнительные функции ведения журналов. Этот инструмент доступен в коллекции портов как security/sudo или с помощью утилиты pkg(8).

Выполните следующую команду для установки:

```
# pkg install sudo
```

После завершения установки установленный `visudo` откроет файл конфигурации в текстовом редакторе. Настоятельно рекомендуется использовать `visudo`, так как он содержит встроенную проверку синтаксиса для подтверждения отсутствия ошибок перед сохранением файла.

Файл конфигурации состоит из нескольких небольших разделов, которые позволяют

проводить детальную настройку. В следующем примере администратору веб-приложения, user1, требуется запускать, останавливать и перезапускать веб-приложение с именем *webservice*. Чтобы предоставить этому пользователю права на выполнение данных задач, добавьте следующую строку в конец файла /usr/local/etc/sudoers:

```
user1   ALL=(ALL)       /usr/sbin/service webservice *
```

Пользователь может теперь запустить *webservice* с помощью следующей команды:

```
% sudo /usr/sbin/service webservice start
```

Хотя данная конфигурация предоставляет одному пользователю доступ к службе `webservice`, в большинстве организаций управление этой службой доверено целой команде веб-разработчиков. Одной строкой можно также предоставить доступ всей группе. Следующие шаги позволят создать группу `web`, добавить в неё пользователя и разрешить всем членам группы управлять службой:

```
# pw groupadd -g 6001 -n webteam
```

Используя ту же команду pw(8), пользователь добавляется в группу webteam:

```
# pw groupmod -m user1 -n webteam
```

Наконец, эта строка в /usr/local/etc/sudoers разрешает любому члену группы webteam управлять *webservice*:

```
%webteam   ALL=(ALL)       /usr/sbin/service webservice *
```

В отличие от su(1), sudo(8) требует только пароль конечного пользователя. Это позволяет избежать совместного использования паролей, что является небезопасной практикой.

Пользователи, которым разрешено запускать приложения с помощью sudo(8), вводят только свои собственные пароли. Это более безопасно и обеспечивает лучший контроль, чем su(1), где вводится пароль `root` и пользователь получает все права `root`.

Большинство организаций переходят или уже перешли на модель двухфакторной аутентификации. В таких случаях у пользователя может не быть пароля для ввода.

sudo(8) можно настроить для поддержки двухфакторной модели аутентификации с использованием переменной `NOPASSWD`. Добавление её в приведённую выше конфигурацию позволит всем членам группы *webteam* управлять службой без требования пароля:

```
%webteam   ALL=(ALL)       NOPASSWD: /usr/sbin/service webservice *
```

16.3.5. Совместное администрирование с Doas

doas(1) - это утилита командной строки, портированная из OpenBSD. Она служит альтернативой широко используемой команде sudo(8) в Unix-подобных системах.

С помощью `doas` пользователи могут выполнять команды с повышенными привилегиями, обычно от имени пользователя `root`, сохраняя при этом простой и безопасный подход. В отличие от sudo(8), `doas` делает акцент на простоте и минимализме, предлагая лаконичное делегирование полномочий без избыточного количества настройки.

Выполните следующую команду для установки:

```
# pkg install doas
```

После установки необходимо настроить /usr/local/etc/doas.conf, чтобы предоставить пользователям доступ к определённым командам или ролям.

Самый простой вариант может выглядеть следующим образом, который предоставляет пользователю `local_user` права `root` без запроса пароля при выполнении команды doas.

```
permit nopass local_user as root
```

После установки и настройки утилиты `doas` команда теперь может быть выполнена с повышенными привилегиями, например:

```
$ doas vi /etc/rc.conf
```

Для дополнительных примеров конфигурации обратитесь к doas.conf(5).

16.4. Система обнаружения вторжений (IDS)

Проверка системных файлов и двоичных файлов важна, поскольку предоставляет администраторам системы и командам безопасности информацию об изменениях в системе. Программное приложение, которое отслеживает изменения в системе, называется системой обнаружения вторжений (Intrusion Detection System — IDS).

FreeBSD предоставляет встроенную поддержку базовой системы IDS под названием mtree(8). Хотя ежедневные письма с информацией о безопасности уведомят администратора об изменениях, эти данные хранятся локально, и существует вероятность, что злоумышленник может изменить их, чтобы скрыть свои изменения в системе. Поэтому рекомендуется создать отдельный набор бинарных сигнатур и хранить их в предназначенном только для чтения каталоге, принадлежащем root, или,

предпочтительно, на съёмном USB-диске или удалённом сервере.

Также рекомендуется запускать `freebsd-update IDS` после каждого обновления.

16.4.1. Генерация файла спецификации

Встроенная утилита mtree(8) может использоваться для создания спецификации содержимого каталога. Спецификация генерируется с использованием начального значения (seed) — числовой константы, которая также необходима для проверки неизменности спецификации. Это позволяет определить, был ли изменён файл или бинарный файл. Поскольку злоумышленник не знает начальное значение, подделать или проверить контрольные суммы файлов будет крайне сложно или невозможно.

Рекомендуется создавать спецификации для каталогов, содержащих исполняемые файлы и конфигурационные файлы, а также для любых каталогов с чувствительными данными. Обычно спецификации создаются для /bin, /sbin, /usr/bin, /usr/sbin, /usr/local/bin, /etc и /usr/local/etc.

Следующий пример создаёт набор хешей `sha512` — по одному для каждого системного бинарного файла в /bin — и сохраняет эти значения в скрытый файл в домашнем каталоге пользователя /home/user/.bin_chksum_mtree:

```
# mtree -s 123456789 -c -K cksum,sha512 -p /bin > /home/user/.bin_chksum_mtree
```

Вывод должен быть похож на следующий:

```
mtree: /bin checksum: 3427012225
```

Значение `123456789` представляет собой зерно (seed) и должно быть выбрано случайным образом. Этот параметр необходимо запомнить, **но не раскрывать**.

Важно скрывать начальное значение и контрольную сумму от злоумышленников.

16.4.2. Структура файла спецификации

Формат `mtree` — это текстовый формат, описывающий набор объектов файловой системы. Такие файлы обычно используются для создания или проверки иерархий каталогов.

Файл mtree состоит из серии строк, каждая из которых содержит информацию об отдельном объекте файловой системы. Начальные пробельные символы всегда игнорируются.

Созданный выше файл спецификации будет использоваться для объяснения формата и содержания:

```
#          user: root ①
#       machine: machinename ②
#          tree: /bin ③
#          date: Thu Aug  24 21:58:37 2023 ④

# .
/set type=file uid=0 gid=0 mode=0555 nlink=1 flags=uarch ⑤
.               type=dir mode=0755 nlink=2 time=1681388848.239523000 ⑥
    \133        nlink=2 size=12520 time=1685991378.688509000 \
                cksum=520880818 \

sha512=5c1374ce0e2ba1b3bc5a41b23f4bbdc1ec89ae82fa01237f376a5eeef41822e68f1d8f75ec46b7b
ceb65396c122a9d837d692740fdebdcc376a05275adbd3471
    cat         size=14600 time=1685991378.694601000 cksum=3672531848 \ ⑦

sha512=b30b96d155fdc4795432b523989a6581d71cdf69ba5f0ccb45d9b9e354b55a665899b16aee21982
fffe20c4680d11da4e3ed9611232a775c69f926e5385d53a2
    chflags     size=8920 time=1685991378.700385000 cksum=1629328991 \

sha512=289a088cbbcbeb436dd9c1f74521a89b66643976abda696b99b9cc1fbfe8b76107c5b54d4a6a9b6
5332386ada73fc1bbb10e43c4e3065fa2161e7be269eaf86a
    chio        size=20720 time=1685991378.706095000 cksum=1948751604 \

sha512=46f58277ff16c3495ea51e74129c73617f31351e250315c2b878a88708c2b8a7bb060e2dc8ff92f
606450dbc7dd2816da4853e465ec61ee411723e8bf52709ee
    chmod       size=9616 time=1685991378.712546000 cksum=4244658911 \

sha512=1769313ce08cba84ecdc2b9c07ef86d2b70a4206420dd71343867be7ab59659956f6f5a458c64e2
531a1c736277a8e419c633a31a8d3c7ccc43e99dd4d71d630
```

① Пользователь, создавший спецификацию.

② Имя хоста машины.

③ Путь к каталогу.

④ Дата и время создания спецификации.

⑤ `/set` специальные команды, определяют некоторые настройки, полученные из проанализированных файлов.

⑥ Ссылается на разобранный каталог и указывает такие параметры, как его тип, режим доступа, количество жёстких ссылок и время изменения в формате UNIX.

⑦ Ссылается на файл и показывает его размер, время и список хешей для проверки целостности.

16.4.3. Проверка файла спецификации

Для проверки неизменности бинарных сигнатур сравните текущее содержимое каталога с ранее созданной спецификацией и сохраните результаты в файл.

Эта команда требует начальное значение, которое использовалось для создания исходной

спецификации:

```
# mtree -s 123456789 -p /bin < /home/user/.bin_chksum_mtree >>
/home/user/.bin_chksum_output
```

Это должно дать такую же контрольную сумму для /bin, какая была получена при создании спецификации. Если в каталоге не было изменений бинарных файлов, выходной файл /home/user/.bin_chksum_output будет пустым.

Для имитации изменения измените дату файла /bin/cat с помощью touch(1) и снова выполните команду проверки:

```
# touch /bin/cat
```

Выполните команду проверки ещё раз:

```
# mtree -s 123456789 -p /bin < /home/user/.bin_chksum_mtree >>
/home/user/.bin_chksum_output
```

И затем проверьте содержимое выходного файла:

```
# cat /root/.bin_chksum_output
```

Вывод должен быть похож на следующий:

```
cat:    modification time (Fri Aug 25 13:30:17 2023, Fri Aug 25 13:34:20 2023)
```

Это просто пример того, что будет отображено при выполнении команды, чтобы показать изменения, которые произойдут в метаданных.

16.5. Уровни безопасности

securelevel — это механизм безопасности, реализованный в ядре. Когда securelevel имеет положительное значение, ядро ограничивает выполнение определённых задач; даже суперпользователь (root) не может их выполнять.

Механизм securelevel ограничивает возможность:

- Снять определённые флаги файлов, такие как `schg` (флаг системной неизменяемости).
- Записать в память ядра через /dev/mem и /dev/kmem.
- Загрузить модули ядра.
- Изменить правила межсетевого экрана.

16.5.1. Определения уровней безопасности

Ядро работает с пятью различными уровнями безопасности. Любой процесс с правами суперпользователя может повысить уровень, но ни один процесс не может его понизить.

Определения безопасности следующие:

-1

Постоянный небезопасный режим — всегда запускать систему в небезопасном режиме. Это значение по умолчанию при начальной настройке.

0

Небезопасный режим - флаги `immutable` и `append-only` могут быть отключены. Доступ на чтение и запись всех устройств разрешён в соответствии с их правами доступа.

1

Безопасный режим - флаги «только для чтения» (system immutable) и «только для дополнения» (system append-only) не могут быть отключены; диски с смонтированными файловыми системами, /dev/mem и /dev/kmem не могут быть открыты для записи; /dev/io (если он есть на вашей платформе) не может быть открыт вообще; загрузка и выгрузка модулей ядра (см. kld(4)) запрещены. Нельзя перейти в отладчик ядра с помощью sysctl debug.kdb.enter. Нельзя вызвать панику или прерывание через sysctl debug.kdb.panic, debug.kdb.panic_str и другие.

2

Высокозащищённый режим — аналогичен защищённому режиму, но дополнительно запрещает открывать диски на запись (за исключением mount(2)), независимо от того, смонтированы они или нет. Этот уровень предотвращает изменение файловых систем путём их размонтирования, но также запрещает запуск newfs(8) в многопользовательском режиме.

3

Режим сетевой безопасности - аналогично режиму высокой безопасности, плюс правила фильтрации IP-пакетов (см. ipfw(8), ipfirewall(4) и pfctl(8)) не могут быть изменены, а конфигурация dummynet(4) или pf(4) не может быть скорректирована.

Вкратце, ключевое различие между `Режимом постоянной незащищённости` и `Незащищённым режимом` в уровнях безопасности FreeBSD заключается в степени предоставляемой защиты. `Режим постоянной незащищённости` полностью снимает все ограничения безопасности, тогда как `Незащищённый режим` ослабляет некоторые ограничения, но сохраняет определённый уровень контроля и защиты.

16.5.2. Изменение уровней безопасности

Для изменения securelevel системы необходимо активировать `kern_securelevel_enable`, выполнив следующую команду:

```
# sysrc kern_securelevel_enable="YES"
```

И установите значение `kern_securelevel` на желаемый уровень безопасности:

```
# sysrc kern_securelevel=2
```

Для проверки уровня безопасности (securelevel) в работающей системе выполните следующую команду:

```
# sysctl -n kern.securelevel
```

Вывод содержит текущее значение уровня securelevel. Если оно больше 0, значит, включена по крайней мере часть защит securelevel.

16.6. Флаговые атрибуты файлов

Флаги файлов позволяют пользователям добавлять дополнительные метаданные или атрибуты к файлам и каталогам, помимо базовых прав доступа и владения. Эти флаги предоставляют способ управления различными поведениями и свойствами файлов без необходимости создавать специальные каталоги или использовать расширенные атрибуты.

Флаги файлов могут использоваться для достижения различных целей, таких как предотвращение удаления файла, разрешение только дополнения файла, синхронизация обновлений файлов и многое другое. Некоторые часто используемые флаги файлов в FreeBSD включают флаг "immutable", который запрещает изменение или удаление файла, и флаг "append-only", который разрешает только добавление данных в конец файла, но не их изменение или удаление.

Эти флаги можно управлять с помощью команды chflags(1) в FreeBSD, что предоставляет администраторам и пользователям больший контроль над поведением и характеристиками их файлов и каталогов. Важно отметить, что флаги файлов обычно управляются root или пользователями с соответствующими привилегиями, так как они могут влиять на доступ и манипуляции с файлами. Некоторые флаги доступны для использования владельцем файла, как описано в chflags(1).

16.6.1. Работа с флагами файлов

В этом примере файл с именем ~/important.txt в домашнем каталоге пользователя необходимо защитить от удаления.

Выполните следующую команду, чтобы установить флаг файла `schg`:

```
# chflags schg ~/important.txt
```

Когда любой пользователь, включая пользователя `root`, пытается удалить файл, система отобразит сообщение:

```
rm: important.txt: Operation not permitted
```

Чтобы удалить файл, необходимо сначала удалить его флаги, выполнив следующую команду:

```
# chflags noschg ~/important.txt
```

Список поддерживаемых флагов файлов и их функциональность можно найти в chflags(1).

16.7. OpenSSH

OpenSSH — это набор сетевых инструментов для подключения, которые используются для обеспечения безопасного доступа к удалённым машинам. Кроме того, TCP/IP-соединения могут быть туннелированы или перенаправлены через SSH-соединения с обеспечением безопасности. OpenSSH шифрует весь трафик, чтобы исключить прослушивание, перехват соединений и другие атаки на сетевом уровне.

OpenSSH разрабатывается проектом OpenBSD и устанавливается по умолчанию в FreeBSD.

Когда данные передаются по сети в незашифрованном виде, снифферы где-либо между клиентом и сервером могут украсть информацию о пользователе/пароле или данные, передаваемые во время сеанса. OpenSSH предлагает различные методы аутентификации и шифрования, чтобы предотвратить это.

Дополнительная информация о OpenSSH доступна на веб-сайте.

В этом разделе представлен обзор встроенных клиентских утилит для безопасного доступа к другим системам и безопасной передачи файлов с системы FreeBSD. Затем описывается, как настроить SSH-сервер на системе FreeBSD.

Как уже упоминалось, в этой главе будет рассмотрена базовая версия OpenSSH. Также доступна версия OpenSSH в пакете security/openssh-portable, которая предоставляет дополнительные параметры конфигурации и регулярно обновляется с новыми функциями.

16.7.1. Использование клиентских утилит SSH

Для входа на SSH-сервер используйте ssh(1), указав имя пользователя, существующее на этом сервере, и IP-адрес или имя хоста сервера. Если подключение к указанному серверу выполняется впервые, пользователю будет предложено сначала подтвердить его отпечаток:

```
# ssh user@example.com
```

Вывод должен быть похож на следующий:

```
The authenticity of host 'example.com (10.0.0.1)' can't be established.
ECDSA key fingerprint is 25:cc:73:b5:b3:96:75:3d:56:19:49:d2:5c:1f:91:3b.
Are you sure you want to continue connecting (yes/no)? yes
Permanently added 'example.com' (ECDSA) to the list of known hosts.
Password for user@example.com: user_password
```

SSH использует систему отпечатков ключей для проверки подлинности сервера при подключении клиента. Когда пользователь принимает отпечаток ключа, введя `yes` при первом подключении, копия ключа сохраняется в файле ~/.ssh/known_hosts в домашнем каталоге пользователя. Последующие попытки входа проверяются по сохранённому ключу, и ssh(1) выведет предупреждение, если ключ сервера не совпадает с сохранённым. В таком случае пользователю следует сначала проверить, почему изменился ключ, прежде чем продолжать подключение.

Как выполнить эту проверку, выходит за рамки данной главы.

Используйте scp(1) для безопасного копирования файла на удалённую машину или с неё.

Этот пример копирует файл `COPYRIGHT` на удалённой системе в файл с тем же именем в текущем каталоге локальной системы:

```
# scp user@example.com:/COPYRIGHT COPYRIGHT
```

Вывод должен быть похож на следующий:

```
Password for user@example.com: *******
COPYRIGHT            100% |*****************************|  4735
```

Поскольку отпечаток этого узла уже был проверен, ключ сервера автоматически проверяется перед запросом пароля пользователя.

Аргументы, передаваемые в scp(1), аналогичны аргументам cp(1). Первым аргументом указывается файл или файлы для копирования, а вторым — место назначения. Поскольку файл передаётся по сети, один или несколько аргументов с файлами имеют вид `user@host:<путь_к_удалённому_файлу>`. Обратите внимание, что при рекурсивном копировании каталогов scp(1) использует `-r`, тогда как cp(1) использует `-R`.

Чтобы открыть интерактивную сессию для копирования файлов, используйте sftp(1).

Обратитесь к sftp(1) для получения списка доступных команд во время сеанса sftp(1).

16.7.2. Аутентификация на основе ключей

Вместо использования паролей клиент может быть настроен для подключения к удалённой

машине с использованием ключей. В целях безопасности это предпочтительный метод.

ssh-keygen(1) можно использовать для генерации ключей аутентификации. Чтобы создать пару из открытого и закрытого ключей, укажите тип ключа и следуйте инструкциям. Рекомендуется защитить ключи запоминающейся, но трудной для угадывания парольной фразой.

```
% ssh-keygen -t rsa -b 4096
```

Вывод должен быть похож на следующий:

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa):
Created directory '/home/user/.ssh/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_rsa.
Your public key has been saved in /home/user/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:54Xm9Uvtv6H4NOo6yjP/YCfODryvUU7yWHzMqeXwhq8 user@host.example.com
The key's randomart image is:
+---[RSA 2048]----+
|                 |
|                 |
|                 |
|        . o..    |
|       .S*+*o    |
|      . O=Oo . . |
|       = Oo= oo..|
|      .oB.* +.oo.|
|       =OE**.o..=|
+----[SHA256]-----+
```

Закрытый ключ хранится в ~/.ssh/id_rsa, а открытый ключ — в ~/.ssh/id_rsa.pub. *Открытый* ключ должен быть скопирован в файл ~/.ssh/authorized_keys на удалённой машине, чтобы аутентификация по ключу работала.

Использование парольной фразы для ключей OpenSSH является важной практикой безопасности, обеспечивающей дополнительный уровень защиты от несанкционированного доступа и повышающей общую кибербезопасность.

В случае утери или кражи это добавляет дополнительный уровень защиты.

16.7.3. Туннелирование SSH

OpenSSH обладает возможностью создания туннеля для инкапсуляции другого протокола в зашифрованном сеансе.

Следующая команда указывает ssh(1) создать туннель:

```
% ssh -D 8080 user@example.com
```

Этот пример использует следующие параметры:

-D

Указывает локальное "динамическое" перенаправление портов на уровне приложений.

user@foo.example.com

Имя пользователя для входа на указанный удалённый SSH-сервер.

Туннель SSH работает путем создания сокета для прослушивания на `localhost` на указанном `localport`.

Этот метод можно использовать для защиты любого количества незащищённых TCP-протоколов, таких как SMTP, POP3 и FTP.

16.7.4. Включение сервера SSH

В дополнение к встроенным клиентским утилитам SSH, система FreeBSD может быть настроена как SSH-сервер, принимающий подключения от других SSH-клиентов.

Как уже было сказано, в этой главе рассматривается базовая версия OpenSSH. **Не** путайте её с security/openssh-portable, версией OpenSSH, которая поставляется с коллекцией портов FreeBSD.

Чтобы включить сервер SSH для автоматического запуска после перезагрузки, выполните следующую команду:

```
# sysrc sshd_enable="YES"
```

Затем выполните следующую команду, чтобы включить службу:

```
# service sshd start
```

При первом запуске `sshd` в системе FreeBSD автоматически создаются ключи хоста системы, и их отпечаток выводится на консоль. Предоставьте пользователям этот отпечаток, чтобы они могли проверить его при первом подключении к серверу.

Обратитесь к sshd(8) для получения списка доступных параметров при запуске sshd, а также полного описания аутентификации, процесса входа и различных конфигурационных файлов.

На этом этапе `sshd` должен быть доступен всем пользователям системы, имеющим имя пользователя и пароль.

16.7.5. Настройка метода аутентификации по открытому ключу

Настройка OpenSSH для использования аутентификации по открытому ключу повышает безопасность за счёт применения асимметричной криптографии. Этот метод устраняет риски, связанные с паролями, такие как слабые пароли или их перехват при передаче, а также предотвращает различные атаки, основанные на паролях. Однако важно обеспечить надёжную защиту закрытых ключей, чтобы предотвратить несанкционированный доступ.

Первым шагом будет настройка sshd(8) для использования требуемого метода аутентификации.

Отредактируйте файл /etc/ssh/sshd_config и раскомментируйте следующую настройку:

```
PubkeyAuthentication yes
```

После завершения настройки пользователям необходимо отправить системному администратору свой **открытый ключ**, и эти ключи будут добавлены в .ssh/authorized_keys. Процесс генерации ключей описан в Аутентификация на основе ключей.

Затем перезапустите сервер, выполнив следующую команду:

```
# service sshd reload
```

Настоятельно рекомендуется выполнить указанные улучшения безопасности, приведённые в Параметры безопасности SSH-сервера.

16.7.6. Параметры безопасности сервера SSH

В то время как `sshd` является наиболее широко используемым средством удалённого администрирования в FreeBSD, атаки методом грубой силы и сканирование уязвимостей распространены для любой системы, доступной из публичных сетей.

В этом разделе описаны дополнительные параметры, которые помогают предотвратить успех подобных атак. Все настройки выполняются в файле /etc/ssh/sshd_config

Не путайте /etc/ssh/sshd_config с /etc/ssh/ssh_config (обратите внимание на дополнительную букву `d` в первом имени файла). Первый файл настраивает сервер, а второй — клиент. Список доступных настроек клиента приведён в ssh_config(5).

По умолчанию аутентификация может выполняться как по открытому ключу, так и по паролю. Чтобы разрешить **только** аутентификацию по открытому ключу, **что настоятельно рекомендуется**, измените переменную:

```
PasswordAuthentication no
```

Хорошей практикой является ограничение пользователей, которые могут подключаться к SSH-серверу, а также мест, откуда они могут это делать, с помощью ключевого слова `AllowUsers` в конфигурационном файле сервера OpenSSH. Например, чтобы разрешить вход только пользователю `user` с адреса `192.168.1.32`, добавьте следующую строку в файл /etc/ssh/sshd_config:

```
AllowUsers user@192.168.1.32
```

Чтобы разрешить пользователю `user` входить из любого места, укажите этого пользователя без указания IP-адреса:

```
AllowUsers user
```

Несколько пользователей следует перечислять в одной строке, например:

```
AllowUsers root@192.168.1.32 user
```

После внесения всех изменений и перед перезапуском службы рекомендуется проверить правильность конфигурации, выполнив следующую команду:

```
# sshd -t
```

Если файл конфигурации верен, вывод не будет отображен. В случае, если файл конфигурации содержит ошибки, будет показано примерно следующее:

```
/etc/ssh/sshd_config: line 3: Bad configuration option: sdadasdasdasads
/etc/ssh/sshd_config: terminating, 1 bad configuration options
```

После внесения изменений и проверки корректности файла конфигурации, дайте команду `sshd` перезагрузить файл конфигурации, выполнив:

```
# service sshd reload
```

16.8. OpenSSL

OpenSSL — это набор криптографических инструментов, реализующий сетевые протоколы Secure Sockets Layer (SSL) и Transport Layer Security (TLS), а также множество криптографических функций.

Программа `openssl` — это инструмент командной строки для использования различных криптографических функций криптобиблиотеки OpenSSL из оболочки. Она может быть использована для

- Создания и управления закрытыми ключами, открытыми ключами и параметрами
- Криптографических операций с открытым ключом
- Создания сертификатов X.509, запросов на подпись сертификатов (CSR) и списков отозванных сертификатов (CRL)
- Вычисления дайджестов сообщений
- Шифрования и дешифрования с использованием шифров
- Тестирования SSL/TLS клиента и сервера
- Обработки почты с подписью или шифрованием S/MIME
- Запросов, генерации и проверки временных меток
- Бенчмаркинга криптографических процедур

Для получения дополнительной информации о OpenSSL ознакомьтесь с бесплатной книгой OpenSSL Cookbook.

16.8.1. Генерация сертификатов

OpenSSL поддерживает создание сертификатов как для проверки ЦС, так и для собственного использования.

Выполните команду openssl(1) для генерации действительного сертификата для центра сертификации (CA) с указанными аргументами. Эта команда создаст два файла в текущем каталоге. Запрос на сертификат, req.pem, можно отправить в центр сертификации (CA), который проверит введённые данные, подпишет запрос и вернёт подписанный сертификат. Второй файл, cert.key, является закрытым ключом для сертификата и должен храниться в безопасном месте. Если он попадёт в чужие руки, его можно использовать для выдачи себя за пользователя или сервер.

Выполните следующую команду для создания сертификата:

```
# openssl req -new -nodes -out req.pem -keyout cert.key -sha3-512 -newkey rsa:4096
```

Вывод должен быть похож на следующий:

```
Generating a RSA private key
.................................................................................
.........................................+++++
.....................................+++++
writing new private key to 'cert.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
```

```
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Valencian Community
Locality Name (eg, city) []:Valencia
Organization Name (eg, company) [Internet Widgits Pty Ltd]:My Company
Organizational Unit Name (eg, section) []:Systems Administrator
Common Name (e.g. server FQDN or YOUR name) []:localhost.example.org
Email Address []:user@FreeBSD.org

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:123456789
An optional company name []:Another name
```

В качестве альтернативы, если подпись от центра сертификации не требуется, можно создать самоподписанный сертификат. Это приведёт к созданию двух новых файлов в текущем каталоге: файла закрытого ключа cert.key и самого сертификата cert.crt. Эти файлы следует поместить в каталог, желательно в /etc/ssl/, с доступом только для пользователя `root`. Для этих файлов подходят права доступа `0700`, которые можно установить с помощью команды `chmod`.

Выполните следующую команду для создания сертификата:

```
# openssl req -new -x509 -days 365 -sha3-512 -keyout /etc/ssl/private/cert.key -out
/etc/ssl/certs/cert.crt
```

Вывод должен быть похож на следующий:

```
Generating a RSA private key
.....................................+++++
...........+++++
writing new private key to '/etc/ssl/private/cert.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Valencian Community
Locality Name (eg, city) []:Valencia
Organization Name (eg, company) [Internet Widgits Pty Ltd]:My Company
Organizational Unit Name (eg, section) []:Systems Administrator
Common Name (e.g. server FQDN or YOUR name) []:localhost.example.org
```

```
Email Address []:user@FreeBSD.org
```

16.8.2. Настройка поставщика FIPS

С внедрением OpenSSL 3 в базовую систему (на FreeBSD 14 и новее) в систему была добавлена новая концепция модулей-провайдеров. Помимо модуля по умолчанию, встроенного в библиотеку, модуль *legacy* реализует теперь необязательные устаревшие криптографические алгоритмы, а модуль *fips* ограничивает реализацию OpenSSL криптографическими алгоритмами, присутствующими в наборе стандартов FIPS. Эта часть OpenSSL получает особое внимание, включая список соответствующих проблем безопасности, и регулярно проходит процесс валидации FIPS 140. Также доступен список версий, прошедших валидацию FIPS. Это позволяет пользователям обеспечивать соответствие FIPS при использовании OpenSSL.

Важно отметить, что модуль fips_module(7) защищен дополнительной мерой безопасности, предотвращающей его использование без прохождения проверки целостности. Эта проверка может быть настроена системным администратором, что позволяет каждому пользователю OpenSSL 3 загружать этот модуль. Если настройка выполнена некорректно, ожидается, что модуль FIPS завершит работу следующим образом:

```
# echo test | openssl aes-128-cbc -a -provider fips -pbkdf2
```

Вывод должен быть похож на следующий:

```
aes-128-cbc: unable to load provider fips
Hint: use -provider-path option or OPENSSL_MODULES environment variable.
00206124D94D0000:error:1C8000D5:Provider routines:SELF_TEST_post:missing config
data:crypto/openssl/providers/fips/self_test.c:275:
00206124D94D0000:error:1C8000E0:Provider routines:ossl_set_error_state:fips module
entering error state:crypto/openssl/providers/fips/self_test.c:373:
00206124D94D0000:error:1C8000D8:Provider routines:OSSL_provider_init_int:self test
post failure:crypto/openssl/providers/fips/fipsprov.c:707:
00206124D94D0000:error:078C0105:common libcrypto routines:provider_init:init
fail:crypto/openssl/crypto/provider_core.c:932:name=fips
```

Проверка может быть настроена путем создания файла /etc/ssl/fipsmodule.cnf, который затем будет указан в основном конфигурационном файле OpenSSL /etc/ssl/openssl.cnf. OpenSSL предоставляет утилиту openssl-fipsinstall(1) для помощи в этом процессе, которую можно использовать следующим образом:

```
# openssl fipsinstall -module /usr/lib/ossl-modules/fips.so -out
/etc/ssl/fipsmodule.cnf
```

Вывод должен быть похож на следующий:

```
INSTALL PASSED
```

Файл /etc/ssl/openssl.cnf затем следует изменить, чтобы:

- Включить файл /etc/ssl/fipsmodule.cnf, созданный выше,
- Предоставить доступ к модулю FIPS для возможного использования,
- И явно активировать модуль по умолчанию.

```
[...]
# For FIPS
# Optionally include a file that is generated by the OpenSSL fipsinstall
# application. This file contains configuration data required by the OpenSSL
# fips provider. It contains a named section e.g. [fips_sect] which is
# referenced from the [provider_sect] below.
# Refer to the OpenSSL security policy for more information.
.include /etc/ssl/fipsmodule.cnf

[...]

# List of providers to load
[provider_sect]
default = default_sect
# The fips section name should match the section name inside the
# included fipsmodule.cnf.
fips = fips_sect

# If no providers are activated explicitly, the default one is activated implicitly.
# See man 7 OSSL_PROVIDER-default for more details.
#
# If you add a section explicitly activating any other provider(s), you most
# probably need to explicitly activate the default provider, otherwise it
# becomes unavailable in openssl.  As a consequence applications depending on
# OpenSSL may not work correctly which could lead to significant system
# problems including inability to remotely access the system.
[default_sect]
activate = 1
```

После этого можно убедиться, что модуль FIPS действительно доступен и работает:

```
# echo test | openssl aes-128-cbc -a -provider fips -pbkdf2
```

Вывод должен быть похож на следующий:

```
enter AES-128-CBC encryption password:
Verifying - enter AES-128-CBC encryption password:
```

```
U2FsdGVkX18idooW6e3LqWeeiKP76kufcOUClh57j8U=
```

Эта процедура должна повторяться каждый раз, когда модуль FIPS изменяется, например, после выполнения обновлений системы или после применения исправлений безопасности, затрагивающих OpenSSL в базовой системе.

16.9. Kerberos

Kerberos — это сетевой протокол аутентификации, изначально созданный Массачусетским технологическим институтом (MIT) для безопасной аутентификации в потенциально враждебной сети. Протокол Kerberos использует надёжное шифрование, позволяя как клиенту, так и серверу подтверждать свою подлинность без передачи незашифрованных секретов по сети. Kerberos можно охарактеризовать как систему проверки подлинности через посредника (прокси) и как систему аутентификации с доверенным третьим лицом. После аутентификации пользователя в Kerberos его передаваемые данные могут шифроваться для обеспечения конфиденциальности и целостности информации.

Единственная функция Kerberos — обеспечение безопасной аутентификации пользователей и серверов в сети. Он не предоставляет функций авторизации или аудита. Рекомендуется использовать Kerberos вместе с другими методами безопасности, которые обеспечивают сервисы авторизации и аудита.

Текущая версия протокола — версия 5, описанная в RFC 4120. Существует несколько свободных реализаций этого протокола, охватывающих широкий спектр операционных систем. MIT продолжает развивать свой пакет Kerberos. Он широко используется в США как криптографический продукт и исторически подпадал под экспортные ограничения США. В FreeBSD MITKerberos доступен в виде пакета security/krb5 или порта. Реализация Heimdal Kerberos была разработана за пределами США специально, чтобы избежать экспортных ограничений. Дистрибутив Heimdal Kerberos включён в базовую установку FreeBSD, а другая версия с более гибкими настройками доступна в коллекции портов как security/heimdal.

В Kerberos пользователи и службы идентифицируются как «принципалы», которые входят в административную группу, называемую «реалмом». Типичный принципал пользователя имеет вид `пользователь@РЕАЛМ` (реалмы традиционно пишутся в верхнем регистре).

Эта часть руководства содержит инструкции по настройке Kerberos с использованием дистрибутива Heimdal, включённого в FreeBSD.

Для демонстрации установки Kerberos пространства имён будут следующими:

- Доменная зона DNS будет `example.org`.
- Realm Kerberos будет `EXAMPLE.ORG`.

Используйте настоящие доменные имена при настройке Kerberos, даже если он будет работать внутри сети. Это позволяет избежать проблем с DNS и обеспечивает взаимодействие с другими доменами Kerberos.

16.9.1. Настройка Heimdal KDC

Центр распределения ключей (Key Distribution Center — KDC) — это централизованная служба аутентификации, предоставляемая Kerberos, «доверенная третья сторона» системы. Это компьютер, который выдаёт билеты Kerberos, используемые клиентами для аутентификации на серверах. Поскольку KDC считается доверенным для всех остальных компьютеров в области Kerberos, к нему предъявляются повышенные требования безопасности. Прямой доступ к KDC должен быть ограничен.

При работе KDC требуется мало вычислительных ресурсов, однако по соображениям безопасности рекомендуется выделить отдельный компьютер, который будет использоваться исключительно в качестве KDC.

Для начала установите пакет security/heimdal следующим образом:

```
# pkg install heimdal
```

Далее обновите /etc/rc.conf с помощью `sysrc` следующим образом:

```
# sysrc kdc_enable=yes
# sysrc kadmind_enable=yes
```

Далее отредактируйте файл /etc/krb5.conf следующим образом:

```
[libdefaults]
    default_realm = EXAMPLE.ORG
[realms]
    EXAMPLE.ORG = {
    kdc = kerberos.example.org
    admin_server = kerberos.example.org
    }
[domain_realm]
    .example.org = EXAMPLE.ORG
```

В этом примере KDC будет использовать полное доменное имя `kerberos.example.org`. Имя хоста KDC должно разрешаться в DNS.

Kerberos также может использовать DNS для поиска KDC вместо раздела `[realms]` в /etc/krb5.conf. Для крупных организаций, имеющих собственные DNS-серверы, приведённый выше пример можно сократить до:

```
[libdefaults]
      default_realm = EXAMPLE.ORG
[domain_realm]
    .example.org = EXAMPLE.ORG
```

Со следующими строками, включёнными в файл зоны `example.org`:

```
_kerberos._udp      IN  SRV     01 00 88 kerberos.example.org.
_kerberos._tcp      IN  SRV     01 00 88 kerberos.example.org.
_kpasswd._udp       IN  SRV     01 00 464 kerberos.example.org.
_kerberos-adm._tcp  IN  SRV     01 00 749 kerberos.example.org.
_kerberos           IN  TXT     EXAMPLE.ORG
```

Чтобы клиенты могли найти службы Kerberos, они *должны* иметь либо полностью настроенный файл /etc/krb5.conf, либо минимально настроенный /etc/krb5.conf *и* правильно настроенный DNS-сервер.

Далее создайте базу данных Kerberos, которая содержит ключи всех принципалов (пользователей и хостов), зашифрованные мастер-паролем. Нет необходимости запоминать этот пароль, так как он будет храниться в /var/heimdal/m-key; разумно использовать для этого случайный пароль длиной 45 символов. Чтобы создать мастер-ключ, выполните `kstash` и введите пароль:

```
# kstash
```

Вывод должен быть похож на следующий:

```
Master key: xxxxxxxxxxxxxxxxxxxxxxx
Verifying password - Master key: xxxxxxxxxxxxxxxxxxxxxxx
```

После создания мастер-ключа следует инициализировать базу данных. Утилита администрирования Kerberos kadmin(8) может быть использована на KDC в режиме, который работает напрямую с базой данных, без использования сетевого сервиса kadmind(8), как `kadmin -l`. Это решает проблему курицы и яйца, когда попытка подключения к базе данных происходит до её создания. В командной строке `kadmin` используйте `init` для создания начальной базы данных realm:

```
# kadmin -l
kadmin> init EXAMPLE.ORG
Realm max ticket life [unlimited]:
```

Наконец, оставаясь в `kadmin`, создайте первый принципал с помощью команды `add`. Пока придерживайтесь стандартных настроек для принципала, так как их можно изменить позже с помощью команды `modify`. Введите `?` в командной строке, чтобы увидеть доступные опции.

```
kadmin> add tillman
```

Вывод должен быть похож на следующий:

```
Max ticket life [unlimited]:
Max renewable life [unlimited]:
Principal expiration time [never]:
Password expiration time [never]:
Attributes []:
Password: xxxxxxxx
Verifying password - Password: xxxxxxxx
```

Далее запустите службы KDC, выполнив:

```
# service kdc start
# service kadmind start
```

Хотя на этом этапе не будет запущено никаких сервисов с поддержкой Kerberos, можно убедиться, что KDC функционирует, получив билет для только что созданного принципала:

```
% kinit tillman
```

Вывод должен быть похож на следующий:

```
tillman@EXAMPLE.ORG's Password:
```

Подтвердите успешное получение билета с помощью `klist`:

```
% klist
```

Вывод должен быть похож на следующий:

```
Credentials cache: FILE:/tmp/krb5cc_1001
    Principal: tillman@EXAMPLE.ORG

  Issued                Expires               Principal
Aug 27 15:37:58 2013  Aug 28 01:37:58 2013  krbtgt/EXAMPLE.ORG@EXAMPLE.ORG
```

Временный билет можно уничтожить после завершения теста:

```
% kdestroy
```

16.9.2. Настройка сервера для использования Kerberos

Первым шагом в настройке сервера для использования аутентификации Kerberos является проверка правильности конфигурации в файле /etc/krb5.conf. Версию с KDC можно

использовать как есть или пересоздать на новой системе.

Затем создайте файл /etc/krb5.keytab на сервере. Это основная часть процесса "керберизации" службы — она соответствует созданию общего секрета между службой и KDC. Секрет представляет собой криптографический ключ, хранящийся в "keytab". Keytab содержит хост-ключ сервера, который позволяет ему и KDC проверять подлинность друг друга. Этот файл должен быть передан на сервер безопасным способом, так как если ключ станет общедоступным, безопасность сервера может быть нарушена. Обычно keytab генерируется на доверенной машине администратора с помощью `kadmin`, а затем безопасно передаётся на сервер, например, с помощью scp(1); его также можно создать непосредственно на сервере, если это соответствует выбранной политике безопасности. Очень важно, чтобы keytab был передан на сервер безопасным способом: если ключ станет известен третьей стороне, эта сторона сможет выдавать себя за любого пользователя на сервере! Использование `kadmin` непосредственно на сервере удобно, так как запись для хостового принципала в базе данных KDC также создаётся с помощью `kadmin`.

Конечно, `kadmin` — это керберизованный сервис; для аутентификации в сетевой службе необходим билет Kerberos, но чтобы убедиться, что пользователь, запускающий `kadmin`, действительно присутствует (и его сеанс не был захвачен), `kadmin` запросит пароль для получения нового билета. Учётная запись, аутентифицирующаяся в службе kadmin, должна иметь разрешение на использование интерфейса `kadmin`, как указано в /var/heimdal/kadmind.acl. Подробнее о создании списков контроля доступа см. в разделе «Удалённое администрирование» в `info heimdal`. Вместо включения удалённого доступа к `kadmin` администратор может безопасно подключиться к KDC через локальную консоль или ssh(1) и выполнять администрирование локально с помощью `kadmin -l`.

После установки /etc/krb5.conf используйте `add --random-key` в `kadmin`. Это добавит главный серверный принципал в базу данных, но не извлечет копию ключа главного принципала в keytab. Чтобы сгенерировать keytab, используйте `ext` для извлечения ключа главного серверного принципала в его собственный keytab:

```
# kadmin
```

Вывод должен быть похож на следующий:

```
kadmin> add --random-key host/myserver.example.org
Max ticket life [unlimited]:
Max renewable life [unlimited]:
Principal expiration time [never]:
Password expiration time [never]:
Attributes []:
kadmin> ext_keytab host/myserver.example.org
kadmin> exit
```

Обратите внимание, что `ext_keytab` по умолчанию сохраняет извлечённый ключ в /etc/krb5.keytab. Это удобно при выполнении на сервере, который керберизируется, но следует использовать аргумент `--keytab путь/к/файлу`, когда извлечение keytab происходит

на другом устройстве:

```
# kadmin
```

Вывод должен быть похож на следующий:

```
kadmin> ext_keytab --keytab=/tmp/example.keytab host/myserver.example.org
kadmin> exit
```

Затем файл keytab можно безопасно скопировать на сервер с помощью scp(1) или съемного носителя. Убедитесь, что указано нестандартное имя keytab, чтобы избежать добавления ненужных ключей в системный keytab.

На этом этапе сервер может читать зашифрованные сообщения от KDC, используя свой общий ключ, хранящийся в krb5.keytab. Теперь он готов к включению служб, использующих Kerberos. Одна из самых распространённых таких служб — sshd(8), которая поддерживает Kerberos через GSS-API. В файле /etc/ssh/sshd_config добавьте строку:

```
GSSAPIAuthentication yes
```

После внесения этого изменения необходимо перезапустить `sshd(8)`, чтобы новая конфигурация вступила в силу: `service sshd restart`.

16.9.3. Настройка клиента для использования Kerberos

Как и для сервера, клиенту требуется настройка в /etc/krb5.conf. Скопируйте файл (безопасным способом) или введите его заново при необходимости.

Проверьте клиент, используя `kinit`, `klist` и `kdestroy` на клиенте, чтобы получить, отобразить, а затем удалить билет для существующего принципала. Приложения Kerberos также должны иметь возможность подключаться к серверам с поддержкой Kerberos. Если это не работает, но получение билета проходит успешно, проблема, скорее всего, на стороне сервера, а не клиента или KDC. В случае с kerberized ssh(1), GSS-API по умолчанию отключен, поэтому проверьте с помощью `ssh -o GSSAPIAuthentication=yes имя_хоста`.

При тестировании приложения с поддержкой Kerberos попробуйте использовать анализатор трафика, например `tcpdump`, чтобы убедиться, что конфиденциальная информация не передаётся в открытом виде.

Доступны различные клиентские приложения Kerberos. С появлением моста, позволяющего приложениям, использующим SASL для аутентификации, также применять механизмы GSS-API, широкий спектр клиентских приложений — от клиентов Jabber до клиентов IMAP — может использовать Kerberos для аутентификации.

Пользователи в пределах realm обычно имеют свой Kerberos-принципал, сопоставленный с локальной учётной записью. Иногда необходимо предоставить доступ к локальной учётной

записи пользователю, у которого нет соответствующего Kerberos-принципала. Например, `tillman@EXAMPLE.ORG` может потребоваться доступ к локальной учётной записи `webdevelopers`. Другие принципалы также могут нуждаться в доступе к этой локальной учётной записи.

Файлы .k5login и .k5users, размещённые в домашнем каталоге пользователя, могут быть использованы для решения этой проблемы. Например, если следующий .k5login поместить в домашний каталог пользователя `webdevelopers`, оба указанных принципала получат доступ к этой учётной записи без необходимости использования общего пароля:

```
tillman@example.org
jdoe@example.org
```

Обратитесь к ksu(1) для получения дополнительной информации о .k5users.

16.9.4. Различия MIT

Основное различие между реализациями MIT и Heimdal заключается в том, что `kadmin` имеет разные, но эквивалентные наборы команд и использует разные протоколы. Если KDC — MIT, то версия `kadmin` от Heimdal не может использоваться для удалённого администрирования KDC, и наоборот.

Клиентские приложения также могут использовать немного другие параметры командной строки для выполнения тех же задач. Рекомендуется следовать инструкциям на сайте http://web.mit.edu/Kerberos/www/. Обратите внимание на пути: порт MIT по умолчанию устанавливается в /usr/local/, а системные приложения FreeBSD будут запускаться вместо версий MIT, если `PATH` указывает на системные каталоги в первую очередь.

При использовании MIT Kerberos в качестве KDC на FreeBSD выполните следующие команды, чтобы добавить необходимые конфигурации в /etc/rc.conf:

```
# sysrc kdc_program="/usr/local/sbin/krb5kdc"
# sysrc kadmind_program="/usr/local/sbin/kadmind"
# sysrc kdc_flags=""
# sysrc kdc_enable="YES"
# sysrc kadmind_enable="YES"
```

16.9.5. Советы, хитрости и устранение неполадок Kerberos

При настройке и устранении неполадок Kerberos учитывайте следующие моменты:

- При использовании Heimdal или MITKerberos из портов убедитесь, что в `PATH` версии клиентских приложений из портов указаны перед системными версиями.
- Если временные настройки всех компьютеров в домене не синхронизированы, аутентификация может завершиться неудачей. “Синхронизация часов с помощью NTP” описывает, как синхронизировать часы с использованием NTP.
- Если имя хоста изменено, необходимо изменить принципал `host/` и обновить keytab. Это

также относится к особым записям keytab, таким как принципал `HTTP/`, используемый для пакета Apache www/mod_auth_kerb.

- Все узлы в области должны разрешаться как в прямом, так и в обратном направлении через DNS или, как минимум, присутствовать в /etc/hosts. CNAME-записи будут работать, но A- и PTR-записи должны быть корректными и присутствовать. Сообщение об ошибке для неразрешимых узлов неочевидно: `Kerberos5 refuses authentication because Read req failed: Key table entry not found`.
- Некоторые операционные системы, выступающие в роли клиентов KDC, не устанавливают для `ksu` права setuid `root`. Это означает, что `ksu` не работает. Это проблема прав доступа, а не ошибка KDC.
- С использованием MITKerberos, чтобы разрешить принципалу иметь билет сроком действия больше стандартных десяти часов, используйте `modify_principal` в командной строке kadmin(8), чтобы изменить `maxlife` как для нужного принципала, так и для принципала `krbtgt`. После этого принципал может использовать `kinit -l` для запроса билета с увеличенным сроком действия.
- При запуске сниффера пакетов на KDC для устранения неполадок во время выполнения `kinit` с рабочей станции, билет на получение билетов (TGT) отправляется немедленно, даже до ввода пароля. Это происходит потому, что сервер Kerberos свободно передаёт TGT на любой неавторизованный запрос. Однако каждый TGT зашифрован с использованием ключа, производного от пароля пользователя. Когда пользователь вводит свой пароль, он не отправляется на KDC, а вместо этого используется для расшифровки TGT, который `kinit` уже получил. Если процесс расшифровки приводит к действительному билету с корректной временной меткой, пользователь получает действительные учётные данные Kerberos. Эти учётные данные включают в себя сеансовый ключ для установления безопасного соединения с сервером Kerberos в будущем, а также сам TGT, зашифрованный собственным ключом сервера Kerberos. Этот второй уровень шифрования позволяет серверу Kerberos проверять подлинность каждого TGT.
- Принципалы хостов могут иметь более длительное время жизни билета. Если принципал пользователя имеет время жизни в неделю, а у хоста, к которому происходит подключение, время жизни составляет девять часов, кэш пользователя будет содержать просроченный принципал хоста, и кэш билетов не будет работать должным образом.
- При настройке файла krb5.dict для запрета использования определённых слабых паролей, как описано в kadmind(8), помните, что он применяется только к принципалам, для которых назначена политика паролей. Формат файла krb5.dict предполагает одну строку на каждую запись. Может быть полезно создать символическую ссылку на /usr/share/dict/words.

16.9.6. Смягчение ограничений Kerberos

Поскольку Kerberos — это подход «всё или ничего», каждая служба, включённая в сети, должна быть либо изменена для работы с Kerberos, либо защищена от сетевых атак другим способом. Это необходимо для предотвращения кражи и повторного использования учётных данных пользователей. Например, если Kerberos включён для всех удалённых оболочек, но не поддерживающий Kerberos POP3-сервер передаёт пароли в открытом виде.

KDC представляет собой единую точку отказа. По замыслу, KDC должен быть таким же

защищённым, как и его главная база данных паролей. На KDC не должно быть запущено никаких других служб, и он должен быть физически защищен. Риск очень высок, поскольку Kerberos хранит все пароли, зашифрованные одним главным ключом, который хранится в виде файла на KDC.

Скомпрометированный мастер-ключ не так страшен, как может показаться. Мастер-ключ используется только для шифрования базы данных Kerberos и в качестве затравки для генератора случайных чисел. Пока доступ к KDC защищен, злоумышленник не сможет сделать многое с мастер-ключом.

Если KDC недоступен, сетевые службы становятся непригодными к использованию, так как аутентификация не может быть выполнена. Это можно смягчить, используя один основной KDC и один или несколько подчинённых, а также тщательно реализовав вторичную или резервную аутентификацию с помощью PAM.

Kerberos позволяет пользователям, хостам и службам аутентифицироваться между собой. Однако у него нет механизма для аутентификации KDC перед пользователями, хостами или службами. Это означает, что троянская версия `kinit` может записывать все имена пользователей и пароли. Инструменты проверки целостности файловой системы, такие как security/tripwire, могут помочь смягчить эту проблему.

16.9.7. Ресурсы и дополнительная информация

- Часто задаваемые вопросы Kerberos
- Проектирование системы аутентификации: диалог в четырёх сценах
- RFC 4120, The Kerberos Network Authentication Service (V5)
- Домашняя страница MIT Kerberos
- Wiki страница проекта Heimdal Kerberos

16.10. TCP Wrappers

TCP Wrappers — это система контроля сетевого доступа на основе хоста. Перехватывая входящие сетевые запросы до их поступления к реальному сетевому сервису, TCP Wrappers определяет, разрешен или запрещен доступ для исходного IP-адреса, на основе предопределённых правил в конфигурационных файлах.

Однако, хотя TCP Wrappers обеспечивают базовый контроль доступа, их не следует рассматривать как замену более надёжным мерам безопасности. Для всесторонней защиты рекомендуется использовать передовые технологии, такие как межсетевые экраны, вместе с надлежащими методами аутентификации пользователей и системами обнаружения вторжений.

16.10.1. Начальная настройка

TCP Wrappers включены по умолчанию в inetd(8). Первым шагом будет включение inetd(8) выполнением следующих команд:

```
# sysrc inetd_enable="YES"
# service inetd start
```

Затем правильно настройте /etc/hosts.allow.

В отличие от других реализаций TCP Wrappers, использование hosts.deny в FreeBSD считается устаревшим. Все параметры конфигурации должны размещаться в /etc/hosts.allow.

В простейшей конфигурации политики подключения к демонам устанавливаются на разрешение или блокировку в зависимости от параметров в /etc/hosts.allow. Конфигурация по умолчанию в FreeBSD разрешает все подключения к демонам, запущенным через inetd.

Базовая конфигурация обычно имеет вид `daemon : address : action`, где `daemon` — это демон, запущенный inetd, `address` — допустимое имя хоста, IP-адрес или адрес IPv6, заключённый в квадратные скобки ([ ]), а `action` — либо `allow`, либо `deny`. TCP Wrappers использует семантику первого совпадения, то есть файл конфигурации сканируется с начала до первого совпадающего правила. При обнаружении совпадения правило применяется, и процесс поиска прекращается.

Например, чтобы разрешить соединения POP3 через демон mail/qpopper, в файл /etc/hosts.allow следует добавить следующие строки:

```
# This line is required for POP3 connections:
qpopper : ALL : allow
```

Всякий раз, когда редактируется этот файл, перезапустите inetd:

```
# service inetd restart
```

16.10.2. Расширенная настройка

TCP Wrappers предоставляет расширенные возможности для более тонкого управления обработкой соединений. В некоторых случаях может быть уместно отправить сообщение определённым хостам или демонам при подключении. В других ситуациях может потребоваться сделать запись в журнал или отправить письмо администратору. Также бывают случаи, когда сервис должен быть доступен только для локальных соединений. Всё это возможно благодаря использованию параметров конфигурации, известных как шаблоны (wildcards), символы подстановки и выполнение внешних команд. Для получения дополнительной информации о шаблонах и связанной с ними функциональности обратитесь к hosts_access(5).

16.11. Списки контроля доступа

Списки контроля доступа (Access Control Lists — ACL) расширяют традиционные права

доступа UNIX®, позволяя детально управлять доступом пользователей и групп к отдельным файлам или каталогам. Каждая запись ACL определяет пользователя или группу и связанные с ними права, такие как чтение, запись и выполнение. FreeBSD предоставляет команды, такие как getfacl(1) и setfacl(1), для управления ACL.

ACL полезны в сценариях, требующих более детального контроля доступа, чем стандартные разрешения, обычно используемые в многопользовательских средах или на shared-хостингах. Однако сложность может быть неизбежной, но требуется тщательное планирование, чтобы обеспечить желаемые свойства безопасности

FreeBSD поддерживает реализацию NFSv4 ACL как в UFS, так и в OpenZFS. Обратите внимание, что некоторые аргументы команды setfacl(1) работают только с POSIX ACL, а другие — с NFSv4 ACL.

16.11.1. Включение поддержки ACL в UFS

ACL включаются с помощью административного флага при монтировании `acls`, который может быть добавлен в /etc/fstab.

Следовательно, необходимо получить доступ к /etc/fstab и в разделе опций добавить флаг `acls` следующим образом:

```
# Device        Mountpoint      FStype    Options     Dump      Pass#
/dev/ada0s1a    /               ufs       rw,acls     1         1
```

16.11.2. Получение информации об ACL

Можно проверить ACL файла или каталога с помощью getfacl(1).

Например, чтобы просмотреть настройки ACL для файла ~/test, выполните следующую команду:

```
% getfacl test
```

Вывод должен быть похож на следующий в случае использования NFSv4 ACL:

```
# file: test
# owner: freebsduser
# group: freebsduser
            owner@:rw-p--aARWcCos:-------:allow
            group@:r-----a-R-c--s:-------:allow
         everyone@:r-----a-R-c--s:-------:allow
```

И вывод должен быть похож на следующий в случае использования POSIX.1e ACL:

```
# file: test
```

```
# owner: freebsduser
# group: freebsduser
user::rw-
group::r--
other::r--
```

16.11.3. Работа с ACL

setfacl(1) можно использовать для добавления, изменения или удаления ACL из файла или каталога.

Как упоминалось выше, некоторые аргументы setfacl(1) не работают с ACL NFSv4, и наоборот. В этом разделе описывается, как выполнять команды для POSIX ACL и для ACL NFSv4, а также приводятся примеры для обоих типов.

Например, чтобы установить обязательные элементы ACL по умолчанию POSIX.1e:

```
% setfacl -d -m u::rwx,g::rx,o::rx,mask::rwx directory
```

Этот другой пример устанавливает права на чтение, запись и выполнение для записи POSIX.1e ACL владельца файла, а также права на чтение и запись для группы mail в файле:

```
% setfacl -m u::rwx,g:mail:rw file
```

Чтобы сделать то же самое, что и в предыдущем примере, но с использованием ACL NFSv4:

```
% setfacl -m owner@:rwxp::allow,g:mail:rwp::allow file
```

Чтобы удалить все записи ACL, кроме трёх обязательных, из файла в POSIX.1e ACL:

```
% setfacl -bn file
```

Чтобы удалить все записи ACL в NFSv4 ACL:

```
% setfacl -b file
```

Обратитесь к getfacl(1) и setfacl(1) для получения дополнительной информации о доступных опциях этих команд.

16.12. Capsicum

Capsicum — это легковесная платформа возможностей ОС и песочницы, реализующая гибридную модель системы возможностей. Возможности (capabilities) являются не

подделываемыми токенами авторизации, которые могут быть делегированы и должны быть предъявлены для выполнения действия. Capsicum преобразует файловые дескрипторы в возможности.

Capsicum можно использовать для разделения приложений и библиотек на изолированные компоненты (песочницы), что позволяет реализовать политики безопасности и снизить последствия уязвимостей в программном обеспечении.

16.13. Учёт процессов

Учёт процессов — это метод безопасности, при помощи которого администратор может отслеживать использование системных ресурсов и их распределение между пользователями, обеспечивать мониторинг системы и минимально фиксировать выполняемые пользователями команды.

Учёт процессов имеет как положительные, так и отрицательные стороны. Один из плюсов заключается в том, что вторжение может быть локализовано до точки входа. Минус — это объём журналов, генерируемых учётом процессов, и дисковое пространство, которое они могут занять. В этом разделе рассматриваются основы учёта процессов для администратора.

Если требуется более детальный учёт, обратитесь к разделу Аудит событий безопасности.

16.13.1. Включение и использование учёта процессов

Прежде чем использовать учёт процессов, его необходимо включить с помощью следующих команд:

```
# sysrc accounting_enable=yes
# service accounting start
```

Информация учёта хранится в файлах, расположенных в /var/account, который автоматически создаётся при необходимости при первом запуске службы учёта. Эти файлы содержат конфиденциальную информацию, включая все команды, выполненные всеми пользователями. Право записи в файлы ограничено для `root`, а право чтения — для `root` и членов группы `wheel`. Чтобы также запретить членам `wheel` читать эти файлы, измените режим доступа к каталогу /var/account, разрешив доступ только для `root`.

После включения учёт начнёт собирать информацию, такую как статистика использования CPU и выполненные команды. Все журналы учёта хранятся в нечитаемом формате, который можно просмотреть с помощью sa(8). Если команда запущена без параметров, sa(8) выводит информацию о количестве вызовов для каждого пользователя, общем затраченном времени в минутах, общем времени CPU и пользователя в минутах, а также среднем количестве операций ввода-вывода. Полный список доступных параметров, управляющих выводом, смотрите в sa(8).

Для отображения команд, выполненных пользователями, используйте `lastcomm`.

Например, эта команда выводит все случаи использования `ls` пользователем `trhodes` на терминале `ttyp1`:

```
# lastcomm ls trhodes ttyp1
```

Существует множество других полезных опций, которые описаны в lastcomm(1), acct(5) и sa(8).

16.14. Ограничения ресурсов

В FreeBSD ограничения ресурсов относятся к механизмам, которые контролируют и управляют выделением различных системных ресурсов процессам и пользователям. Эти ограничения предназначены для предотвращения ситуации, когда один процесс или пользователь потребляет чрезмерное количество ресурсов, что может привести к снижению производительности или нестабильности системы. Ограничения ресурсов помогают обеспечить справедливое распределение ресурсов между всеми активными процессами и пользователями в системе.

FreeBSD предоставляет несколько методов, позволяющих администратору ограничивать объём системных ресурсов, которые может использовать отдельный пользователь.

Традиционный метод определения классов входа в систему предполагает редактирование файла /etc/login.conf. Хотя этот метод по-прежнему поддерживается, любые изменения требуют многоэтапного процесса: правки этого файла, пересборки базы данных ресурсов, внесения необходимых изменений в /etc/master.passwd и пересборки базы данных паролей. Это может быть затратным по времени в зависимости от количества настраиваемых пользователей.

rctl(8) может использоваться для более детального управления ограничениями ресурсов. Эта команда поддерживает не только пользовательские ограничения, так как также может применяться для установки ограничений ресурсов на процессы и jail.

Этот раздел демонстрирует оба метода управления ресурсами, начиная с традиционного метода.

16.14.1. Типы ресурсов

FreeBSD устанавливает ограничения для различных типов ресурсов, включая:

Таблица 30. Типы ресурсов

Тип	Описание
Время процессора	Ограничивает количество процессорного времени, которое может использовать процесс

Тип	Описание
Память	Управляет количеством физической памяти, которое может использовать процесс
Открытые файлы	Ограничивает количество файлов, которые процесс может открыть одновременно
Процессы	Управляет количеством процессов, которые пользователь или процесс может создать
Размер файла	Ограничивает максимальный размер файлов, которые процесс может создать
Дампы памяти	Управляет возможностью процессов создавать файлы дампа памяти
Сетевые ресурсы	Ограничивает количество сетевых ресурсов (например, сокетов), которые может использовать процесс

Для полного списка типов см. login.conf(5) и rctl(8).

16.14.2. Настройка классов входа

В традиционном методе классы входа и ограничения ресурсов, применяемые к классу входа, определяются в файле /etc/login.conf. Каждой учётной записи пользователя может быть назначен класс входа, где `default` является классом входа по умолчанию. Каждый класс входа имеет набор связанных с ним возможностей входа. Возможность входа — это пара `имя=значение`, где *имя* — это общеизвестный идентификатор, а *значение* — произвольная строка, которая обрабатывается соответствующим образом в зависимости от *имени*.

Первым шагом для настройки ограничения ресурсов будет открытие файла /etc/login.conf с помощью следующей команды:

```
# ee /etc/login.conf
```

Затем найдите раздел для изменяемого класса пользователей. В этом примере предположим, что класс пользователей называется `limited`; создайте его, если он не существует.

```
limited:\ ①
        :maxproc=50:\ ②
        :tc=default: ③
```

① Имя класса пользователя.

② Устанавливает максимальное количество процессов (maxproc) равным 50 для пользователей в классе `limited`.

③ Указывает, что этот класс пользователя наследует настройки по умолчанию из класса "default".

После изменения файла /etc/login.conf выполните команду cap_mkdb(1) для создания базы данных, которую FreeBSD использует для применения этих настроек:

```
# cap_mkdb /etc/login.conf
```

chpass(1) может быть использован для изменения класса пользователя на желаемый, выполнив следующую команду:

```
# chpass username
```

Это откроет текстовый редактор, где нужно добавить новый класс `limited` следующим образом:

```
#Changing user information for username.
Login: username
Password: $6$2H.419USdGaiJeqK$6kgcTnDadasdasd3YnlNZsOni5AMymibkAfRCPirc7ZFjjv
DVsKyXx26daabdfqSdasdsmL/ZMUpdHiO0
Uid [#]: 1001
Gid [# or name]: 1001
Change [month day year]:
Expire [month day year]:
Class: limited
Home directory: /home/username
Shell: /bin/sh
Full Name: User &
Office Location:
Office Phone:
Home Phone:
Other information:
```

Теперь пользователь, назначенный классу `limited`, будет иметь ограничение на максимальное количество процессов в 50. Помните, что это лишь один пример настройки ограничения ресурсов с помощью файла /etc/login.conf.

Имейте в виду, что после внесения изменений в файл /etc/login.conf пользователю необходимо выйти из системы и снова войти, чтобы изменения вступили в силу. Кроме того, всегда соблюдайте осторожность при редактировании системных конфигурационных файлов, особенно при использовании привилегированного доступа.

16.14.3. Включение и настройка ограничений ресурсов

Система rctl(8) предоставляет более детализированный способ установки и управления ограничениями ресурсов для отдельных процессов и пользователей. Она позволяет динамически назначать ограничения ресурсов конкретным процессам или пользователям, независимо от их класса.

Первым шагом для использования rctl(8) будет его включение путем добавления

следующей строки в /boot/loader.conf и перезагрузки системы:

```
kern.racct.enable=1
```

Затем включите и запустите службу rctl(8), выполнив следующие команды:

```
# sysrc rctl_enable="YES"
# service rctl start
```

Затем rctl(8) может быть использован для установки правил в системе.

Синтаксис правил (rctl.conf(5)) определяется с использованием субъекта, идентификатора субъекта, ресурса и действия, как показано в следующем примере правила:

```
subject:subject-id:resource:action=amount/per
```

Например, чтобы ограничить пользователя не более чем 10 процессами, выполните следующую команду:

```
# rctl -a user:username:maxproc:deny=10/user
```

Для проверки установленных ограничений ресурсов можно выполнить команду rctl(8):

```
# rctl
```

Вывод должен быть похож на следующий:

```
user:username:maxproc:deny=10
```

Правила сохранятся после перезагрузки, если они добавлены в /etc/rctl.conf. Формат записи — это правило без указания команды в начале. Например, предыдущее правило можно добавить так:

```
user:username:maxproc:deny=10
```

16.15. Мониторинг проблем безопасности приложений сторонних разработчиков

В последние годы в сфере безопасности было внесено множество улучшений в методы оценки уязвимостей. Угроза взлома системы возрастает по мере установки и настройки сторонних утилит практически для любой доступной сегодня операционной системы.

Оценка уязвимостей является ключевым фактором в обеспечении безопасности. Хотя FreeBSD выпускает рекомендации для базовой системы, делать это для каждой сторонней утилиты выходит за рамки возможностей проекта FreeBSD. Существует способ снизить риски уязвимостей стороннего ПО и предупредить администраторов о известных проблемах безопасности. Дополнительная утилита FreeBSD, известная как `pkg`, включает возможности, предназначенные именно для этой цели.

pkg опрашивает базу данных на наличие уязвимостей безопасности. База данных обновляется и поддерживается командой FreeBSD Security Team и разработчиками портов.

Установка предоставляет файлы конфигурации periodic(8) для поддержания базы данных pkg audit и предлагает программный метод её обновления.

После установки, а также для проверки сторонних утилит из Коллекции портов в любое время, администратор может обновить базу данных и просмотреть известные уязвимости установленных пакетов, выполнив:

```
% pkg audit -F
```

Вывод должен быть похож на следующий:

```
vulnxml file up-to-date
chromium-116.0.5845.96_1 is vulnerable:
  chromium -- multiple vulnerabilities
  CVE: CVE-2023-4431
  CVE: CVE-2023-4427
  CVE: CVE-2023-4428
  CVE: CVE-2023-4429
  CVE: CVE-2023-4430
  WWW: https://vuxml.FreeBSD.org/freebsd/5fa332b9-4269-11ee-8290-a8a1599412c6.html

samba413-4.13.17_5 is vulnerable:
  samba -- multiple vulnerabilities
  CVE: CVE-2023-3347
  CVE: CVE-2023-34966
  CVE: CVE-2023-34968
  CVE: CVE-2022-2127
  CVE: CVE-2023-34967
  WWW: https://vuxml.FreeBSD.org/freebsd/441e1e1a-27a5-11ee-a156-080027f5fec9.html

2 problem(s) in 2 installed package(s) found.
```

Направив веб-браузер по указанному URL, администратор может получить дополнительную информацию об уязвимости.

Это будет включать затронутые версии, указанные по версии порта FreeBSD, а также другие веб-сайты, которые могут содержать рекомендации по безопасности.

16.16. Рекомендации по безопасности FreeBSD

Как и многие разработчики качественных операционных систем, проект FreeBSD имеет команду безопасности, которая отвечает за определение даты окончания срока службы (EoL) для каждого выпуска FreeBSD и предоставляет обновления безопасности для поддерживаемых выпусков, которые ещё не достигли своего EoL. Дополнительная информация о команде безопасности FreeBSD и поддерживаемых выпусках доступна на странице безопасности FreeBSD.

Одна из задач команды безопасности — реагировать на сообщения об уязвимостях в операционной системе FreeBSD. После подтверждения уязвимости команда безопасности проверяет шаги, необходимые для её устранения, и обновляет исходный код с исправлением. Затем она публикует подробности в виде «Рекомендации по безопасности» (Security Advisory). Рекомендации по безопасности публикуются на сайте FreeBSD и рассылаются в списки рассылки Список рассылки FreeBSD, посвящённый срочным сообщениям, связанным с безопасностью, Список рассылки FreeBSD, посвящённый информационной безопасности и Список рассылки анонсов FreeBSD.

16.16.1. Формат рекомендации по безопасности

Вот пример рекомендации по безопасности FreeBSD:

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA512

=============================================================================
FreeBSD-SA-23:07.bhyve                                      Security Advisory
                                                          The FreeBSD Project

Topic:          bhyve privileged guest escape via fwctl

Category:       core
Module:         bhyve
Announced:      2023-08-01
Credits:        Omri Ben Bassat and Vladimir Eli Tokarev from Microsoft
Affects:        FreeBSD 13.1 and 13.2
Corrected:      2023-08-01 19:48:53 UTC (stable/13, 13.2-STABLE)
                2023-08-01 19:50:47 UTC (releng/13.2, 13.2-RELEASE-p2)
                2023-08-01 19:48:26 UTC (releng/13.1, 13.1-RELEASE-p9)
CVE Name:       CVE-2023-3494

For general information regarding FreeBSD Security Advisories,
including descriptions of the fields above, security branches, and the
following sections, please visit <URL:https://security.FreeBSD.org/>.

I.   Background

bhyve(8)'s fwctl interface provides a mechanism through which guest
firmware can query the hypervisor for information about the virtual
```

```
machine.  The fwctl interface is available to guests when bhyve is run
with the "-l bootrom" option, used for example when booting guests in
UEFI mode.

bhyve is currently only supported on the amd64 platform.

II.  Problem Description

The fwctl driver implements a state machine which is executed when the
guest accesses certain x86 I/O ports.  The interface lets the guest copy
a string into a buffer resident in the bhyve process' memory.  A bug in
the state machine implementation can result in a buffer overflowing when
copying this string.

III. Impact

A malicious, privileged software running in a guest VM can exploit the
buffer overflow to achieve code execution on the host in the bhyve
userspace process, which typically runs as root.  Note that bhyve runs
in a Capsicum sandbox, so malicious code is constrained by the
capabilities available to the bhyve process.

IV.  Workaround

No workaround is available.  bhyve guests that are executed without the
"-l bootrom" option are unaffected.

V.   Solution

Upgrade your vulnerable system to a supported FreeBSD stable or
release / security branch (releng) dated after the correction date.

Perform one of the following:

1) To update your vulnerable system via a binary patch:

Systems running a RELEASE version of FreeBSD on the amd64, i386, or
(on FreeBSD 13 and later) arm64 platforms can be updated via the
freebsd-update(8) utility:

# freebsd-update fetch
# freebsd-update install

Restart all affected virtual machines.

2) To update your vulnerable system via a source code patch:

The following patches have been verified to apply to the applicable
FreeBSD release branches.

a) Download the relevant patch from the location below, and verify the
```

```
detached PGP signature using your PGP utility.

[FreeBSD 13.2]
# fetch https://security.FreeBSD.org/patches/SA-23:07/bhyve.13.2.patch
# fetch https://security.FreeBSD.org/patches/SA-23:07/bhyve.13.2.patch.asc
# gpg --verify bhyve.13.2.patch.asc

[FreeBSD 13.1]
# fetch https://security.FreeBSD.org/patches/SA-23:07/bhyve.13.1.patch
# fetch https://security.FreeBSD.org/patches/SA-23:07/bhyve.13.1.patch.asc
# gpg --verify bhyve.13.1.patch.asc

b) Apply the patch.  Execute the following commands as root:

# cd /usr/src
# patch < /path/to/patch

c) Recompile the operating system using buildworld and installworld as
described in <URL:https://www.FreeBSD.org/handbook/makeworld.html>.

Restart all affected virtual machines.

VI.  Correction details

This issue is corrected by the corresponding Git commit hash or Subversion
revision number in the following stable and release branches:

Branch/path                             Hash                     Revision
- -------------------------------------------------------------------------
stable/13/                              9fe302d78109    stable/13-n255918
releng/13.2/                            2bae613e0da3  releng/13.2-n254625
releng/13.1/                            87702e38a4b4  releng/13.1-n250190
- -------------------------------------------------------------------------

Run the following command to see which files were modified by a
particular commit:

# git show --stat <commit hash>

Or visit the following URL, replacing NNNNNN with the hash:

<URL:https://cgit.freebsd.org/src/commit/?id=NNNNNN>

To determine the commit count in a working tree (for comparison against
nNNNNNN in the table above), run:

# git rev-list --count --first-parent HEAD

VII. References

<URL:https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2023-3494>
```

```
The latest revision of this advisory is available at
<URL:https://security.FreeBSD.org/advisories/FreeBSD-SA-23:07.bhyve.asc>
-----BEGIN PGP SIGNATURE-----

iQIzBAEBCgAdFiEEthUnfoEIffdcgYM7bljekB8AGu8FAmTJdsIACgkQbljekB8A
Gu8Q1Q/7BFw5Aa0cFxBzbdz+O5NAImj58MvKS6xw61bXcYr12jchyT6ENC7yiR+K
qCqbe5TssRbtZ1gg/94gSGEXccz5OcJGxW+qozhcdPUh2L2nzBPkMCrclrYJfTtM
cnmQKjg/wFZLUVr71GEM95ZFaktlZdXyXx9Z8eBzow5rXexpl1TTHQQ2kZZ41K4K
KFhup91dzGCIj02cqbl+1h5BrXJe3s/oNJt5JKIh/GBh5THQu9n6AywQYl18HtjV
fMb1qRTAS9WbiEP5QV2eEuOG86ucuhytqnEN5MnXJ2rLSjfb9izs9HzLo3ggy7yb
hN3tlbfIPjMEwYexieuoyP3rzKkLeYfLXqJU4zKCRnIbBIkMRy4mcFkfcYmI+MhF
NPh2R9kccemppKXeDhKJurH0vsetr8ti+AwOZ3pgO21+9w+mjE+EfaedIi+JWhip
hwqeFv03bAQHJdacNYGV47NsJ91CY4ZgWC3ZOzBZ2Y5SDtKFjyc0bf83WTfU9A/0
drC0z3xaJribah9e6k5d7lmZ7L6aHCbQ70+aayuAEZQLr/N1doB0smNi0IHdrtY0
JdIqmVX+d1ihVhJ05prC460AS/KolqiaysunligxR+ZnctE9Xdo1BlLEbYu2KjT4
LpWvSuhRMSQaYkJU72SodQc0FM5mqqNN42Vx+X4EutOfvQuRGlI=
=MlAY
-----END PGP SIGNATURE-----
```

Каждый бюллетень безопасности использует следующий формат:

- Каждое уведомление о безопасности подписано PGP-ключом офицера безопасности. Открытый ключ офицера безопасности можно проверить в OpenPGP Keys.
- Название рекомендации по безопасности всегда начинается с `FreeBSD-SA-` (для FreeBSD Security Advisory), за которым следует год в двузначном формате (`23:`), затем номер рекомендации за этот год (`07.`), а затем название затронутого приложения или подсистемы (`bhyve`).
- Поле `Topic` содержит краткое описание уязвимости.
- `Category` относится к затронутой части системы, которая может быть одной из следующих: `core`, `contrib` или `ports`. Категория `core` означает, что уязвимость затрагивает основной компонент операционной системы FreeBSD. Категория `contrib` означает, что уязвимость затрагивает программное обеспечение, включённое в состав FreeBSD, например BIND. Категория `ports` указывает, что уязвимость затрагивает программное обеспечение, доступное через Коллекцию портов.
- Поле `Module` указывает на расположение компонента. В данном примере затронут модуль `bhyve`; следовательно, эта уязвимость затрагивает приложение, установленное вместе с операционной системой.
- Поле `Announced` указывает дату публикации уведомления о безопасности. Это означает, что команда безопасности подтвердила наличие проблемы и что исправление было добавлено в репозиторий исходного кода FreeBSD.
- Поле `Credits` указывает на человека или организацию, которые обнаружили уязвимость и сообщили о ней.
- Поле `Affects` указывает, какие выпуски FreeBSD подвержены данной уязвимости.
- Поле `Corrected` указывает дату, время, смещение времени и версии, в которых была исправлена ошибка. Раздел в скобках показывает каждую ветку, в которую было внесено

исправление, и номер версии соответствующего выпуска из этой ветки. Идентификатор выпуска включает номер версии и, если необходимо, уровень патча. Уровень патча обозначается буквой `p` с последующим числом, указывающим порядковый номер патча, что позволяет пользователям отслеживать, какие патчи уже были применены к системе.

- Поле `CVE Name` содержит номер рекомендации, если он существует, в публичной базе данных уязвимостей безопасности cve.mitre.org.
- Поле `Background` содержит описание затронутого модуля.
- Поле `Описание проблемы` поясняет уязвимость. Оно может содержать информацию о проблемном коде и о том, как утилита может быть использована злоумышленниками.
- Поле `Impact` описывает, какой тип воздействия проблема может оказать на систему.
- Поле `Workaround` указывает, доступно ли временное решение для системных администраторов, которые не могут немедленно установить исправление.
- Поле `Solution` содержит инструкции по исправлению уязвимости на затронутой системе. Это пошаговый проверенный метод, позволяющий обновить систему и обеспечить её безопасную работу.
- Поле `Correction Details` отображает каждую затронутую ветку Subversion или Git с номером ревизии, содержащей исправленный код.
- Поле `References` содержит источники дополнительной информации об уязвимости.

Глава 17. Клетки и контейнеры

17.1. Обзор

Поскольку администрирование системы — сложная задача, было разработано множество инструментов, чтобы облегчить жизнь администратору. Эти инструменты часто улучшают способы установки, настройки и обслуживания систем. Один из инструментов, который можно использовать для повышения безопасности системы FreeBSD, — это *клетки (jails)*. Клетки доступны начиная с FreeBSD 4.X и продолжают совершенствоваться в плане полезности, производительности, надёжности и безопасности.

Клетки расширяют концепцию chroot(2), которая используется для изменения корневого каталога для набора процессов. Это создаёт безопасное окружение, изолированное от остальной системы. Процессы, созданные в окружении chroot, не могут обращаться к файлам или ресурсам за его пределами. По этой причине компрометация службы, работающей в окружении chroot, не должна позволить злоумышленнику скомпрометировать всю систему.

Однако `chroot` имеет несколько ограничений. Он подходит для простых задач, не требующих большой гибкости или сложных, продвинутых функций. Со временем было найдено множество способов выйти из окружения `chroot`, что делает его не самым идеальным решением для защиты сервисов.

Клетки улучшают концепцию традиционной изолированной среды chroot несколькими способами.

В традиционной среде chroot процессы ограничены только в части файловой системы, к которой они могут получить доступ. Остальные системные ресурсы, пользователи системы, запущенные процессы и подсистема сети разделяются между процессами в chroot и процессами основной системы. Клетки расширяют эту модель, виртуализируя доступ к файловой системе, набору пользователей и подсистеме сети. Доступны более детальные настройки для регулирования доступа в изолированной среде. Клетки можно рассматривать как один из видов виртуализации на уровне операционной системы.

Эта глава охватывает:

- Что такое клетка и для каких целей он может использоваться в FreeBSD.
- Типы клеток.
- Различные способы настройки сети для клетки.
- Файл конфигурации клетки.
- Как создать различные типы клетки.
- Как настроить трансляцию сетевых адресов для клеток.
- Как запустить, остановить и перезапустить клетку.
- Основы администрирования клеток, как изнутри, так и снаружи клетки.
- Как обновить различные типы клеток.

- Как ограничить ресурсы, которые может использовать клетка.
- Дополнительные параметры безопасности и функций на уровне клетки.
- Как делегировать клетке наборы данных ZFS.
- Неполный список различных менеджеров клеток FreeBSD.

17.2. Типы клеток

Некоторые администраторы разделяют клетки на различные типы, хотя базовые технологии остаются одинаковыми. Каждому администратору необходимо определить, какой тип клетки создавать в каждом конкретном случае, в зависимости от решаемой задачи.

Ниже приведён список различных типов, их характеристики и рекомендации по использованию.

17.2.1. Толстые клетки (Thick Jails)

Толстая клетка (thick jail) — это традиционная форма клетки FreeBSD. В толстой клетке полная копия базовой системы реплицируется внутри окружения клетки. Это означает, что клетка имеет свою собственную отдельную копию базовой системы FreeBSD, включая библиотеки, исполняемые файлы и конфигурационные файлы. Клетку можно рассматривать как почти полноценную автономную установку FreeBSD, но работающую в рамках хост-системы. Такая изоляция гарантирует, что процессы внутри клетки остаются отделёнными от процессов на хосте и в других клетках.

Преимущества толстых клеток:

- Высокая степень изоляции: процессы внутри клеток изолированы от хост-системы и других клеток.
- Независимость: толстые клетки могут иметь версии библиотек, настройки и программное обеспечение, отличные от хост-системы или других клеток.
- Безопасность: поскольку клетка содержит собственную базовую систему, уязвимости или проблемы, затрагивающие среду клетки, не оказывают прямого влияния на хост-систему или другие клетки.

Недостатки толстых клеток:

- Ресурсные затраты: поскольку каждая клетка поддерживает свою собственную отдельную базовую систему, толстые клетки потребляют больше ресурсов по сравнению с тонкими клетками.
- Обслуживание: каждая клетка требует собственного обслуживания и обновлений для своих базовых системных компонентов.

17.2.2. Тонкие клетки (Thin Jails)

Тонкая клетка использует базовую систему из шаблона, используя либо снимки OpenZFS,

либо монтирования NullFS. Для каждой тонкой клетки дублируется лишь минимальный поднабор базовой системы, что приводит к меньшему потреблению ресурсов по сравнению с толстой клеткой. Два механизма ведут себя по-разному после создания клеток. При методе OpenZFS каждая тонкая клетка создается как клон снимка шаблона: независимая копия с записью при копировании, которая разделяет неизмененные блоки с шаблоном, но расходится при изменении. Более поздние изменения шаблона не распространяются на существующие клоны, и каждый клон обслуживается и обновляется индивидуально, во многом как толстая клетка. При методе NullFS каждая тонкая клетка монтирует шаблон только для чтения во время выполнения, поэтому базовая система используется совместно в реальном времени, и обновление шаблона обновляет каждую клетку, которая его монтирует. Эта общая база, доступная только для чтения, также обеспечивает тонким клеткам NullFS меньшую изоляцию и независимость, чем толстым клеткам, поскольку изменения в общих компонентах одновременно затрагивают каждую клетку, использующую их.

Вкратце, тонкая клетка в FreeBSD — это тип клетки FreeBSD, который воспроизводит значительную часть, но не всю базовую систему, в изолированной среде.

Преимущества тонких клеток:

- Эффективность использования ресурсов: тонкие клетки более эффективны в использовании ресурсов по сравнению с толстыми клетками. Поскольку они используют общую базовую систему, они занимают меньше места на диске и оперативной памяти. Это позволяет запускать больше клеток на том же оборудовании без чрезмерного потребления ресурсов.
- Быстрое развёртывание: создание и запуск тонких клеток обычно происходит быстрее по сравнению с толстыми клетками. Это может быть особенно полезно при быстром развертывании множества экземпляров.
- Унифицированное обслуживание: используя метод NullFS, когда тонкие клетки используют одну копию базовой системы, доступную для чтения, обновления и обслуживание общих компонентов базовой системы (таких как библиотеки и исполняемые файлы) необходимо выполнять только один раз на шаблоне. Это упрощает процесс обслуживания по сравнению с поддержкой отдельной базовой системы для каждой толстой клетки. Тонкие клетки в клоне OpenZFS не обладают таким преимуществом: так как каждый клон независим, обновление должно быть применено к каждой клетке индивидуально, как и в случае с толстыми клетками.
- Общие ресурсы: тонкие клетки могут проще разделять общие ресурсы, такие как библиотеки и исполняемые файлы, с хост-системой. Это может привести к более эффективному использованию кэширования диска и повышению производительности приложений внутри клетки.

Недостатки тонких клеток:

- Уменьшенная изоляция: основной недостаток тонких клеток заключается в том, что они обеспечивают меньшую изоляцию по сравнению с толстыми клетками. В частности клетки с методом NullFS пользуются копией базовой системы шаблона в реальном времени, поэтому уязвимости или проблемы, затрагивающие эти общие компоненты, могут потенциально повлиять на несколько клеток одновременно.

- Проблемы безопасности: уменьшенная изоляция в тонких клетках может представлять угрозу безопасности, так как компрометация одной клетки может с большей вероятностью повлиять на другие клетки или на хост-систему.
- Конфликты зависимостей: если нескольким тонким клеткам требуются разные версии одних и тех же библиотек или программного обеспечения, управление зависимостями может усложниться. В некоторых случаях это может потребовать дополнительных усилий для обеспечения совместимости.
- Проблемы совместимости: приложения в тонкой клетке могут столкнуться с проблемами совместимости, если они рассчитаны на определённое окружение базовой системы, отличающееся от общих компонентов, предоставляемых шаблоном.

17.2.3. Сервисные клетки (Service Jails)

Сервисная клетка (service jail) напрямую разделяет всё дерево файловой системы с хостом (корневой путь клетки — /) и, таким образом, может получать доступ и изменять любые файлы на хосте, а также использует те же учётные записи пользователей, что и хост. По умолчанию у неё нет доступа к сети или другим ресурсам, ограниченным в клетках, но её можно настроить для повторного использования сети хоста и снятия некоторых ограничений клетки. Основное применение сервисных клеток — автоматическое ограничение служб/демонов внутри клетки с минимальной настройкой и без необходимости знания файлов, требуемых такой службой/демоном. Сервисные клетки доступны с версии FreeBSD 15.0.

Преимущества сервисных клеток:

- Нулевое администрирование: для службы, готовой к использованию в клетке, достаточно одной строки конфигурации в /etc/rc.conf, для службы, не готовой к использованию в клетке, требуется две строки конфигурации.
- Ресурсоэффективность: сервисные клетки более эффективны в использовании ресурсов, чем тонкие клетки, так как они не требуют дополнительного дискового пространства или сетевых ресурсов.
- Быстрое развёртывание: создание и запуск сервисных клеток обычно выполняется быстрее по сравнению с тонкими клетками, если требуется изолировать только отдельные сервисы/демоны и не нужны параллельные экземпляры одного и того же сервиса/демона.
- Общие ресурсы: сервисные клетки разделяют все ресурсы, такие как библиотеки и исполняемые файлы, с хост-системой. Это может привести к более эффективному использованию кэширования диска и повышению производительности приложений внутри клетки.
- Изоляция процессов: сервисные клетки изолируют определённую службу, она не может видеть процессы, которые не являются дочерними по отношению к этой сервисной клетке, даже если они выполняются в рамках той же учётной записи пользователя.

Недостатки сервисных клеток:

- Уменьшенная изоляция: основной недостаток сервисных клеток заключается в

отсутствии изоляции файловой системы по сравнению с толстой или тонкой клеткой.

- Проблемы безопасности: уменьшенная изоляция в сервисных клетках может создавать угрозы безопасности, поскольку компрометация одной клетки потенциально способна сильнее повлиять на всё в хост-системе.

Большая часть настройки клеток, обсуждаемая ниже, не требуется для сервисных клеток. Чтобы понять, как работают клетки, рекомендуется разобраться в этих возможностях конфигурации. Подробности о том, что необходимо для настройки сервисной клетки, приведены в разделе Настройка сервисных клеток.

17.2.4. Клетки VNET

Клетка FreeBSD VNET — это виртуализированная среда, которая обеспечивает изоляцию и контроль сетевых ресурсов для процессов, выполняющихся внутри неё. Она предоставляет высокий уровень сетевой сегментации и безопасности, создавая отдельный сетевой стек для процессов внутри клетки, что гарантирует изоляцию сетевого трафика внутри клетки от хост-системы и других клеток.

По сути, механизм VNET в FreeBSD добавляет возможность настройки сети. Это означает, что клетку VNET можно создать как толстой, так и тонкой.

17.2.5. Клетки Linux

Клетки Linux в FreeBSD — это функция в операционной системе FreeBSD, которая позволяет использовать исполняемые файлы Linux и приложения внутри клетки FreeBSD. Эта функциональность достигается за счёт совместимости, которая позволяет транслировать и выполнять определённые системные вызовы Linux и использовать библиотеки Linux с ядром FreeBSD. Цель клеток Linux — облегчить выполнение Linux-программ в системе FreeBSD без необходимости в отдельной виртуальной машине или среде Linux.

17.3. Конфигурация хоста

Прежде чем создавать клетку на хост-системе, необходимо выполнить определённые настройки и получить некоторую информацию от хост-системы.

Потребуется настроить утилиту jail(8), создать необходимые каталоги для настройки и установки клетки, получить информацию о сети хоста и проверить, использует ли хост файловую систему OpenZFS или UFS.

Версия FreeBSD, работающая в клетке, не может быть новее версии, работающей на хосте.

17.3.1. Утилита jail

Утилита jail(8) управляет клетками.

Чтобы запускать клетки при загрузке системы, выполните следующие команды:

```
# sysrc jail_enable="YES"
# sysrc jail_parallel_start="YES"
```

С `jail_parallel_start` все настроенные клетки будут запущены в фоновом режиме.

По умолчанию каждая клетка, которая имеет запись в конфигурационных файлах, запускается при загрузке. Чтобы запустить только выбранные клетки, перечислите их имена в `jail_list`:

```
# sysrc jail_list="www db"
```

Имена в `jail_list` разделяются пробелами и должны соответствовать именам клеток, определённым в /etc/jail.conf или /etc/jail.conf.d/. При загрузке запускаются только перечисленные клетки, и они запускаются в указанном порядке, поэтому любую клетку, пропущенную в `jail_list`, необходимо запускать вручную. Когда `jail_list` остаётся пустым, запускаются все клетки, определённые в файлах конфигурации.

Чтобы остановить клетки в обратном порядке их `jail_list` при завершении работы, установите `jail_reverse_stop`:

```
# sysrc jail_reverse_stop="YES"
```

С включённым `jail_parallel_start` клетки запускаются в фоновом режиме, и порядок `jail_list` больше не гарантируется. Когда одна клетка должна работать до запуска другой, выразите эту зависимость с помощью параметра `depend` в /etc/jail.conf вместо того, чтобы полагаться на порядок списка. Зависимости клетки создаются автоматически до её запуска и останавливаются в обратном порядке, независимо от `jail_list`. Параметр `depend` описан в разделе Файлы конфигурации клеток.

17.3.2. Сетевое взаимодействие

Сеть для клеток FreeBSD может быть настроена несколькими различными способами:

Режим сетевого взаимодействия хоста (разделение IP)

В режиме сетевого доступа хоста клетка использует сетевой стек хост-системы, а не получает собственный. Совместное использование стека не означает совместное использование адреса: распространённая настройка предоставляет клетке собственный выделенный `ip4.addr` (и `ip6.addr`), который jail(8) добавляет как псевдоним на интерфейсе хост-системы, поэтому клетка и хост-система отвечают на разных адресах через один и тот же стек. Только клетка, настроенная с `ip4 = inherit`, полностью видит и может пользоваться адресами хост-системы и отвечает на собственном адресе хост-системы, получая неограниченный доступ ко всем адресам, которыми владеет хост-система.

Виртуальные сети (VNET)

Виртуальные сети (VNET) — это функция клеток FreeBSD, предоставляющая более продвинутые и гибкие сетевые решения по сравнению с базовыми режимами, такими как режим сетевого взаимодействия хоста (host networking). VNET позволяет создавать изолированные сетевые стеки для каждой клетки, предоставляя им отдельные IP-адреса, таблицы маршрутизации и сетевые интерфейсы. Это обеспечивает более высокий уровень сетевой изоляции и позволяет клеткам функционировать так, как если бы они работали на отдельных виртуальных машинах.

Система netgraph

netgraph(4) — это универсальная инфраструктура ядра для создания пользовательских сетевых конфигураций. Она может использоваться для определения того, как сетевой трафик передаётся между клеткой и хост-системой, а также между различными клетками.

Откуда берется IP-адрес клетки

Клетка с общим стеком получает доступ к сети через адрес на интерфейсе хоста. В параметре `interface` указывается имя этого интерфейса, и для каждого адреса в `ip4.addr` и `ip6.addr` jail(8) сам управляет псевдонимом: как выражается jail(8), «псевдоним для каждого адреса будет добавлен к интерфейсу до создания клетки и будет удалён из интерфейса после удаления клетки». Ничто на хост-системе не обязано предварительно настраивать адрес.

Интерфейс также может быть указан для каждого адреса, так что одна строка связывает адрес с интерфейсом, который должен этот адрес использовать:

```
ip4.addr = "em0|192.168.1.151/24";
```

Когда адрес должен сохраняться независимо от клетки — потому что другая служба на хост-системе полагается на него, или потому что параметр `interface` не используется — назначьте его на хост-системе как псевдоним rc.conf и исключите его из управления jail(8):

```
# sysrc ifconfig_em0_aliases="inet 192.168.1.151/32"
```

Более старая, нумерованная форма `ifconfig_em0_alias0="inet 192.168.1.151/32"` по-прежнему работает, но форма `_aliases` предпочтительнее для новых конфигураций, поскольку она позволяет избежать хрупкого требования, чтобы нумерованные псевдонимы образовывали непрерывную последовательность `alias0`, `alias1`, `alias2`.

Адреса loopback в клетках с общей IP-адресацией

Клетка с общим стеком не имеет собственного интерфейса loopback. Ядро FreeBSD компенсирует это: в клетке без VNET, когда `127.0.0.1` или `::1` сами по себе не являются адресами клетки, ядро молча переписывает как `bind()`, так и `connect()` на адрес loopback на первый `ip4.addr` или `ip6.addr` клетки. Это поведение не документировано в jail(8), но оно важно, поскольку несет последствия для безопасности.

Не предполагайте, что привязка службы к `localhost` ограничивает её клеткой. В клетке с общим IP-адресом демон, которому указано прослушивать `127.0.0.1`, на самом деле привязывается к первому адресу клетки, который часто обращён к локальной сети, и поэтому становится доступным из сети.

Стандартным средством смягчения является предоставление клетке частного адреса loopback и указание его первым. Клонируйте выделенный интерфейс loopback на хосте и назначьте ему адрес из диапазона `127.0.0.0/8` (или частный адрес RFC 1918):

```
# sysrc cloned_interfaces="lo1"
# sysrc ifconfig_lo1="inet 127.0.1.1/32"
```

Затем укажите этот адрес первым в `ip4.addr` клетки, перед любым маршрутизируемым адресом, чтобы переопределение loopback попадало на частный адрес, а не на адрес, обращённый к локальной сети.

Подготовка сервисов хост-системы

Каждый псевдоним адреса клетки разделяет порты хост-системы. Демон хост-системы, привязанный к адресу с подстановочным знаком — показанному как `*:port` в списках прослушивателей — поэтому также прослушивает каждый псевдоним клетки, и он занимает этот порт прежде, чем собственная служба клетки сможет его заявить. Перед запуском любых клеток проверьте, что хост-система уже прослушивает:

```
# sockstat -4 -6 -l
```

Ограничьте каждый демон, который привязывается к адресу-шаблону, чтобы он использовал только собственный адрес хост-системы и оставлял адреса клеток свободными:

- syslogd(8): задайте `syslogd_flags="-ss"` в /etc/rc.conf. Одинарный `-s` уже запускает syslogd в безопасном режиме, чтобы он не прослушивал соединения с удаленных хостов для отправки сообщений журнала; согласно syslogd(8), если `-s` "указан дважды, сетевой сокет вообще не будет открыт", что также предотвращает отправку syslogd на удаленные хосты.
- sshd(8): закрепите `ListenAddress` на адресе хост-системы в файле /etc/ssh/sshd_config вместо того, чтобы оставлять его с подстановочным знаком(wildcard) по умолчанию.
- ntpd(8) и sendmail(8): привяжите их к адресу хост-системы также через их собственную конфигурацию.

Для контроля на уровне межсетевого экрана за тем, какие адреса может использовать клетка, см. Межсетевые экраны.

17.3.3. Настройка дерева каталогов клетки

Для файлов клеток не назначено какого-то заранее определённого места.

Некоторые администраторы используют /jail, другие — /usr/jail, а третьи — /usr/local/jails. В этой главе будет использоваться /usr/local/jails.

Помимо каталога /usr/local/jails будут созданы другие каталоги:

- media будет содержать сжатые файлы загруженных пользовательских окружений.
- templates будет содержать шаблоны при использовании тонких клеток.
- containers будет содержать клетки.

При использовании OpenZFS выполните следующие команды для создания наборов данных для этих каталогов:

```
# zfs create -o mountpoint=/usr/local/jails zroot/jails
# zfs create zroot/jails/media
# zfs create zroot/jails/templates
# zfs create zroot/jails/containers
```

В данном случае для родительского набора данных использовался `zroot`, но могли быть использованы и другие наборы данных.

Создание отдельного дочернего набора данных для каждой клетки в дальнейшем позволяет делать снимки, клоны, квоты и делегирование наборов данных для каждой клетки. Поскольку каталог каждой клетки затем становится собственным набором данных, OpenZFS может как ограничивать, так и гарантировать использование ею дискового пространства с помощью свойств `quota` и `reservation`; см. Ограничения ресурсов клетки.

При использовании UFS выполните следующие команды для создания каталогов:

```
# mkdir /usr/local/jails/
# mkdir /usr/local/jails/media
# mkdir /usr/local/jails/templates
# mkdir /usr/local/jails/containers
```

17.3.4. Файлы конфигурации клетки

Существует два способа настройки клеток.

Первый вариант — добавить запись для каждой клетки в файл /etc/jail.conf. Другой вариант — создать отдельный файл для каждой клетки в каталоге /etc/jail.conf.d/.

В случае, если на хост-системе мало клеток, записи для каждого клетки можно добавить в файл /etc/jail.conf. Если на хост-системе много клеток, рекомендуется создать отдельный конфигурационный файл для каждой клетки в каталоге /etc/jail.conf.d/.

Файлы в /etc/jail.conf.d/ должны иметь расширение `.conf`, и их необходимо подключить в /etc/jail.conf:

```
.include "/etc/jail.conf.d/*.conf";
```

Типичная запись с настройками клетки выглядит следующим образом:

```
jailname { ①
  # STARTUP/LOGGING
  exec.start = "/bin/sh /etc/rc"; ②
  exec.stop = "/bin/sh /etc/rc.shutdown"; ③
  exec.consolelog = "/var/log/jail_console_${name}.log"; ④

  # PERMISSIONS
  allow.raw_sockets; ⑤
  exec.clean; ⑥
  mount.devfs; ⑦

  # HOSTNAME/PATH
  host.hostname = "${name}"; ⑧
  path = "/usr/local/jails/containers/${name}"; ⑨

  # NETWORK
  ip4.addr = 192.168.1.151; ⑩
  ip6.addr = "2001:db8::151"; ⑪
  interface = em0; ⑫
}
```

① `jailname` — имя клетки.

② `exec.start` — команда(ы), выполняемые в среде клетки при её создании. Типичная команда для выполнения — "/bin/sh /etc/rc".

③ `exec.stop` — команда(ы), выполняемые в среде клетки перед её удалением. Типичная команда для выполнения — "/bin/sh /etc/rc.shutdown".

④ `exec.consolelog` — файл для вывода результатов выполнения команды (stdout и stderr).

⑤ `allow.raw_sockets` — разрешает создание raw-сокетов внутри клетки. Установка этого параметра позволяет использовать такие утилиты, как ping(8) и traceroute(8), внутри клетки.

⑥ `exec.clean` — выполнение команд в чистом окружении.

⑦ `mount.devfs` - монтирование файловой системы devfs(5) в каталог /dev внутри chroot, чтобы ограничить видимые в клетке устройства. Применяется набор правил, заданный параметром `devfs_ruleset`, или, когда этот параметр оставлен по умолчанию равным `0`, используется запасной вариант `mount.devfs` с набором правил 4, `devfsrules_jail`, который скрывает все устройства хоста, кроме минимального набора, безопасного для клетки (см. Devfs Rulesets).

⑧ `host.hostname` — имя хоста для клетки.

⑨ `path` — каталог, который будет корневым для клетки. Любые команды, выполняемые внутри клетки, либо с помощью `jail`, либо через jexec(8), запускаются из этого каталога.

⑩ `ip4.addr` - Один IPv4-адрес или разделённый запятыми список адресов, назначенных клетке. Установка `ip4.addr` подразумевает `ip4 = new`; сам параметр `ip4` не имеет значения по умолчанию (если он не указан, то его значение зависит от значений других параметров). Параметр `ip4` принимает три значения: `new` ограничивает клетку адресами, перечисленными в `ip4.addr`, `inherit` предоставляет клетке неограниченный доступ ко всем IPv4-адресам хоста, а `disable` полностью отключает IPv4 в клетке. Клетка, в которой не заданы ни `ip4`, ни `ip4.addr`, ведёт себя как при `inherit`, и параметры `interface` и `ip4.addr` не действуют при `inherit`. В клетке с несколькими адресами `ip4.saddrsel` позволяет ядру выбирать исходный адрес для исходящих соединений на несвязанных сокетах, тогда как `ip4.nosaddrsel` принудительно использует первый адрес из списка.

⑪ `ip6.addr` - Один IPv6-адрес или разделенный запятыми список адресов, назначенных клетке. Параметр `ip6` принимает те же значения `new`, `inherit` и `disable`, что и `ip4`, при этом `ip6.saddrsel` и `ip6.nosaddrsel` управляют выбором адреса источника. Используйте `ip6 = disable` для клетки только с IPv4 или `ip4 = disable` для клетки только с IPv6.

⑫ `interface` — Сетевой интерфейс для добавления IP-адресов клетки. Обычно это интерфейс хоста.

Клетка не имеет виртуальной консоли: init(8) и getty(8) не запускаются внутри неё, поэтому не к чему подключиться в интерактивном режиме. Файл `exec.consolelog` захватывает только стандартный вывод и стандартный поток ошибок команд `exec.*`, то есть вывод сценариев /etc/rc запуска и /etc/rc.shutdown завершения работы. Это не интерактивная консоль; для этого используйте jexec(8) (см. Доступ к клетке) или запустите sshd(8) внутри клетки. Собственные демоны клетки отправляют свой последующий вывод в syslog клетки.

Файлы `exec.consolelog` растут без ограничений. Выполняйте их ротацию с помощью newsyslog(8), добавив одну запись с подстановочным знаком (glob), например, в /etc/newsyslog.conf.d/jail:

```
/var/log/jail_console_*.log           640  7     100  *     J
```

Глобальные параметры, подстановочные знаки и переменные

Параметры, заданные вне любого блока клетки, являются подстановочными значениями по умолчанию: jail.conf(5) применяет их к каждой клетке, точно так же, как если бы они были записаны в разделе подстановочного знака `*`. Установка общих параметров один раз делает каждый блок клетки коротким.

Имя, перед которым стоит знак доллара и которое, при необходимости, заключено в фигурные скобки, заменяется значением этого параметра или переменной. Встроенное выражение `${name}` разворачивается в собственное имя клетки, а пользовательская переменная определяется точно так же, как параметр, но с ведущим знаком доллара.

Следующий /etc/jail.conf определяет общий заголовок один раз и переопределяет только то, что отличается для каждой клетки:

```
# Shared defaults applied to every jail
exec.start = "/bin/sh /etc/rc";
exec.stop = "/bin/sh /etc/rc.shutdown";
exec.consolelog = "/var/log/jail_console_${name}.log";
exec.clean;
mount.devfs;
host.hostname = "${name}";
path = "/usr/local/jails/containers/${name}";
interface = em0;

www {
    ip4.addr = 192.168.1.151;
}

db {
    host.hostname = "database.example.com";
    ip4.addr = 192.168.1.152;
}
```

Здесь `www` наследует всё из заголовка и задаёт только свой адрес, тогда как `db` также переопределяет `host.hostname`.

Командные хуки

jail(8) запускает клетку через последовательность перехватчиков жизненного цикла (хуков), и важно, выполняется ли каждый из хуков в среде хост-системы или внутри клетки. В порядке создания:

`exec.prestart`

: Выполняется в среде хост-системы до создания клетки.

`exec.created`

: Выполняется в среде хост-системы после создания клетки, но до запуска любого процесса внутри неё.

`exec.start`

: Выполняется внутри клетки, обычно `/bin/sh /etc/rc` для запуска её служб.

`exec.poststart`

: Выполняется в среде хост-системы после запуска клетки.

При завершении работы хуки выполняется зеркально в обратном порядке:

`exec.prestop`

: Выполняется в среде хост-системы до остановки клетки.

`exec.stop`

: Выполняется внутри клетки, обычно `/bin/sh /etc/rc.shutdown`.

`exec.poststop`

Выполняется в среде хост-системы после остановки клетки, чтобы разобрать всё, что было настроено с помощью `exec.prestart`.

Хуку может быть назначено более одной команды путём добавления с помощью `+=`:

```
exec.poststart += "logger jail ${name} started";
```

`exec.timeout` ограничивает время, в течение которого jail(8) ожидает завершения команды, а `exec.consolelog` захватывает вывод этих команд. В разделе VNET (VNET Jails) используются `exec.prestart` и `exec.poststop` для настройки и демонтажа сетевой инфраструктуры каждой клетки на хосте.

Тогда как хуки упорядочивают команды внутри одной клетки, параметр `depend` упорядочивает запуск целых клеток. Запись `depend = otherjail;` сообщает jail(8), что эта клетка требует `otherjail`: зависимость создаётся первой, до завершения её последней команды `exec.poststart`, прежде чем будет создана эта клетка, и клетки уничтожаются в обратном порядке.

Наборы правил Devfs

Каждая клетка, которая устанавливает `mount.devfs`, получает набор правил devfs, определяющий, какие узлы устройств появляются в её /dev: либо тот, который задан параметром `devfs_ruleset`, либо, когда этот параметр не установлен, значение по умолчанию для `mount.devfs` — набор правил 4. devfs.rules(5) определяет именованные наборы правил в /etc/defaults/devfs.rules. Два из них предназначены для клеток: `devfsrules_jail`, набор правил 4, является стандартным минимальным набором устройств для клетки, а `devfsrules_jail_vnet`, набор правил 5, добавляет /dev/pf, чтобы клетка VNET могла запускать собственный межсетевой экран. Примеры клеток VNET устанавливают `devfs_ruleset = 5`, а пример клетки Linux устанавливает `devfs_ruleset = 4`; примеры классической и тонкой клеток устанавливают только `mount.devfs` и поэтому используют его значение по умолчанию — набор правил 4.

Чтобы открыть доступ к дополнительному устройству, напишите пользовательский набор правил в /etc/devfs.rules, который включает один из стандартных наборов правил и показывает то, что ему необходимо. Следующий набор правил начинается с `devfsrules_jail` и дополнительно показывает устройства bpf(4):

```
[devfsrules_jail_bpf=100]
add include $devfsrules_jail
add path 'bpf*' unhide
```

Перезагрузите правила, затем укажите новый набор правил по его номеру в конфигурации клетки:

```
# service devfs restart
```

```
devfs_ruleset = 100;
```

Правила в /etc/devfs.rules переопределяют правила с теми же номерами в /etc/defaults/devfs.rules; в остальных случаях эффект двух файлов фактически объединяется.

Дополнительную информацию о переменных конфигурации можно найти в jail(8) и jail.conf(5).

17.4. Классическая клетка (Толстая клетка)

Эти клетки напоминают настоящую систему FreeBSD. Ими можно управлять почти так же, как обычной хост-системой, и обновлять независимо.

17.4.1. Создание классической клетки

В принципе, для клетки требуется только имя хоста, корневой каталог, IP-адрес и пользовательское пространство.

Пользовательское окружение для клетки можно получить с официальных серверов загрузки FreeBSD.

Выполните следующую команду, чтобы загрузить пользовательское окружение:

```
# fetch https://download.freebsd.org/releases/amd64/amd64/15.1-RELEASE/base.txz -o
/usr/local/jails/media/15.1-RELEASE-base.txz
```

После завершения загрузки необходимо извлечь содержимое в каталог клетки.

Выполните следующие команды, чтобы извлечь пользовательское окружение в каталог клетки:

```
# mkdir -p /usr/local/jails/containers/classic
# tar -xf /usr/local/jails/media/15.1-RELEASE-base.txz -C
/usr/local/jails/containers/classic --unlink
```

С извлечённой пользовательской средой в каталоге клетки потребуется скопировать файлы часового пояса и DNS-сервера:

```
# cp /etc/resolv.conf /usr/local/jails/containers/classic/etc/resolv.conf
# cp /etc/localtime /usr/local/jails/containers/classic/etc/localtime
```

Скопировав файлы, следующим шагом будет обновление до последнего уровня исправлений с помощью выполнения следующей команды:

```
# freebsd-update -b /usr/local/jails/containers/classic/ fetch install
```

Последним шагом является настройка клетки. Необходимо добавить запись в конфигурационный файл /etc/jail.conf или в jail.conf.d с параметрами клетки.

Пример может выглядеть следующим образом:

```
classic {
  # STARTUP/LOGGING
  exec.start = "/bin/sh /etc/rc";
  exec.stop = "/bin/sh /etc/rc.shutdown";
  exec.consolelog = "/var/log/jail_console_${name}.log";

  # PERMISSIONS
  allow.raw_sockets;
  exec.clean;
  mount.devfs;

  # HOSTNAME/PATH
  host.hostname = "${name}";
  path = "/usr/local/jails/containers/${name}";

  # NETWORK
  ip4.addr = 192.168.1.151;
  interface = em0;
}
```

Выполните следующую команду для запуска клетки:

```
# service jail start classic
```

Дополнительная информация об управлении клетками приведена в разделе Управление клетками.

17.4.2. Получение пользовательского окружения клетки

Процедура fetch-and-extract, показанная выше, является традиционным способом заполнения корневого каталога клетки, но это не единственный способ. Та же самая пользовательская среда может быть установлена с помощью bsdinstall(8), из базовых системных пакетов или из локально собранного дерева исходных кодов.

bsdinstall(8) предоставляет выделенную цель `jail`, которая настраивает новое пользовательское окружение в заданном каталоге для использования с jail(8). Она загружает и распаковывает наборы дистрибутивов без установки ядра, пропуская шаги разбиения диска и настройки сети обычной установки, что делает её однокомандным скриптом — альтернативой ручной последовательности загрузки и tar(1) следующим образом:

```
# bsdinstall jail /usr/local/jails/containers/classic
```

Цель `jail` по умолчанию устанавливает базовый дистрибутив (base.txz). Начиная с FreeBSD 15.0 и более поздних версий, bsdinstall(8) также может заполнять корневой каталог пакетами базовой системы с помощью своей цели `pkgbase`; метод на основе пакетов описан далее.

Начиная с FreeBSD 15.0, базовая система также публикуется в виде пакетов, поэтому пользовательское окружение клетки можно установить полностью с помощью pkg(8) из репозитория `FreeBSD-base`. Укажите pkg(8) на корневой каталог клетки с помощью глобальной опции `--rootdir` и установите вариант клетки для базового набора:

```
# pkg --rootdir /usr/local/jails/containers/classic install -r FreeBSD-base FreeBSD-
set-base-jail
```

Используйте `FreeBSD-set-minimal-jail` вместо `FreeBSD-set-base-jail` для минимального пользовательского окружения. Клетка, установленная таким образом, впоследствии обновляется и получает исправления с помощью pkg(8), а не freebsd-update(8); см. Обновление клеток pkgbase.

Пакеты базовой системы — это технологическая предварительная версия в FreeBSD 15.0. Репозиторий `FreeBSD-base` определён в /etc/pkg/FreeBSD.conf, но по умолчанию отключён, поэтому выбирайте его явно с помощью `install -r FreeBSD-base`. Целевой корневой каталог и его подкаталоги var/cache/pkg и var/db/pkg возможно, потребуется создать заранее. Традиционное распространение base.txz по-прежнему публикуется для FreeBSD 15.0, поэтому метод fetch-and-extract продолжает работать и там.

Пользовательское окружение, собранное из дерева исходных кодов, может быть установлено в корневой каталог клетки с помощью переменной `DESTDIR`, что является способом запуска пользовательского окружения `-STABLE` или собранного вручную в клетке. После сборки мира установите его и остальные дистрибутивные файлы в каталог клетки:

```
# cd /usr/src
# make buildworld
# make installworld DESTDIR=/usr/local/jails/containers/classic
# make distribution DESTDIR=/usr/local/jails/containers/classic
```

`make distribution` устанавливает файлы конфигурации по умолчанию в /etc клетки и выполняется только один раз, при заполнении клетки; повторение этого действия во время обновления перезапишет конфигурацию внутри клетки. Чтобы обновить такую клетку из исходников, следуйте той же последовательности, что и для хост-системы: запустите `installworld` с тем же `DESTDIR`, объедините файлы конфигурации с помощью `etcupdate -D`, указав корневой каталог клетки, и удалите устаревшие файлы и библиотеки с помощью `make delete-old delete-old-libs`, снова с тем же `DESTDIR`, вместо использования freebsd-

update(8). Полную процедуру обновления из исходного кода см. в Обновление FreeBSD из исходного кода.

17.4.3. Клонирование клетки как шаблона

После того как толстая клетка полностью настроена, с установленными пакетами, включёнными службами и локальной конфигурацией, она может служить шаблонным образом: шаблоном, из которого почти мгновенно развёртываются множество идентичных клеток.

В OpenZFS это делается путем создания снимка настроенного набора данных клетки и его клонирования. Это требует, чтобы исходная клетка располагалась в собственном наборе данных, поэтому создайте её с помощью `zfs create`, а не `mkdir`, как в классической процедуре клетки выше:

```
# zfs create -p zroot/jails/containers/classic
```

Остановите клетку перед созданием снимка, чтобы образ был консистентным, затем создайте снимок и клонируйте набор данных один раз для каждой новой клетки:

```
# service jail stop classic
# zfs snapshot zroot/jails/containers/classic@template
# zfs clone zroot/jails/containers/classic@template zroot/jails/containers/web1
```

На хосте UFS скопируйте дерево каталогов настроенной клетки вместо этого:

```
# cp -a /usr/local/jails/containers/classic /usr/local/jails/containers/web1
```

Каждый клон требует своей собственной записи в /etc/jail.conf, или в jail.conf.d, с уникальным именем, путем и IP-адресом. Поскольку параметры `path` и `host.hostname` в приведенном выше примере выводятся из имени блока с помощью `${name}`, отличаются только имя блока и адрес:

```
web1 {
  # STARTUP/LOGGING
  exec.start = "/bin/sh /etc/rc";
  exec.stop = "/bin/sh /etc/rc.shutdown";
  exec.consolelog = "/var/log/jail_console_${name}.log";

  # PERMISSIONS
  allow.raw_sockets;
  exec.clean;
  mount.devfs;

  # HOSTNAME/PATH
  host.hostname = "${name}";
```

```
  path = "/usr/local/jails/containers/${name}";

  # NETWORK
  ip4.addr = 192.168.1.152;
  interface = em0;
}
```

Клон шаблонного образа отличается от тонкой клетки (Тонкие клетки). Тонкая клетка продолжает использовать общую базу только для чтения и обслуживается один раз в шаблоне, тогда как каждый клон шаблонного образа представляет собой полную, независимо записываемую копию, которая обновляется и модернизируется отдельно, как описано в Обновление клетки. Клон OpenZFS изначально является эффективной по пространству копией при записи (copy-on-write), которая разделяет блоки со снимком, и сохраняет зависимость от этого исходного снимка, пока `zfs promote` не разорвёт её. Для полностью независимой копии или для репликации клетки на другой пул или хост используйте `zfs send`, переданный через конвейер в `zfs receive`, вместо `zfs clone`.

17.5. Тонкие клетки (Thin Jails)

Хотя тонкие клетки используют ту же технологию, что и толстые клетки, процедура их создания отличается. Тонкие клетки можно создавать с использованием снимков OpenZFS или шаблонов и NullFS. Использование снимков OpenZFS и шаблонов с NullFS имеет определённые преимущества перед классическими клетками, например, возможность быстрого создания из снимков или обновления нескольких клеток с помощью NullFS.

17.5.1. Создание тонкой клетки с использованием снимков OpenZFS

Благодаря хорошей интеграции между FreeBSD и OpenZFS создание новых тонких клеток с использованием снимков OpenZFS очень просто.

Для создания тонкой клетки с использованием снимков OpenZFS первым шагом является создание дерева каталогов клетки, следуя инструкциям в разделе Настройка дерева каталогов клетки.

Далее создайте шаблон. Шаблоны будут использоваться только для создания новых клеток. По этой причине они создаются в режиме "только для чтения", чтобы клетки создавались на неизменяемой основе.

Для создания набора данных для шаблона выполните следующую команду:

```
# zfs create -p zroot/jails/templates/15.1-RELEASE
```

Затем выполните следующую команду для загрузки пользовательской среды:

```
# fetch https://download.freebsd.org/releases/amd64/amd64/15.1-RELEASE/base.txz -o
/usr/local/jails/media/15.1-RELEASE-base.txz
```

После завершения загрузки необходимо извлечь содержимое в каталог шаблона, выполнив следующую команду:

```
# tar -xf /usr/local/jails/media/15.1-RELEASE-base.txz -C
/usr/local/jails/templates/15.1-RELEASE --unlink
```

Когда пользовательская среда записана в каталог шаблонов, необходимо скопировать файлы часового пояса и DNS-сервера в каталог шаблона, выполнив следующую команду:

```
# cp /etc/resolv.conf /usr/local/jails/templates/15.1-RELEASE/etc/resolv.conf
# cp /etc/localtime /usr/local/jails/templates/15.1-RELEASE/etc/localtime
```

Следующее, что нужно сделать, — обновиться до последнего уровня исправлений, выполнив следующую команду:

```
# freebsd-update -b /usr/local/jails/templates/15.1-RELEASE/ fetch install
```

После завершения обновления шаблон готов.

Для создания снимка OpenZFS из шаблона выполните следующую команду:

```
# zfs snapshot zroot/jails/templates/15.1-RELEASE@base
```

После создания снимка OpenZFS можно создавать бесконечное количество клеток с помощью функции клонирования OpenZFS.

Для создания тонкой клетки с именем `thinjail` выполните следующую команду:

```
# zfs clone zroot/jails/templates/15.1-RELEASE@base zroot/jails/containers/thinjail
```

Последним шагом является настройка клетки. Необходимо добавить запись в конфигурационный файл /etc/jail.conf или в jail.conf.d с параметрами клетки.

Пример может выглядеть следующим образом:

```
thinjail {
  # STARTUP/LOGGING
  exec.start = "/bin/sh /etc/rc";
  exec.stop = "/bin/sh /etc/rc.shutdown";
  exec.consolelog = "/var/log/jail_console_${name}.log";

  # PERMISSIONS
  allow.raw_sockets;
  exec.clean;
```

```
  mount.devfs;

  # HOSTNAME/PATH
  host.hostname = "${name}";
  path = "/usr/local/jails/containers/${name}";

  # NETWORK
  ip4 = inherit;
  interface = em0;
}
```

Выполните следующую команду для запуска клетки:

```
# service jail start thinjail
```

Дополнительная информация об управлении клетками приведена в разделе Управление клетками.

17.5.2. Создание тонкой клетки с использованием NullFS

Клетка может быть создана с уменьшенным дублированием системных файлов с использованием техники тонкой клетки и NullFS для выборочного совместного использования определённых каталогов из хост-системы в клетке.

Первым шагом является создание набора данных для сохранения шаблона, выполните следующую команду, если используется OpenZFS:

```
# zfs create -p zroot/jails/templates/15.1-RELEASE-base
```

Или эту, если используется UFS:

```
# mkdir /usr/local/jails/templates/15.1-RELEASE-base
```

Затем выполните следующую команду для загрузки пользовательской среды:

```
# fetch https://download.freebsd.org/releases/amd64/amd64/15.1-RELEASE/base.txz -o
/usr/local/jails/media/15.1-RELEASE-base.txz
```

После завершения загрузки необходимо извлечь содержимое в каталог шаблона, выполнив следующую команду:

```
# tar -xf /usr/local/jails/media/15.1-RELEASE-base.txz -C
/usr/local/jails/templates/15.1-RELEASE-base --unlink
```

Когда пользовательская среда записана в каталог шаблонов, необходимо скопировать файлы часового пояса и DNS-сервера в каталог шаблона, выполнив следующую команду:

```
# cp /etc/resolv.conf /usr/local/jails/templates/15.1-RELEASE-base/etc/resolv.conf
# cp /etc/localtime /usr/local/jails/templates/15.1-RELEASE-base/etc/localtime
```

После перемещения файлов в шаблон следующим шагом будет обновление до последнего уровня исправлений с помощью выполнения следующей команды:

```
# freebsd-update -b /usr/local/jails/templates/15.1-RELEASE-base/ fetch install
```

В дополнение к базовому шаблону также необходимо создать каталог, где будет располагаться `skeleton`. Некоторые каталоги будут скопированы из шаблона в `skeleton`.

Выполните следующую команду, чтобы создать набор данных для `skeleton` в случае использования OpenZFS:

```
# zfs create -p zroot/jails/templates/15.1-RELEASE-skeleton
```

Или эту в случае использования UFS:

```
# mkdir /usr/local/jails/templates/15.1-RELEASE-skeleton
```

Затем создайте `skeleton`-каталоги. `Skeleton`-каталоги будут содержать локальные каталоги для клеток.

Выполните следующие команды для создания каталогов:

```
# mkdir -p /usr/local/jails/templates/15.1-RELEASE-skeleton/home
# mkdir -p /usr/local/jails/templates/15.1-RELEASE-skeleton/usr
# mv /usr/local/jails/templates/15.1-RELEASE-base/etc /usr/local/jails/templates/15.1-
RELEASE-skeleton/etc
# mv /usr/local/jails/templates/15.1-RELEASE-base/usr/local
/usr/local/jails/templates/15.1-RELEASE-skeleton/usr/local
# mv /usr/local/jails/templates/15.1-RELEASE-base/tmp /usr/local/jails/templates/15.1-
RELEASE-skeleton/tmp
# mv /usr/local/jails/templates/15.1-RELEASE-base/var /usr/local/jails/templates/15.1-
RELEASE-skeleton/var
# mv /usr/local/jails/templates/15.1-RELEASE-base/root
/usr/local/jails/templates/15.1-RELEASE-skeleton/root
```

Следующий шаг — создать символьные ссылки на `skeleton`, выполнив следующие команды:

```
# cd /usr/local/jails/templates/15.1-RELEASE-base/
```

```
# mkdir skeleton
# ln -s skeleton/etc etc
# ln -s skeleton/home home
# ln -s skeleton/root root
# ln -s ../skeleton/usr/local usr/local
# ln -s skeleton/tmp tmp
# ln -s skeleton/var var
```

Когда `skeleton` готов, необходимо скопировать данные в клетку.

В случае использования OpenZFS, снимки OpenZFS могут быть использованы для простого создания необходимого количества клеток с помощью выполнения следующих команд:

```
# zfs snapshot zroot/jails/templates/15.1-RELEASE-skeleton@base
# zfs clone zroot/jails/templates/15.1-RELEASE-skeleton@base
zroot/jails/containers/thinjail
```

В случае использования UFS можно использовать программу cp(1), выполнив следующую команду:

```
# cp -R /usr/local/jails/templates/15.1-RELEASE-skeleton
/usr/local/jails/containers/thinjail
```

Затем создайте каталог, в котором будут смонтированы базовый шаблон и skeleton:

```
# mkdir -p /usr/local/jails/thinjail-nullfs-base
```

Добавьте запись о клетке в /etc/jail.conf или файл в jail.conf.d следующим образом:

```
thinjail {
  # STARTUP/LOGGING
  exec.start = "/bin/sh /etc/rc";
  exec.stop = "/bin/sh /etc/rc.shutdown";
  exec.consolelog = "/var/log/jail_console_${name}.log";

  # PERMISSIONS
  allow.raw_sockets;
  exec.clean;
  mount.devfs;

  # HOSTNAME/PATH
  host.hostname = "${name}";
  path = "/usr/local/jails/${name}-nullfs-base";

  # NETWORK
  ip4.addr = 192.168.1.153;
```

```
  interface = em0;

  # MOUNT
  mount.fstab = "/usr/local/jails/${name}-nullfs-base.fstab";
}
```

Затем создайте файл /usr/local/jails/thinjail-nullfs-base.fstab следующего содержания:

```
/usr/local/jails/templates/15.1-RELEASE-base  /usr/local/jails/thinjail-nullfs-base/
nullfs   ro          0 0
/usr/local/jails/containers/thinjail     /usr/local/jails/thinjail-nullfs-
base/skeleton nullfs  rw  0 0
```

Файл fstab управляет тем, как mount_nullfs(8) представляет базовый шаблон и скелет внутри клетки. Базовый шаблон монтируется только для чтения (`ro`), чтобы каждая клетка NullFS использовала одну неизменяемую копию базовой системы. Скелет каждой клетки монтируется с возможностью чтения и записи (`rw`), и именно здесь каждая клетка хранит свои собственные доступные для записи каталоги /etc, /var, /usr/local и другие локальные каталоги. Поскольку все клетки используют один и тот же базовый шаблон только для чтения, обновление одного шаблона сразу обновляет все клетки NullFS, в отличие от независимых клонов OpenZFS, каждый из которых необходимо обновлять отдельно.

Выполните следующую команду для запуска клетки:

```
# service jail start thinjail
```

17.6. Клетки VNET

В клетках FreeBSD VNET используется отдельный сетевой стек, включающий интерфейсы, IP-адреса, таблицы маршрутизации и правила межсетевого экрана.

Первым шагом для создания VNET-клетки является создание bridge(4) с помощью выполнения следующей команды:

```
# ifconfig bridge create
```

Вывод должен быть похож на следующий:

```
bridge0
```

Назначение IP-адреса интерфейсу, который является членом моста, устарело. Назначайте адрес хоста самому `bridge0`, а не `em0`, точно так же, как это делает приведённый ниже блок /etc/rc.conf. Такие конфигурации пока продолжают работать: FreeBSD 15.0 записывает предупреждение ядра, когда

> интерфейс с адресом добавляется в мост, и отказывает ему с EINVAL только тогда, когда sysctl net.link.bridge.member_ifaddrs изменён с его значения по умолчанию 1 на 0. bridge(4) объявляет об удалении этого sysctl в FreeBSD 16.0, после чего адреса на членах моста больше не будут разрешены.

Создав bridge, необходимо подключить его к интерфейсу em0 и включить оба, выполнив следующие команды:

```
# ifconfig bridge0 addm em0 up
# ifconfig em0 up
```

Чтобы сохранить этот параметр после перезагрузки, добавьте следующие строки в /etc/rc.conf:

```
defaultrouter="192.168.1.1"
cloned_interfaces="bridge0"
ifconfig_bridge0="inet 192.168.1.150/24 addm em0 up"
ifconfig_em0="up"
```

Для получения дополнительной информации о мостах см. Сетевые мосты.

Следующий шаг — создать клетку, как указано выше.

Можно использовать как процедуру Классической Клетки (Толстой клетки), так и процедуру Тонких Клеток. Единственное, что изменится — это конфигурация в файле /etc/jail.conf.

В качестве примера для созданной клетки будет использоваться путь /usr/local/jails/containers/vnet.

Вот пример конфигурации для VNET-клетки:

```
vnet {
  # STARTUP/LOGGING
  exec.consolelog = "/var/log/jail_console_${name}.log";

  # PERMISSIONS
  allow.raw_sockets;
  exec.clean;
  mount.devfs;
  devfs_ruleset = 5;

  # PATH/HOSTNAME
  path = "/usr/local/jails/containers/${name}";
  host.hostname = "${name}";

  # VNET/VIMAGE
  vnet;
```

```
  vnet.interface = "${epair}b";

  # NETWORKS/INTERFACES
  $id = "154"; ①
  $ip = "192.168.1.${id}/24";
  $gateway = "192.168.1.1";
  $bridge = "bridge0"; ②
  $epair = "epair${id}";

  # ADD TO bridge INTERFACE
  exec.prestart  = "/sbin/ifconfig ${epair} create up";
  exec.prestart += "/sbin/ifconfig ${epair}a up descr jail:${name}";
  exec.prestart += "/sbin/ifconfig ${bridge} addm ${epair}a up";
  exec.start    += "/sbin/ifconfig ${epair}b ${ip} up";
  exec.start    += "/sbin/route add default ${gateway}";
  exec.start    += "/bin/sh /etc/rc";
  exec.stop      = "/bin/sh /etc/rc.shutdown";
  exec.poststop = "/sbin/ifconfig ${bridge} deletem ${epair}a";
  exec.poststop += "/sbin/ifconfig ${epair}a destroy";
}
```

① Номер, **уникальный** для каждой клетки, используемый для получения как имени устройства epair (`epair${id}`), так и последнего октета адреса клетки (`192.168.1.${id}`); присвойте каждой клетке на мосту различное значение.

② Относится к ранее созданному мосту.

Параметр `vnet.interface` задает имя интерфейса, который jail(8) перемещает в сетевой стек клетки после ее создания, здесь это сторона epair, относящаяся к клетке, `${epair}b`. Этот интерфейс исчезает с хоста, пока клетка работает, и автоматически возвращается хосту при остановке клетки. Физическая сетевая карта или виртуальная функция SR-IOV могут быть переданы клетке таким же образом, обеспечивая ей выделенную сеть без моста или epair.

Строка `devfs_ruleset = 5` применяет набор правил `devfsrules_jail_vnet` из /etc/defaults/devfs.rules, который расширяет стандартный набор правил для клеток, открывая доступ к /dev/pf. Таким образом, клетка VNET может запускать собственный межсетевой экран pf(4) с `pf_enable="YES"` и приватным pf.conf, независимым от хоста. Смотрите Devfs Rulesets для описания определений наборов правил и Firewalls для информации о самом pf.

17.6.1. Автоматизация сети VNET с помощью jib

Настройка epair и подключение его к мосту вручную в `exec.prestart` и `exec.poststop`, как показано выше, становится утомительной при большом количестве клеток. В базовой системе есть вспомогательный скрипт, /usr/share/examples/jails/jib, который выполняет ту же самую настройку автоматически.

Ссылайтесь на него из хуков клетки вместо пяти самодельных строк ifconfig(8):

```
  exec.prestart += "jib addm ${name} em0";
```

```
  exec.poststop += "jib destroy ${name}";

  vnet.interface = "e0b_${name}";
```

`jib addm` создает epair, чья сторона хоста называется `e0a_<name>`, а сторона клетки — `e0b_<name>`, затем присоединяет сторону хоста к мосту, который он создает автоматически, названному в честь интерфейса-члена (для `em0` мост называется `em0bridge`). Укажите эту сторону клетки в `vnet.interface`. Когда клетка останавливается, `jib destroy` удаляет epair. Скрипт также предоставляет подкоманду `jib show` для вывода списка интерфейсов, которыми он управляет.

17.6.2. Сетевые взаимодействия Netgraph VNET с помощью jng

Третий режим работы сети, упомянутый в обзоре Сеть, использует netgraph(4) вместо bridge(4) и epair. В базовой системе поставляется второй вспомогательный скрипт, /usr/share/examples/jails/jng, который создает `ng_bridge`, подключенный к `ng_eiface` для каждой клетки:

```
  exec.prestart += "jng bridge ${name} em0";
  exec.poststop += "jng shutdown ${name}";

  vnet.interface = "ng0_${name}";
```

`jng bridge` создает узлы netgraph и интерфейс на стороне клетки с именем `ng0_<name>`, который указывается в `vnet.interface`; `jng shutdown` разрушает топологию при остановке клетки. Результат функционально эквивалентен настройке с epair и bridge, но построен на узлах netgraph, а не на bridge(4).

17.6.3. Запуск межсетевого экрана и DHCP внутри клетки VNET

Поскольку `devfs_ruleset = 5` открывает доступ к /dev/pf, клетка VNET может запускать собственный межсетевой экран. Включите его внутри клетки точно так же, как на любом хосте, с помощью `pf_enable="YES"` в /etc/rc.conf клетки и её собственном /etc/pf.conf.

Однако правила набора 5 не открывают доступ к /dev/bpf. Клетка VNET, которая получает свой адрес через DHCP, например, с помощью `ifconfig_e0b_myjail="SYNCDHCP"` в её /etc/rc.conf, запускает dhclient(8), которому требуется bpf, и поэтому он терпит неудачу при использовании набора правил 5. Следуя рецепту из Devfs Rulesets, определите собственный набор правил в /etc/devfs.rules, который начинается с `devfsrules_jail_vnet` и дополнительно открывает устройства bpf:

```
[devfsrules_jail_vnet_dhcp=101]
add include $devfsrules_jail_vnet
add path 'bpf*' unhide
```

После выполнения `service devfs restart` на хосте, укажите клетке на новый набор правил с

помощью `devfs_ruleset = 101;`.

17.7. Клетки Linux

FreeBSD может запускать пользовательские программы Linux внутри клетки, используя Двоичную совместимость с Linux и sysutils/debootstrap. Клетки не имеют собственного ядра. Они работают на ядре хоста. Поэтому необходимо включить Двоичную совместимость с Linux в хост-системе.

Чтобы включить ABI Linux при загрузке, выполните следующую команду:

```
# sysrc linux_enable="YES"
```

После включения его можно запустить без перезагрузки, выполнив следующую команду:

```
# service linux start
```

Следующим шагом будет создание клетки, как указано выше, например, в разделе Создание тонкой клетки с использованием снимков OpenZFS, но **без** выполнения настройки. Клетки FreeBSD Linux требуют особой конфигурации, которая будет подробно описана ниже.

Если на хосте ещё нет файла /etc/jail.conf — например, в системе, где клетки до сих пор создавались только из командной строки, — создайте пустой файл, чтобы service(8) затем мог остановить клетку:

```
# touch /etc/jail.conf
```

Скрипт service(8) jail читает /etc/jail.conf при остановке клетки, поэтому команда остановки завершается ошибкой, если файл не существует. Если /etc/jail.conf уже присутствует из предыдущего раздела, этот шаг не требуется.

Следующая команда jail(8) выполняет необходимую настройку для клетки и запускает её напрямую, без записи в /etc/jail.conf. Флаг `-c` создаёт новую клетку, а `-m` изменяет существующую; в сочетании как `-cm`, jail(8) создаёт клетку, если она ещё не существует, и обновляет её, если она существует:

```
# jail -cm \
    name=ubuntu \
    host.hostname="ubuntu.example.com" \
    path="/usr/local/jails/containers/ubuntu" \
    interface="em0" \
    ip4.addr="192.168.1.155" \
    exec.start="/bin/sh /etc/rc" \
    exec.stop="/bin/sh /etc/rc.shutdown" \
    mount.devfs \
    devfs_ruleset=4 \
```

```
    allow.mount \
    allow.mount.devfs \
    allow.mount.fdescfs \
    allow.mount.procfs \
    allow.mount.linprocfs \
    allow.mount.linsysfs \
    allow.mount.tmpfs \
    enforce_statfs=1
```

Получите доступ к вновь созданной клетке как `root` с помощью jexec(8):

```
# jexec -u root ubuntu sh
```

Внутри клетки установите sysutils/debootstrap и используйте его для сборки пользовательского окружения Ubuntu в /compat/ubuntu. sysutils/debootstrap — это инструмент, который собирает пользовательское окружение Linux; после этого он не требуется для входа в клетку. Первый запуск pkg(7) автоматически выполняет его начальную настройку:

```
# pkg install debootstrap
# debootstrap jammy /compat/ubuntu
```

`jammy` выбирает Ubuntu 22.04 LTS. Можно выбрать более новую LTS, например `noble` (Ubuntu 24.04), при условии, что она работает в linuxulator.

Когда процесс завершится и на консоли появится сообщение `Base system installed successfully`, из хост-системы остановите клетку. Так как эта клетка не активирована в /etc/rc.conf и еще не перечислена в /etc/jail.conf, используйте действие service(8) с префиксом `one`, которое запускает клетки, которые не активированы:

```
# service jail onestop ubuntu
```

Действия `onestart` и `onestop` являются аналогами `start` и `stop` для клеток, которые не включены через `jail_enable` или не перечислены в `jail_list`.

Затем добавьте запись в /etc/jail.conf для Linux-контейнера:

```
ubuntu {
  # STARTUP/LOGGING
  exec.start = "/bin/sh /etc/rc";
  exec.stop = "/bin/sh /etc/rc.shutdown";
  exec.consolelog = "/var/log/jail_console_${name}.log";

  # PERMISSIONS
  allow.raw_sockets;
  exec.clean;
```

```
  mount.devfs;
  devfs_ruleset = 4;

  # HOSTNAME/PATH
  host.hostname = "${name}";
  path = "/usr/local/jails/containers/${name}";

  # NETWORK
  ip4.addr = 192.168.1.155;
  interface = em0;

  # MOUNT
  mount += "devfs     $path/compat/ubuntu/dev     devfs     rw  0 0";
  mount += "tmpfs     $path/compat/ubuntu/dev/shm tmpfs     rw,size=1g,mode=1777  0
0";
  mount += "fdescfs   $path/compat/ubuntu/dev/fd  fdescfs   rw,linrdlnk 0 0";
  mount += "linprocfs $path/compat/ubuntu/proc    linprocfs rw  0 0";
  mount += "linsysfs  $path/compat/ubuntu/sys     linsysfs  rw  0 0";
  mount += "/tmp      $path/compat/ubuntu/tmp     nullfs    rw  0 0";
  mount += "/home     $path/compat/ubuntu/home    nullfs    rw  0 0";
}
```

Клетка также может представлять конкретную идентичность ядра Linux программам, работающим внутри неё. Параметр `linux` выбирает, наследует ли клетка среду эмуляции Linux хост-системы (`inherit`) или получает собственную (`new`), а `linux.osname`, `linux.osrelease` и `linux.oss_version` задают имя ОС, релиз и версию OSS, которые видят бинарные файлы Linux. Например, `linux.osrelease=6.1.0` заставляет uname(1) и проверяющие версию программы Linux внутри клетки сообщать указанный релиз ядра вместо значения `compat.linux.osrelease` для всей хоста-системы.

Затем клетку можно запустить как обычно с помощью следующей команды:

```
# service jail start ubuntu
```

Окружение Ubuntu можно запустить с помощью следующей команды:

```
# jexec ubuntu chroot /compat/ubuntu /bin/bash
```

Дополнительную информацию можно найти в главе Двоичная совместимость с Linux.

17.8. NAT и перенаправление портов для клеток

Клетка, которой присвоен частный адрес — на клонированном интерфейсе lo(4), таком как `lo1`, на частном мосту или на VNET epair во внутренней сети, — не имеет собственного маршрута в интернет. Хост должен транслировать трафик клетки в маршрутизируемый

адрес, а для доступа к службе, работающей внутри такой клетки, извне — перенаправлять входящие соединения на клетку.

Сначала разрешите хосту пересылать пакеты между его интерфейсами. Включите это при загрузке и немедленно с помощью следующих команд:

```
# sysrc gateway_enable="YES"
# sysctl net.inet.ip.forwarding=1
```

Для IPv6 используйте `ipv6_gateway_enable="YES"` и `net.inet6.ip6.forwarding=1`.

Затем настройте трансляцию сетевых адресов на хосте с помощью pf(4). Следующий /etc/pf.conf предоставляет каждой клетке в частной сети `192.168.0.0/24` исходящий доступ через внешний интерфейс хоста `em0` и публикует веб-сервер внутри клетки на порту 80 хоста:

```
ext_if = "em0"

# Translate outbound jail traffic to the host address
nat on $ext_if from 192.168.0.0/24 to any -> ($ext_if)

# Forward inbound web traffic to the jailed server
rdr on $ext_if proto tcp to port 80 -> 192.168.0.151
```

Цель `rdr` должна быть собственным адресом клетки.

Активируйте и запустите pf(4) с помощью следующей команды:

```
# sysrc pf_enable="YES"
# service pf start
```

FreeBSD 15.0 добавляет синтаксис в стиле OpenBSD для встроенных `nat-to` и `rdr-to`, который привязывает трансляцию к правилу `pass` или `match`, например `match out on $ext_if from 192.168.0.0/24 to any nat-to ($ext_if)`. Классические правила `nat on` и `rdr on`, показанные выше, работают как на FreeBSD 14.x, так и на 15.x.

Для полного рассмотрения pf(4) и ipfw(8), включая полные наборы правил и NAT на основе ipfw(8), см. Межсетевые экраны.

17.9. Сервисные клетки (Service Jails)

Сервисная клетка полностью настраивается через /etc/rc.conf или sysrc(8). Базовые системные сервисы готовы для работы в сервисных клетках. Они содержат строку конфигурации, которая включает сеть или снимает другие ограничения клеток. Базовые системные сервисы, которые не имеет смысла запускать внутри клеток, настроены так,

чтобы не запускаться как сервисные клетки, даже если они включены в /etc/rc.conf. Некоторые примеры таких сервисов — это сервисы, которые хотят монтировать или размонтировать что-то в методе start или stop, или только настраивают что-то, например маршрут, межсетевой экран или подобное.

Сторонние службы могут быть или не быть готовы к использованию в сервисных клетках. Чтобы проверить, готова ли служба к работе в сервисной клетке, можно использовать следующую команду:

```
# grep _svcj_options /path/to/rc.d/servicename
```

Если вывод отсутствует, служба не готова к работе в клетке или не требует дополнительных привилегий, таких как, например, доступ к сети.

Если служба не готова к работе в клетке и требует доступа к сети, её можно подготовить, добавив необходимую конфигурацию в /etc/rc.conf:

```
# sysrc servicename_svcj_options=net_basic
```

Для всех возможных `_svcj_options` см. справочную страницу rc.conf(5).

Для включения сервисной клетки для указанной службы необходимо остановить службу и установить переменную `servicename_svcj` в значение YES. Чтобы поместить syslogd(8) в сервисную клетку, используйте следующую последовательность команд:

```
# service syslogd stop
# sysrc syslogd_svcj=YES
# service syslogd start
```

Если переменная `servicename_svcj` изменяется, службу необходимо остановить перед внесением изменений. Если она не будет остановлена, rc-фреймворк не определит корректное состояние службы и не сможет выполнить запрошенное действие.

Сервисные клетки управляются только через rc.conf(5)/sysrc(8) и команду service(8). Утилиты для работы с клетками, такие как jls(8), описанные в разделе Управление клетками, могут использоваться для проверки их работы, но команда jail(8) не предназначена для управления ими.

17.10. Управление клетками

После создания клетки можно выполнить ряд операций, таких как запуск, перезагрузка или удаление клетки, установка программного обеспечения в неё и т.д. В этом разделе описаны различные действия, которые можно выполнять с клетками с хоста.

17.10.1. Список работающих клеток

Для вывода списка клеток, запущенных в хост-системе, можно использовать команду jls(8):

```
# jls
```

Вывод должен быть похож на следующий:

```
   JID  IP Address      Hostname                      Path
     1  192.168.1.151  classic
/usr/local/jails/containers/classic
```

jls(8) поддерживает аргумент `--libxo`, который через библиотеку libxo(3) позволяет отображать данные в других форматах, таких как `JSON`, `HTML` и т.д.

Например, выполните следующую команду для получения вывода в формате `JSON`:

```
# jls --libxo=json
```

Вывод должен быть похож на следующий:

```
{"__version": "2", "jail-information": {"jail":
[{"jid":1,"ipv4":"192.168.1.151","hostname":"classic","path":"/usr/local/jails/contain
ers/classic"}]}}
```

Стандартный вывод является лишь малой частью того, что может сообщить jls(8). Добавьте флаг `-v` для подробной многострочной сводки по каждой клетке, включающей её состояние и назначенный идентификатор набора процессоров (cpuset ID):

```
# jls -v
```

Используйте `-h` для вывода строки заголовка с названиями полей над значениями запрашиваемых параметров:

```
# jls -h jid name path
```

Чтобы запросить конкретные параметры одной клетки, укажите её с помощью `-j` и перечислите параметры для вывода:

```
# jls -j classic path host.hostname ip4.addr
```

Вывод должен быть похож на следующий:

```
/usr/local/jails/containers/classic classic 192.168.1.151
```

Добавьте `-d`, чтобы включить клетки, которые завершают работу и в противном случае скрыты, пока находятся в состоянии завершения:

```
# jls -d
```

17.10.2. Запуск, перезапуск и остановка клетки

service(8) используется для запуска, перезагрузки или остановки клетки на хосте.

Например, чтобы запустить клетку, выполните следующую команду:

```
# service jail start jailname
```

Измените аргумент `start` на `restart` или `stop`, чтобы выполнить другие действия с клеткой.

Остановка клетки — это не просто завершение процессов внутри неё. Когда клетка останавливается, jail(8) сначала выполняет команды из её параметра `exec.stop` — обычно `/bin/sh /etc/rc.shutdown` — внутри клетки, чтобы её службы корректно завершились. Затем он отправляет `SIGTERM` всем процессам, всё ещё работающим в клетке, и ждёт до `stop.timeout` секунд (по умолчанию 10) их завершения. По истечении этого времени клетка удаляется, что завершает оставшиеся процессы сигналом `SIGKILL`. Если `stop.timeout` установлен в ноль, `SIGTERM` не отправляется, и клетка удаляется немедленно. Именно поэтому клетка с медленно останавливающимся демоном может казаться зависшей на десять секунд при остановке; увеличьте `stop.timeout` в /etc/jail.conf для клетки, в которой работает база данных или другая служба, требующая больше времени для корректного завершения.

17.10.3. Удаление клетки

Удаление клетки, это не просто остановка клетки с помощью service(8) и удаление каталога клетки и записи в /etc/jail.conf.

FreeBSD очень серьезно относится к безопасности системы. По этой причине существуют определённые файлы, которые не может удалить даже пользователь root. Эта функциональность называется Флаги Файлов. В клетках, чьи файлы находятся на файловой системе UFS, эти флаги должны быть очищены до удаление каталога клетки.

Первым шагом является остановка нужной клетки с помощью выполнения следующей команды:

```
# service jail stop classic
```

Второй шаг — удалить эти флаги с помощью chflags(1), выполнив следующую команду, где `classic` — имя удаляемой клетки:

```
# chflags -R 0 /usr/local/jails/containers/classic
```

Третий шаг — удалить каталог, в котором находилась клетка:

```
# rm -rf /usr/local/jails/containers/classic
```

Два шага выше относятся к клетке, хранящейся в файловой системе UFS. Клетка, поддерживаемая собственным набором данных ZFS — как созданная в Настройка дерева каталогов клетки — удаляется вместо этого путем уничтожения набора данных. Уничтожение набора данных не блокируется файловыми флагами, поэтому шаг с chflags(1) в этом случае не требуется:

```
# service jail stop classic
# zfs destroy zroot/jails/containers/classic
```

Добавьте `-r`, чтобы также уничтожить любые снимки, которые хранит набор данных.

Тонкая клетка на ZFS, созданная клонированием снимка шаблона (см. Тонкие клетки), зависит от этого снимка. Пока существует любой клон, zfs(8) отказывается уничтожать снимок шаблона и выводит список зависимых клонов. Чтобы удалить шаблон, сначала уничтожьте каждую клетку, клонированную из него, или выполните `zfs promote` на клоне, чтобы он больше не зависел от шаблона.

Наконец, в любом случае удалите запись клетки в /etc/jail.conf или в /etc/jail.conf.d. Для тонкой клетки NullFS также удалите её файл .fstab и её теперь пустой каталог точки монтирования.

17.10.4. Работа с пакетами в клетке

Утилита pkg(8) поддерживает аргумент `-j` для управления пакетами, установленными внутри клетки.

Например, чтобы установить пакет www/nginx-lite в клетке, можно выполнить следующую команду **с хоста**:

```
# pkg -j classic install nginx-lite
```

Для получения дополнительной информации о работе с пакетами в FreeBSD см. Установка приложений: Пакеты и порты.

17.10.5. Управление службами в клетке с хост-системы

Службы внутри клетки можно включать и контролировать с хост-системы, не заходя

сначала в клетку через jexec(8). Как service(8), так и sysrc(8) принимают аргумент `-j`, который указывает целевую клетку по ID или имени и работает с конфигурацией rc(8) этой клетки.

Продолжая пример с пакетами выше, включите и запустите nginx в клетке `classic` непосредственно с хост-системы:

```
# sysrc -j classic nginx_enable=YES
# service -j classic nginx start
```

Эти команды имеют тот же эффект, что и запуск sysrc(8) и service(8) через jexec(8), но они сохраняют единообразие администрирования и написания сценариев на стороне хост-системы с системой без клеток.

17.10.6. Доступ к клетке

Как уже упоминалось выше, оптимальным является управление клетками из хост-системы, однако в клетку можно войти с помощью jexec(8).

В клетку можно войти, выполнив jexec(8) с хоста:

```
# jexec -u root jailname
```

Флаг `-u` выбирает пользователя, от имени которого выполняется команда, разрешая имя через базу данных паролей **хост-системы**. Чтобы разрешить пользователя через **собственную** базу данных паролей клетки, используйте `-U`. Если команда не задана, как в данном случае, jexec(8) запускает оболочку входа этого пользователя внутри клетки. Добавьте `-l`, чтобы запустить её в чистой среде входа, сохраняющей только `HOME`, `SHELL`, `TERM` и `USER`, сбрасывающей `PATH` на /bin:/usr/bin и ведущей себя как новый вход; `-d` задаёт рабочую директорию, по умолчанию это корень клетки.

При входе в клетку будет отображено сообщение, настроенное в motd(5).

17.10.7. Выполнение команд в клетке

Для выполнения команды в клетке из хост-системы можно использовать jexec(8).

Например, чтобы остановить службу, работающую внутри клетки, будет выполнена команда:

```
# jexec -l jailname service nginx stop
```

17.10.8. Просмотр процессов и сокетов в клетках с хост-системы

С хост-системы процессы и сокеты каждой клетки видны и могут фильтроваться по клетке.

ps(1) принимает `-J` для ограничения своего вывода одной клеткой, заданной по JID или имени:

```
# ps -J classic
```

Вывод должен быть похож на следующий, где `J` в столбце `STAT` отмечает процесс, запущенный в клетке:

```
  PID TT  STAT    TIME COMMAND
 2841  -  IsJ  0:00.02 /usr/sbin/syslogd -ss
 2903  -  IsJ  0:00.00 nginx: master process /usr/local/sbin/nginx
```

Используйте `ps -J 0`, чтобы показать только процессы самой хост-системы. Ключевые слова `jail` и `jid` также можно добавить в формат вывода `-o`, чтобы показать, к какой клетке относится каждый процесс. Например, чтобы вывести список всех процессов на хост-системе вместе с их клеткой:

```
# ps ax -o pid,jail,command
```

top(1) принимает `-J` для фильтрации своего живого отображения по одной клетке, что подразумевает `-j` и добавляет столбец `JID`:

```
# top -J classic
```

sockstat(1) выводит список сокетов, принадлежащих клетке, с помощью `-j`, что является естественным способом проверить, какие порты демоны клетки фактически привязали после запуска в ней службы:

```
# sockstat -4 -6 -l -j classic
```

Изнутри клетки картина обратная: процесс, заключенный в клетку, видит только процессы и сокеты своей собственной клетки и никогда не видит процессы или сокеты хоста или других клеток.

17.10.9. Управление клетками напрямую с помощью утилиты jail

Представленная выше обёртка service(8) — это обычный способ запуска клеток при загрузке, но jail(8) также может создавать, изменять и удалять клетки напрямую, что полезно для написания скриптов и тестирования конфигурации без фреймворка rc.

Создайте настроенную клетку по имени:

```
# jail -c classic
```

Удалите работающую клетку; это также удалит все дочерние клетки, которые она создала, и завершит процессы внутри неё:

```
# jail -r classic
```

Флаг `-m` изменяет параметры уже запущенной клетки без её перезапуска. Например, чтобы добавить второй IPv4-адрес к запущенной клетке:

```
# jail -m name=classic ip4.addr+=192.168.250.71
```

jail(8) отмечает, что «некоторые параметры нельзя изменить в работающей клетке»; на практике это включает `path` и `vnet`, для изменения которых требуется перезапуск клетки.

Наконец, `-e` отображает все настроенные не-шаблонные клетки и их параметры без создания, изменения или удаления чего-либо, что удобно для проверки файла конфигурации. В качестве аргумента она принимает строку, используемую для разделения параметров:

```
# jail -e ,
```

17.10.10. Резервное копирование и перенос клетки

Клетка в значительной степени самодостаточна, поэтому её резервное копирование или перенос на другой хост в основном сводится к копированию её файлов и её конфигурации.

Остановите клетку сначала для получения консистентной копии; клетку на основе ZFS можно вместо этого сделать снимок во время её работы. Для клетки на собственном наборе данных ZFS выполните рекурсивный снимок и передайте его на целевой хост:

```
# service jail stop classic
# zfs snapshot -r zroot/jails/containers/classic@migrate
# zfs send -R zroot/jails/containers/classic@migrate | ssh newhost zfs receive
zroot/jails/containers/classic
```

Полная механика zfs-send(8) и zfs-receive(8), включая инкрементальные передачи, описана в Репликация.

Для клетки на файловой системе UFS используйте tar(1) с опцией `-p`, чтобы сохранить владельца, права доступа и флаги файлов:

```
# tar -cpf classic.tar -C /usr/local/jails/containers classic
```

Извлеките архив как `root` на целевой системе с помощью `tar -xpf`, чтобы сохранились флаги. dump(8) и restore(8) являются альтернативой, когда клетка занимает выделенную

файловую систему UFS; они работают с целыми файловыми системами, поэтому не применимы к структуре, используемой в этой главе, где все клетки являются каталогами в одной общей файловой системе.

Наконец, скопируйте блок строк, относящихся к клетке, из /etc/jail.conf — или его файл в /etc/jail.conf.d — на новый хост, настройте любые параметры, специфичные для хоста, такие как `interface`, `ip4.addr` или мост, используемый клеткой VNET, и запустите клетку там.

Целевая хост-система не должна запускать более старую версию FreeBSD, чем пользовательское окружение клетки, следуя тому же правилу, что клетка никогда не может запускать более новую версию, чем её хост-система. Тонкая клетка на NullFS не является автономной: её шаблон и скелет также должны существовать на целевой хост-системе, поэтому перенесите и их, или сначала преобразуйте клетку в толстую копию. Любые правила rctl(8), хранящиеся в /etc/rctl.conf, должны быть перенесены отдельно.

17.11. Администрирование изнутри клетки

Большая часть администрирования клеток выполняется с хост-системы, но некоторые задачи выполняются изнутри самой клетки, и скрипт, запущенный внутри клетки, иногда должен знать, что он находится в клетке. В этом разделе описывается, что администратор видит и может делать изнутри клетки.

Процесс может определить, находится ли он внутри клетки, прочитав sysctl `security.jail.jailed`, который равен `1` внутри клетки и `0` на хост-системе:

```
# sysctl security.jail.jailed
security.jail.jailed: 1
```

Внутри клетки окружение намеренно ограничено:

- Вид на хост-систему скрыт. Процесс, находящийся в клетке, видит только процессы, сокеты и файловые системы своей собственной клетки, но никогда не видит процессы, сокеты и файловые системы хост-системы или других клеток.
- Состояние ядра в значительной степени доступно только для чтения. Настраиваемые параметры ядра не могут быть записаны изнутри клетки, kldload(8) и kldunload(8) не разрешены, а dmesg(8) и буфер сообщений ядра недоступны, если не установлен параметр `allow.read_msgbuf`.
- Привилегированные операции, которые пересекали бы границу клетки, запрещены, если только хост-система явно не разрешает их. Создание raw-сокетов — как того требуют ping(8) и traceroute(8) — требует `allow.raw_sockets` (см. описание параметра в разделе Файлы конфигурации клеток), монтирование файловых систем требует `allow.mount` и его дочерних параметров для конкретных файловых систем, а chflags(1) по умолчанию отключен (см. Обновление клеток).

Горстка переменных sysctl, относящихся к каждой клетке, может быть изменена изнутри клетки без влияния на хост-систему, включая `kern.hostname`, `kern.domainname`, `kern.hostid`, `kern.hostuuid`, `kern.securelevel` и `security.bsd.suser_enabled`. Текущий набор доступных параметров клетки можно вывести с помощью sysctl(8):

```
# sysctl -d security.jail.param
```

Управление собственными службами клетки изнутри работает точно так же, как и в системе вне клетки: service(8) и sysrc(8) работают с собственной конфигурацией rc(8) клетки. Это внутриклеточный аналог показанных выше service -j и sysrc -j из хост-системы.

17.12. Обновление клетки

Обновление клеток FreeBSD гарантирует, что изолированные среды остаются безопасными, актуальными и соответствуют последним функциям и улучшениям, доступным в экосистеме FreeBSD.

Правильный инструмент зависит от того, как была установлена пользовательская среда клетки. Клетки, установленные из набора дистрибутивов (файл base.txz, извлечённый в разделах создания, или созданный с помощью bsdinstall(8)), обновляются с хоста с помощью freebsd-update(8) через его флаг `-j`, как на FreeBSD 14.x, так и на 15.x. Клетки, установленные из базовых пакетов, доступных с FreeBSD 15.0, обновляются с помощью pkg(8) вместо этого; freebsd-update(8) к ним не применяется. Клетки, собранные из исходного кода, обновляются с помощью процедуры для исходного кода (`make installworld` и `make distribution`). Никогда не смешивайте эти механизмы в одной клетке.

Поддержка наборов дистрибутивов планируется к удалению в FreeBSD 16, но будет продолжена, вместе с freebsd-update(8), на протяжении всего срока жизни стабильной ветки FreeBSD 15. Базовые пакеты предлагаются как технологическая превью в FreeBSD 15.0 и, как ожидается, станут стандартным методом в будущих выпусках, поэтому pkgbase — это перспективный путь.

В OpenZFS выполните снимок набора данных клетки перед обновлением, чтобы неудачное обновление можно было мгновенно отменить:

```
# zfs snapshot zroot/jails/containers/classic@preupgrade
```

Если обновление не удалось, остановите клетку и выполните `zfs rollback zroot/jails/containers/classic@preupgrade`, чтобы восстановить предыдущее состояние. Для этого требуется, чтобы клетка была отдельным набором данных: тонкая клетка OpenZFS уже является таковым, но классическая клетка, созданная с помощью mkdir(1), не является, поэтому создайте её с помощью `zfs create` вместо этого. Для тонкой клетки NullFS сделайте снимок набора данных шаблона. zfs-rollback(8) откатывается только к самому последнему снимку, если не указан параметр `-r` для уничтожения более

новых.

17.12.1. Обновление классической клетки или тонкой клетки с использованием снимков OpenZFS

Клетки **должны обновляться их операционной системы хоста**. В FreeBSD по умолчанию запрещено использование chflags(1) в клетке. Это предотвращает обновление некоторых файлов, поэтому обновление изнутри клетки завершится ошибкой.

Для обновления клетки до последнего патч-релиза версии FreeBSD, под которой он работает, выполните следующие команды на хосте:

```
# freebsd-update -j classic fetch install
# service jail restart classic
```

Для обновления клетки до новой основной или промежуточной версии сначала обновите хост-систему, как описано в Выполнение обновлений основной и промежуточной версий. После обновления и перезагрузки хоста можно обновить клетку.

В случае обновления с одной версии на другую проще создать новую клетку, чем выполнять полное обновление.

Например, для обновления с `классической` клетки до 15.1-RELEASE выполните следующие команды на хосте:

```
# freebsd-update -j classic -r 15.1-RELEASE upgrade
# freebsd-update -j classic install
# service jail restart classic
# freebsd-update -j classic install
# service jail restart classic
```

Шаг `install` намеренно запускается дважды. Клетка не имеет собственного ядра (она работает на ядре хоста), поэтому здесь ничего не обновляет ядро. Первый `install` записывает новое пользовательское окружение в клетку; после того как клетка перезапускается на обновлённых файлах, второй `install` завершает замену компонентов, которые нельзя было изменить, пока их предыдущие версии всё ещё использовались.

Затем, если это было обновление основной версии, переустановите все установленные пакеты и перезапустите клетку снова. Это необходимо, потому что версия ABI изменяется при обновлении между основными версиями FreeBSD.

С хоста:

```
# pkg -j classic upgrade -f
```

```
# service jail restart classic
```

17.12.2. Обновление тонкой клетки OpenZFS

Тонкие клетки, созданные в Создание тонкой клетки с помощью снимков OpenZFS, являются клонами OpenZFS снимка шаблона, но после создания клон независим от этого шаблона. Обновление шаблона **не** обновляет клетки, которые уже были клонированы из него, а обновление клона **не** обновляет шаблон. Поэтому существует два способа поддерживать эти клетки в актуальном состоянии.

Первый вариант — обновить каждый клон на месте с помощью freebsd-update(8), точно так же, как и для классической клетки, как описано в Обновление классической клетки или тонкой клетки с использованием снимков OpenZFS. Это самый простой подход для нескольких клеток, но каждый пропатченный клон отклоняется от шаблонного снимка: изменённые блоки дублируются для каждой клетки, что сокращает экономию места, а крупное обновление переписывает так много пользовательского окружения, что большая часть совместного использования теряется.

Второй вариант сохраняет совместное использование, обновляя шаблон и повторно клонируя его. Обновите шаблон до последнего уровня исправлений на хосте:

```
# freebsd-update -b /usr/local/jails/templates/15.1-RELEASE/ fetch install
```

freebsd-update(8) с параметром `-b` читает текущую версию выпуска из `uname -r` хост-системы, а не из шаблона. Если шаблон работает на более старой версии выпуска, чем хост-система, добавьте `--currently-running <template-release>`, чтобы он применял исправления к версии выпуска шаблона, а не хоста-системы, как показано в Обновление тонкой клетки с использованием NullFS.

Затем создайте новый снимок и клонируйте будущие клетки из него:

```
# zfs snapshot zroot/jails/templates/15.1-RELEASE@base-p2
# zfs clone zroot/jails/templates/15.1-RELEASE@base-p2 zroot/jails/containers/newjail
```

Существующие клоны сохраняют базу, из которой они были созданы, до тех пор, пока они не будут пересозданы из нового снимка. Для обновления основной или дополнительной версии создайте новый шаблонный набор данных для нового выпуска, следуя разделу Создание тонкой клетки с помощью снимков OpenZFS, вместо обновления существующего шаблона на месте, и клонируйте новые клетки из его снимка `@base`.

Храните данные приложений и конфигурацию каждой клетки на отдельных наборах данных или в локальном хранилище клетки, чтобы повторное создание клетки из обновлённого снимка шаблона было дешёвым и предпочтительным по сравнению с отдельным обновлением

каждого клона.

17.12.3. Обновление тонкой клетки с использованием NullFS

Поскольку тонкие клетки, использующие NullFS, разделяют большинство системных каталогов, их очень легко обновлять. Достаточно обновить шаблон. Это позволяет обновлять несколько клеток одновременно.

Для обновления шаблона до последнего патч-релиза версии FreeBSD, на которой он работает, выполните следующие команды на хосте:

```
# freebsd-update -b /usr/local/jails/templates/15.1-RELEASE-base/ fetch install
# service jail restart
```

Для обновления шаблона до новой основной или промежуточной версии сначала обновите хост-систему, как описано в Выполнение обновлений основной и промежуточной версий. После обновления и перезагрузки хост-системы можно обновить шаблон.

Поскольку хост-система была обновлёна первой, freebsd-update(8) с `-b` в противном случае обнаружил бы новый выпуск хост-системы, а не более старый выпуск шаблона: `-b` считывает текущую версию из `uname -r` хост-системы, в отличие от `-j`, который определяет собственную пользовательскую среду клетки. Передайте `--currently-running <release-шаблона>`, чтобы шаблон был обновлён с его фактического выпуска. Следующий пример обновляет шаблон, работающий на 14.2-RELEASE, до 14.3-RELEASE; подставьте применимые выпуски:

```
# freebsd-update -b /usr/local/jails/templates/14.2-RELEASE-base/ --currently-running
14.2-RELEASE -r 14.3-RELEASE upgrade
# freebsd-update -b /usr/local/jails/templates/14.2-RELEASE-base/ install
# service jail restart
# freebsd-update -b /usr/local/jails/templates/14.2-RELEASE-base/ install
# service jail restart
```

17.12.4. Обновление клеток с pkgbase

Клетки, установленные из базовых пакетов, доступных с FreeBSD 15.0, обновляются с помощью pkg(8) через репозиторий `FreeBSD-base`, а не с помощью freebsd-update(8).

Репозиторий `FreeBSD-base` определён в /etc/pkg/FreeBSD.conf, но по умолчанию отключён в FreeBSD 15.0, поэтому включите его перед обновлением.

С хост-системы укажите pkg(8) на корневой каталог клетки с помощью `-r` (`--rootdir`):

```
# pkg -r /usr/local/jails/containers/pkgjail update
# pkg -r /usr/local/jails/containers/pkgjail upgrade
# service jail restart pkgjail
```

Тот же самый апгрейд можно выполнить изнутри клетки с помощью jexec(8) и обычной команды `pkg upgrade`:

```
# jexec -l pkgjail pkg upgrade
# service jail restart pkgjail
```

pkg(8) выполняет трехстороннее слияние размеченных конфигурационных файлов в /etc. Когда слияние не может быть завершено автоматически, он оставляет новую версию как <file>.pkgnew, а когда он забирает файл, который ранее не был упакован, он сохраняет старую копию как <file>.pkgsave. Проверьте /etc после обновления и вручную согласуйте любые такие файлы. Не запускайте etcupdate(8) в системе pkgbase; pkg(8) сам обрабатывает слияние конфигураций. Смотрите Обновление и модернизация FreeBSD для описания рабочего процесса с базовыми пакетами на стороне хоста.

17.13. Ограничения ресурсов клетки

Управление ресурсами хост-системы, которые использует клетка — это задача, которую должен учитывать системный администратор.

Используйте rctl(8) для управления ресурсами хост-системы, которые клетка может использовать.

Настройка `kern.racct.enable` должна быть включена в файле /boot/loader.conf.

Синтаксис для ограничения ресурсов клетки выглядит следующим образом:

```
rctl -a jail:<jailname>:resource:action=amount[/per]
```

Необязательный компонент `per` не является процентным соотношением; это учетная единица, к которой применяется ограничение: одна из `process`, `user`, `loginclass` или `jail`. Например, `deny=2G/jail` ограничивает сумму для всей клетки, тогда как `deny=2G/process` применяет то же значение к каждому процессу по отдельности. Процентное соотношение имеет смысл только для ресурса `pcpu`, количество которого выражается в процентах от одного ядра процессора.

Например, чтобы ограничить максимальный объём оперативной памяти, доступной для клетки, выполните следующую команду:

```
# rctl -a jail:classic:memoryuse:deny=2G
```

Чтобы ограничение сохранялось после перезагрузки хост-системы, необходимо добавить правило в файл /etc/rctl.conf следующим образом:

```
jail:classic:memoryuse:deny=2G/jail
```

rctl(8) может ограничивать гораздо больше, чем память. Ресурсы, наиболее значимые для клетки, включают `pcpu` (процент одного ядра ЦП), `maxproc` (количество процессов), `openfiles` (открытые файловые дескрипторы), `vmemoryuse` (адресное пространство) и `swapuse` (пространство подкачки). Каждый из них настраивается одинаково:

```
# rctl -a jail:classic:pcpu:deny=80
# rctl -a jail:classic:maxproc:deny=1000
# rctl -a jail:classic:openfiles:deny=5000
# rctl -a jail:classic:vmemoryuse:deny=4G
```

Пропускная способность диска ограничивается параметрами `readbps` и `writebps`, но эти ресурсы не поддерживают действие `deny`; вместо этого объединяйте их с `throttle`:

```
# rctl -a jail:classic:writebps:throttle=1m
```

Действие после ресурса определяет, что происходит при достижении лимита: `deny` блокирует операцию, `log` записывает её в системный журнал, а `devctl` отправляет уведомление через devd(8).

Сообщите о текущем использовании ресурсов клетки с помощью флага `-u`, добавив `-h` для единиц, удобных для чтения человеком:

```
# rctl -hu jail:classic
```

Перечислите правила, которые применяются к клетке:

```
# rctl jail:classic
```

Удалите каждое правило, которое соответствует клетке, с помощью `-r`:

```
# rctl -r jail:classic
```

Дополнительная информация об ограничениях ресурсов доступна в главе о безопасности в разделе Ограничения ресурсов.

17.13.1. Привязка CPU с помощью cpuset

rctl(8) с ограничением `pcpu` определяет, сколько времени ЦП может потреблять клетка, но она не привязывает клетку к конкретным ядрам; cpuset(1) вместо этого назначает конкретные ядра.

Ограничить работающую клетку набором CPU по её имени или JID:

```
# cpuset -l 0-3 -j classic
```

Запросить ядра, которые клетка в настоящее время может использовать, с помощью `-g`:

```
# cpuset -g -j classic
```

Назначенный набор также отображается в столбце `cpuset` команды `jls -v`. В файле `jail.conf` нет параметра для привязки к процессору, поэтому закрепите клетку при запуске, выполнив cpuset(1) из хука `exec.created` или `exec.poststart`, оба из которых выполняются в среде хоста:

```
exec.poststart = "cpuset -l 0-3 -j classic";
```

17.13.2. Ограничение дискового пространства

rctl(8) не имеет ресурса для файлового пространства. Когда каждая клетка находится в собственном наборе данных ZFS, ограничьте и гарантируйте использование дискового пространства с помощью квот и резервирований ZFS вместо этого:

```
# zfs set quota=20G zroot/jails/containers/classic
# zfs set reservation=5G zroot/jails/containers/classic
```

Для этого требуется, чтобы клетка была собственным набором данных. Созданные ранее тонкие клетки ZFS уже таковыми являются, но процедура Классической Клетки создает каталог клетки с помощью `mkdir` внутри общего набора данных containers, поэтому сначала создайте его с помощью `zfs create`, чтобы эти команды применялись. См. Настройка дерева каталогов клетки для структуры наборов данных и zfs-set(8) для свойств `quota` и `reservation`.

17.14. Дополнительные параметры клетки

jail(8) предоставляет гораздо больше параметров, чем используется в примерах из этой главы. Клетка является сильной границей изоляции, но не абсолютной. Каждое послабление `allow.*` обменивает изоляцию на возможности и приближает привилегированных пользователей клетки на шаг ближе к привилегиям `root` на хост-системе. Предоставляйте клетке только те разрешения, которые действительно необходимы для выполняемой в ней рабочей нагрузки, а всё остальное оставляйте в более ограничительном режиме по умолчанию. Полный список приведён в jail(8) и jail.conf(5); параметры ниже — это те, к которым администраторы обращаются чаще всего.

17.14.1. Разрешения allow.*

Параметры `allow.*` являются логическими значениями, которые предоставляют отдельные привилегии клетке. Согласно jail(8), "За исключением allow.set_hostname и allow.reserved_ports, эти логические параметры по умолчанию выключены", поэтому клетка

запускается в заблокированном состоянии, и каждая возможность должна быть предоставлена явно.

Таблица 31. Общие параметры `allow.*`

Параметр	По умолчанию	Эффект
`allow.set_hostname`	On	Клетка может изменить собственное имя хоста.
`allow.reserved_ports`	On	`root` в клетке может открывать (делать bind) зарезервированные порты ниже 1024.
`allow.raw_sockets`	Off	Включает «сырые» (raw) сокеты, чтобы ping(8) и traceroute(8) работали внутри клетки.
`allow.chflags`	Off	`root` в клетке может управлять системными файловыми флагами так же, как и на хост-системе.
`allow.mount`	Off	Клетка может монтировать и размонтировать файловые системы, вместе с дочерними опциями `allow.mount.*`.
`allow.quotas`	Off	Клетка может администрировать квоты файловой системы.
`allow.read_msgbuf`	Off	Процессы в клетках могут читать буфер сообщений ядра с помощью dmesg(8).
`allow.socket_af`	Off	Разрешает семейства адресов сокетов, отличные от IPv4, IPv6, локальных и маршрутных.
`allow.mlock`	Off	Процессы в клетках могут блокировать физическую память с помощью mlock(2).
`allow.unprivileged_proc_debug`	Off	Непривилегированные пользователи могут использовать средства отладки для своих собственных процессов.
`allow.vmm`	Off	Клетка может использовать гипервизор vmm(4) для запуска гостевых систем bhyve(8).
`allow.nfsd`	Off	Клетка может запускать NFS-сервер.

Взвесьте каждое послабление относительно того, что оно открывает: `allow.raw_sockets` позволяет процессам в клетке подделывать пакеты, `allow.chflags` позволяет `root` в клетке снимать флаги неизменяемости и только для добавления, на которые полагается хост-система, а `allow.mount` открывает пространство имён файловой системы.

Запуск NFS-сервера внутри клетки, отмеченный в примечаниях к выпуску FreeBSD 14.0, требует установки параметра `allow.nfsd`, VNET-клетки, корневой каталог которой сам

является точкой монтирования файловой системы, а также параметра `enforce_statfs`, не установленного в значение `0`.

17.14.2. IPC System V (межпроцессное взаимодействие)

Базы данных и другое программное обеспечение, которое полагается на разделяемую память System V, семафоры и очереди сообщений, управляются по модулям с помощью параметров `sysvmsg`, `sysvsem` и `sysvshm`. Каждый из них принимает одно из трёх значений: `inherit` делает все объекты IPC узла видимыми для клетки, `new` предоставляет клетке собственное частное пространство имён ключей, а `disable` приводит к сбою этих системных вызовов.

Для классического случая запуска PostgreSQL в клетке, выбор `new` является правильным, предоставляя базе данных собственное пространство имён разделяемой памяти, не раскрывая пространство хост-системы:

```
sysvmsg = new;
sysvsem = new;
sysvshm = new;
```

Устаревший параметр `allow.sysvipc` больше не используется. Он эквивалентен установке всех трех параметров модуля в значение `inherit`, что разделяет общесистемное пространство имен IPC с клеткой, поэтому предпочтительнее использовать указанные выше параметры для каждого модуля.

17.14.3. Монтирование файловых систем

`allow.mount` позволяет клетке монтировать файловые системы, а семейство дочерних параметров выбирает, какие именно типы: `allow.mount.devfs`, `allow.mount.fdescfs`, `allow.mount.fusefs`, `allow.mount.nullfs`, `allow.mount.procfs`, `allow.mount.linprocfs`, `allow.mount.linsysfs`, `allow.mount.tmpfs` и `allow.mount.zfs`. Согласно jail(8), каждый дочерний параметр «действует только вместе с `allow.mount` и только когда `enforce_statfs` установлен в значение ниже 2». Удобные параметры `mount.devfs`, `mount.fdescfs` и `mount.procfs` автоматически монтируют соответствующие файловые системы для клетки, а `mount.fstab` указывает на файл fstab с дополнительными точками монтирования.

Параметр `enforce_statfs` управляет как тем, какие точки монтирования может видеть клетка, так и, согласно приведенному выше правилу, тем, действуют ли разрешения `allow.mount.*`. Он принимает три значения:

Таблица 32. `enforce_statfs` значения

Значение	Точки монтирования, видимые клетке
`0`	Все точки монтирования на хост-системе, без каких-либо ограничений.
`1`	Видны только точки монтирования ниже корневого каталога клетки.
`2` (по умолчанию)	Видна только единственная точка монтирования, где расположен корневой каталог клетки.

Поскольку для прав `allow.mount.*` требуется значение ниже `2`, клетка, которой необходимо монтировать собственные файловые системы, обычно настраивается с параметром `enforce_statfs = 1`.

17.14.4. Уровень безопасности (securelevel) для каждой клетки

Параметр `securelevel` устанавливает значение `kern.securelevel` для клетки и повышает уровень защиты ядра от изменений флагов и устройств внутри клетки. Согласно jail(8), «клетка никогда не имеет более низкого уровня securelevel, чем её родительская система, но, установив этот параметр, она может иметь более высокий». Таким образом, клетка, обращённая к сети, может работать при ужесточённом `securelevel = 3`, в то время как хост-система остаётся на уровне по умолчанию.

17.14.5. Иерархичские клетки (Hierarchical Jails)

Клетка сама может создавать дочерние клетки, именно так строятся вложенные или иерархические структуры клеток. Параметр `children.max` задаёт, сколько дочерних клеток может создать клетка; по умолчанию он равен `0`, что означает отсутствие разрешения на создание дочерних клеток, а доступный только для чтения параметр `children.cur` сообщает, сколько их существует в данный момент. Дочерняя клетка никогда не может превысить привилегии своего родителя: согласно jail(8), «Процессы, находящиеся в клетке, не имеют права предоставлять большие полномочия, чем те, которые даны им самим». Это делает возможным запуск менеджера клеток внутри клетки.

17.14.6. Пустые и постоянные клетки

По умолчанию клетка существует только пока в ней есть запущенные процессы; когда её последний процесс завершается, клетка исчезает. Установка параметра `persist`, по словам jail(8), «позволяет клетке существовать без каких-либо процессов». Именно это поддерживает клетку в живых после её создания пустой и до того, как в неё будет что-либо помещено, а также закрепляет VNET-клетку, которая хранит только сетевое состояние. Клетка может быть создана пустой и поддерживаться в живых для экспериментов:

```
# jail -c name=test path=/rescue persist
```

Сбросьте флаг с помощью `nopersist`, чтобы клетка могла быть удалена после того, как снова станет неактивной.

17.14.7. Представление другого выпуска ОС

Пользовательское окружение клетки может быть старше, чем ядро хост-системы, но инструменты внутри нее все равно читают версию хост-системы через uname(1), если не указано иное. Параметр `osrelease` задает строку, сообщаемую через `kern.osrelease` и `uname -r` клетки, а `osreldate` задает число, сообщаемое через `kern.osreldate` и `uname -K`. Установка их таким образом, чтобы более старое пользовательское окружение сообщало о собственной версии, обеспечивает корректную работу uname(1), потребителей sysctl(8) и инструментов сборки, определяющих версию, включая freebsd-update(8), запускаемый внутри клетки.

Управление пакетами не затрагивается ни в ту, ни в другую сторону, потому что pkg(8) получает свой ABI из ELF-заметок собственных бинарных файлов клетки, а не из uname(1). Это дополняет правило, согласно которому пользовательское окружение клетки никогда не может быть новее, чем ядро хоста; см. Обновление клеток.

17.15. Клетки и ZFS

Клетка может получить контроль над одним или несколькими наборами данных ZFS, позволяя привилегированному пользователю внутри клетки создавать, снимать снимки, клонировать и откатывать дочерние наборы данных без участия администратора хост-системы. Это рекомендуемый способ предоставить клетке собственное записываемое хранилище с поддержкой снимков.

Хост-система сохраняет право собственности на набор данных и делегирует клетке только управление делегированным деревом и его потомками. Свойство `jailed`, подкоманда `zfs jail` и взаимодействие между ними описаны в ZFS и клетки.

17.15.1. Ручное делегирование набора данных

Этот метод работает на всех поддерживаемых выпусках FreeBSD.

На хост-системе создайте набор данных для делегирования и пометьте его как клетку:

```
# zfs create zroot/jails/data
# zfs set jailed=on zroot/jails/data
```

Установка свойства `jailed` сообщает ZFS, что набор данных будет управляться изнутри клетки, поэтому хост-система автоматически перестаёт его монтировать.

Далее, разрешите клетке монтировать файловые системы ZFS и присоединять набор данных при создании клетки. Добавьте следующее в раздел клетки в /etc/jail.conf:

```
allow.mount;
allow.mount.zfs;
enforce_statfs = 1;
exec.created += "zfs jail ${name} zroot/jails/data";
```

Права `allow.mount.zfs` действуют только вместе с `allow.mount` и значением `enforce_statfs` ниже 2; см. More Jail Options для прав на монтирование и их взаимодействия с `enforce_statfs`. Поскольку делегированный набор данных отсоединяется при остановке клетки, хук `exec.created` повторно присоединяет его при каждом запуске, где `${name}` расширяется до имени клетки. То же присоединение можно выполнить к уже запущенной клетке с помощью команды `zfs jail <jid|name> zroot/jails/data`.

Клетке требуется доступ к /dev/zfs. Стандартный набор правил devfs для клеток (`devfsrules_jail`, набор правил 4) уже открывает его, поэтому для

делегирования ZFS не требуется пользовательский набор правил devfs.

Внутри клетки смонтируйте делегированные наборы данных и управляйте ими как любыми другими:

```
# zfs mount -a
# zfs create zroot/jails/data/work
# zfs snapshot zroot/jails/data/work@monday
```

root-пользователь в клетке может создавать, делать снимки, клонировать и откатывать наборы данных в делегированном дереве, но свойство `quota` делегированного набора данных может быть установлено только с хоста-системы.

17.15.2. Автоматическое подключение наборов данных

Параметр `zfs.dataset` требует FreeBSD 15.0. Он отсутствует в jail(8) для FreeBSD 14.3.

Начиная с FreeBSD 15.0, jail(8) может самостоятельно подключать наборы данных, устраняя необходимость в ручных шагах `jailed` и `zfs jail`. Перечислите наборы данных для делегирования в параметре `zfs.dataset`:

```
allow.mount;
allow.mount.zfs;
enforce_statfs = 1;
zfs.dataset = "zroot/jails/data";
```

jail(8) присоединяет каждый перечисленный набор данных при запуске клетки, поэтому показанный выше хук `exec.created` больше не требуется. Этот параметр требует, чтобы было установлено значение `allow.mount.zfs`.

Для неустаревших менеджеров клеток, которые еще не понимают `zfs.dataset`, стартовый сценарий rc.d/jail также учитывает устаревшую переменную `jail_<jailname>_zfs_dataset` в /etc/rc.conf.

17.15.3. Доступ к снимкам изнутри клетки

Параметр `zfs.mount_snapshot`, доступный в FreeBSD 14.x и 15.x, позволяет пользователям внутри клетки получать доступ к содержимому снимков без делегирования всего набора данных. Когда он установлен в значение 1, пользователи клетки могут читать содержимое снимков файловой системы в её каталоге .zfs/snapshot:

```
zfs.mount_snapshot = 1;
```

Это удобно для восстановления отдельных файлов из снимков, сделанных на хост-системе. Если также установлено `allow.mount.zfs`, снимки могут быть дополнительно смонтированы

внутри клетки. Свойство `snapdir` набора данных управляет тем, будет ли виден каталог .zfs.

Делегированный набор данных в сочетании с `allow.mount.zfs` позволяет root-пользователю внутри клетки устанавливать связанные с монтированием свойства ZFS и монтировать файловые системы внутри клетки. Предоставляйте эти разрешения обдуманно и только тем клеткам, которым доверено управлять собственным хранилищем.

17.16. Менеджеры клеток и контейнеры

Как уже объяснялось ранее, каждый тип клетки FreeBSD может быть создан и настроен вручную, но в FreeBSD также существуют сторонние утилиты для упрощения настройки и администрирования.

Ниже приведён неполный список различных менеджеров клеток FreeBSD:

Таблица 33. Менеджеры клеток

Имя	Лицензия	Пакет	Documentation
BastilleBSD	BSD-3	sysutils/bastille	Документация
pot	BSD-3	sysutils/pot	Документация
cbsd	BSD-2	sysutils/cbsd	Документация
AppJail	BSD-3	sysutils/appjail, для разработки sysutils/appjail-devel	Документация
iocage	BSD-2	sysutils/iocage	Документация
ezjail	Пивная Лицензия	sysutils/ezjail	Документация

FreeBSD публикует официальные OCI-образы базовой системы, начиная с FreeBSD 14.3-RELEASE, а OCI-контейнеры (стиль Docker), которые построены на той же технологии клеток и управляются с помощью инструментария Podman, рассматриваются в главе Контейнеры, которая дополняет обычные клетки, описанные здесь.

Глава 18. OCI Containers

18.1. Synopsis

The Open Container Initiative, commonly referred to as `OCI`, provides a vendor and OS-agnostic way to describe, distribute, and run containers. The OCI specifications provide these in a way that can be used on many different operating systems, including FreeBSD. The underlying virtualization technology is still FreeBSD jails, with the same feature set, but OCI tooling enables additional ways of working, and constructing, container-based workloads. 14.3-RELEASE and upwards, including snapshots, now include OCI-compatible images, and the Podman toolkit on FreeBSD is ready to use them, on both amd64 and arm64 architectures.

For FreeBSD users familiar with jails, there is a loose analogy:

- FreeBSD's `base.txz` tarball is an example of a container image.
- the `jail.conf` file describes the desired container properties, or `Containerfile`.
- use the `jail ..` command to run a container, given a filesystem path, with the `podman` suite of tools.

By importing this container stack, FreeBSD users both benefit from common tooling, but also enjoy wide support across public and private container registries, and container-specific tooling and services. In the aarch64 and amd64 download directories, you'll see official OCI-format images. The naming may be a little confusing at first, but should make sense once you start using them.

The same images are also available through common public container registries, including Docker Hub, and Github Container Registry, but for the strongest chain of trust, you should fetch your image directly from Official FreeBSD Releases, and import them to your local system.

Each image comprises a subset of a standard FreeBSD `base.txz` release tarball, for various use cases, and the usual FreeBSD `CHECKSUM.*` files, that can be verified against the PGP-signed release announcement. This gives a very strong chain of provenance, directly verifiable from the FreeBSD release team.

```
CHECKSUM.SHA256
CHECKSUM.SHA512
FreeBSD-15.1-RELEASE-amd64-container-image-static.txz
FreeBSD-15.1-RELEASE-amd64-container-image-dynamic.txz
FreeBSD-15.1-RELEASE-amd64-container-image-runtime.txz
FreeBSD-15.1-RELEASE-amd64-container-image-notoolchain.txz
FreeBSD-15.1-RELEASE-amd64-container-image-toolchain.txz
```

18.2. Container Image Naming and Tagging

18.2.1. Introduction

FreeBSD's official container images are built from base system packages, themselves built during

the FreeBSD release process, and published to public registries. This guide explains how images are named and tagged, helping you choose the right image for your needs, and ensuring you understand the implications of each choice, as new images are published, and in some cases, mutable tags are amended. This naming convention aligns with the new base system package naming scheme, giving clear provenance between container images and the corresponding FreeBSD releases.

18.2.2. Available Image Types

Five types of FreeBSD container images are available, for `amd64` and `aarch64` architectures.

`freebsd-static`

The static image is intended as a base image, for a workload which is entirely statically linked. It contains no libraries, nor binaries, just the supporting files that most applications of this nature require.

- public TLS certificates
- minimal password file
- minimal termcap
- timezone database

Note that it has, by design, a lean footprint, to make a security compromise of the container less useful to the attacker. There's no UNIX shell, no command-line tools, no dynamic libraries, nor package manager. It is the smallest image provided by the FreeBSD project.

`freebsd-dynamic`

The dynamic image uses the static image as a parent layer, and supports using shared libraries, including `libc`. Most FreeBSD software should run without issue with this image, with minor changes. It doesn't have a shell, rc system, nor a package manager. These limitations are additional security, making it awkward for attackers in a compromised container to move laterally, or make use of tools that were never installed.

`freebsd-runtime`

Again, runtime builds on the preceding dynamic layer, and finally adds the minimum that a user would expect - a UNIX shell, rc system, and the package manager. It is the ideal base image for porting existing applications with a minimum of changes. Users will need to include additional FreeBSD base system libraries, as well as additional packages from the Ports tree. This is the closest to a typical FreeBSD system, including the pkg(8) tool, allowing users to install, or do, almost anything, similar to a non-containerised system.

`freebsd-notoolchain`

This base image contains almost all tools one would expect on a typical FreeBSD system, excluding those that are directly hardware-related, and thus not generally useful within a container, and the compiler and related toolchain, as it is quite large.

`freebsd-toolchain`

The Toolchain base image is the sum of all preceding images, including a full compiler and toolchain. It is generally possible to compile almost any software for FreeBSD in the same way as a normal non-jailed FreeBSD system. All images follow a consistent naming pattern, derived from the FreeBSD release they are based upon.

18.2.3. Image Tag Structure

Immutable Tags

Immutable tags never change and are ideal for production systems where you need stability and predictability. These are the most common tags used by the release process. They will not change, even after FreeBSD security patches, or errata notices. They are ideal for base images for software deployments where a high degree of reproducibility is expected, even as a trade-off against more pro-active security patching.

- `major.minor` (e.g., `14.4`): Points to a specific FreeBSD RELEASE version
- `major.minor.pX` (e.g., `14.4p1`): Points to a specific patch level of a RELEASE
- `major.snapYYYYMMDDHHMMSS` (e.g., `16.snap20260325035941`): a snapshot build with timestamp

Mutable Tags

Mutable tags are updated over time and are ideal for production, or CI/CD systems that always want the latest updates, but still wish to align with FreeBSD ABI compatibility.

- `major` (e.g., `15`) will always be the latest support RELEASE version along the 15.x stable ABI
- `major.snap` (e.g., `15.snap`) follows the latest developer snapshot along the 15.x stable ABI train, will often be in advance of official releases, and carries a risk of breakage, incompatibility, or other unanticipated issues. This is the most mutable tag, and should be used with caution. They are exactly the same as using FreeBSD's STABLE or CURRENT snapshots, as they use those images as their source.

18.2.4. When to Use Each Tag Type

For Production

Use immutable tags to ensure your environment stays consistent:

- `freebsd-runtime:15.0` - pinned to FreeBSD 15.0-RELEASE

When security patches are released, you must explicitly rebuild or update with a new image tag.

For Continuous Integration and Development

Use mutable tags to automatically get the latest updates:

- `freebsd-runtime:15` - always the latest official RELEASE in the 15.x series

These additional mutable tags will follow the regular published snapshots of FreeBSD STABLE and

CURRENT. Use these if you want to test against what subsequent future releases of FreeBSD may contain, and accept occasional breakage or unanticipated changes.

- `freebsd-runtime:14.snap` - always the latest build from 14-STABLE branch
- `freebsd-runtime:15.snap` - always the latest build from 15-STABLE branch
- `freebsd-runtime:16.snap` - always the latest build from 16-CURRENT branch

18.3. Official Registry Locations

Architecture-independent images can be downloaded and directly imported into your own registry from the Official FreeBSD Releases site, or alternatively, pulled from the FreeBSD project's Docker Hub, or GitHub Container Registry.

Both Alpine Linux and the Debian project publish tagged and mutable images, using similar approaches.

18.4. Introduction to Podman

The Podman suite comprises three main tools and a number of supporting ones:

- podman is a daemonless container engine using docker-compatible commands
- buildah generates OCI formatted container images
- skopeo works with remote OCI image registries

18.5. Preparing the System for Podman Containers

If running commands over SSH, or in a similar remote session, use screen(1), tmux(1), or similar to avoid losing the connection while restarting the firewall.

If ZFS is not available, switch to UFS storage and skip the ZFS commands below:

```
# sed -I .bak -e 's/driver = "zfs"/driver = "vfs"/' \
    /usr/local/etc/containers/storage.conf
```

Otherwise, create the container storage dataset:

```
# zfs create -o mountpoint=/var/db/containers zroot/containers
# zfs snapshot zroot/containers@empty
```

The File Descriptor filesystem is required:

```
# mount -t fdescfs fdesc /dev/fd
```

Podman uses FreeBSD's packet filter to forward container ports to the host network:

```
# test -c /dev/pf || kldload pf
# sysctl net.pf.filter_local=1
```

Amend `/etc/sysctl.conf` and `/etc/fstab` as appropriate, to make these changes permanent.

18.6. Installing Podman

Only the `sysutils/podman-suite` meta-package is required, but if the additional emulators/qemu-user-static package is installed, it becomes possible to build or test images for other architectures such as `arm64` on an `amd64` host.

```
# pkg install -r FreeBSD -y podman-suite emulators/qemu-user-static
```

Integrate changes from `/usr/local/etc/containers/pf.conf.sample` into `/etc/pf.conf`, setting egress macros appropriately, then restart the firewall:

```
# service pf restart
```

The packages install a number of template configuration files, none of which need to be edited immediately. Review and amend these as needed:

```
# pkg list buildah podman conmon \
    ocijail containers-common \
    containernetworking-plugins \
    | grep /etc/
/usr/local/etc/containers/containers.conf.sample
/usr/local/etc/containers/policy.json.sample
/usr/local/etc/containers/registries.conf.sample
/usr/local/etc/containers/storage.conf.sample
/usr/local/etc/containers/pf.conf.sample
```

18.7. Importing and Running Containers

At present, all Podman containers on FreeBSD must run as root, as FreeBSD jails require this.

With the necessary tools and firewall rules in place, the officially published images can be fetched from FreeBSD Releases, for aarch64 or amd64 as appropriate.

18.7.1. Importing FreeBSD OCI Images

It is simplest to pull images directly from a public container registry, but for the strongest chain of

trust, download them from FreeBSD.org directly, and verify the checksums against the PGP-signed release announcement.

```
# export OCIBASE=https://download.freebsd.org/releases/OCI-IMAGES/15.1-
RELEASE/amd64/Latest
# podman load -i=$OCIBASE/FreeBSD-15.1-RELEASE-amd64-container-image-static.txz
Getting image source signatures
Copying blob 44defc6cee2e done   |
Copying config 5921c62e76 done   |
Writing manifest to image destination
Loaded image: localhost/freebsd-static:15.1-RELEASE-amd64

# podman load -i=$OCIBASE/FreeBSD-15.1-RELEASE-amd64-container-image-dynamic.txz
Getting image source signatures
Copying blob 6054dffc3887 done   |
Copying config 33427d59f9 done   |
Writing manifest to image destination
Loaded image: localhost/freebsd-dynamic:15.1-RELEASE-amd64

# podman load -i=$OCIBASE/FreeBSD-15.1-RELEASE-amd64-container-image-runtime.txz
Getting image source signatures
Copying blob dcbbfcb598fe done   |
Copying config 90c4936754 done   |
Writing manifest to image destination
Loaded image: localhost/freebsd-runtime:15.1-RELEASE-amd64

# podman load -i=$OCIBASE/FreeBSD-15.1-RELEASE-amd64-container-image-notoolchain.txz
Getting image source signatures
Copying blob a16c94176cc3 done   |
Copying config 086572f35c done   |
Writing manifest to image destination
Loaded image: localhost/freebsd-notoolchain:15.1-RELEASE-amd64

# podman load -i=$OCIBASE/FreeBSD-15.1-RELEASE-amd64-container-image-toolchain.txz
Getting image source signatures
Copying blob b04531b7d93d done   |
Copying config b927c11b2e done   |
Writing manifest to image destination
Loaded image: localhost/freebsd-toolchain:15.1-RELEASE-amd64
```

18.7.2. Listing Images

```
# podman images
REPOSITORY                    TAG                   IMAGE ID      CREATED       SIZE
localhost/freebsd-toolchain   15.1-RELEASE-amd64  b927c11b2e8d  13 days ago   660 MB
localhost/freebsd-notoolchain 15.1-RELEASE-amd64  086572f35cc9  13 days ago   152 MB
localhost/freebsd-runtime     15.1-RELEASE-amd64  90c493675429  13 days ago   34.1 MB
localhost/freebsd-dynamic     15.1-RELEASE-amd64  33427d59f990  13 days ago   14.5 MB
```

```
localhost/freebsd-static      15.1-RELEASE-amd64  5921c62e76c1  13 days ago   2.92 MB
```

Note that the image IDs are reproducible each time, and the `IMAGE_ID` column matches the hashes reported when importing the images.

18.7.3. Listing Layers

It is possible to show the layers that comprise an image, demonstrating how the `runtime` image has three components, each one a layer in its own right:

```
# podman image tree ghcr.io/freebsd/freebsd-runtime:14.2
Image ID: c5f3e77557a9
Tags:     [ghcr.io/freebsd/freebsd-runtime:14.2]
Size:     35.07MB
Image Layers
├──── ID: cd53fb07fb66 Size: 5.449MB Top Layer of: [ghcr.io/freebsd/freebsd-
static:14.2]
├──── ID: a01d37f7777b Size:  10.4MB Top Layer of: [ghcr.io/freebsd/freebsd-
dynamic:14.2]
└──── ID: 36b0c80ca1f7 Size: 19.21MB Top Layer of: [ghcr.io/freebsd/freebsd-
runtime:14.2]
```

18.7.4. Using Public Registries

While the most secure provenance is downloading from Official FreeBSD Releases, there are two public container registries managed by the FreeBSD Release and Cluster Admin teams. The images from both registries are identical, but provide a simpler and more container-friendly workflow.

- Docker Hub
- GitHub Container Registry

18.7.5. Fetching Containers

Images can be fetched from a public registry using `podman pull`:

```
# podman pull ghcr.io/freebsd/freebsd-notoolchain:15.1
Trying to pull ghcr.io/freebsd/freebsd-notoolchain:15.1...
Getting image source signatures
Copying blob 711cf0d3c0ef done   |
Copying config 086572f35c done   |
Writing manifest to image destination
086572f35cc90b1a1d2b7904b63e8a849071b4ba89b15492f6da47a802372aa2
```

18.7.6. Running a Container

Use `podman run` to start a container. The `-it` flags allocate an interactive terminal, and `--rm` removes

the container when it exits. If no command is specified, the image's default entrypoint is used:

```
# podman run -it --rm ghcr.io/freebsd/freebsd-notoolchain:15.1
# exit
```

A specific command can be given as well:

```
# podman run -it --rm ghcr.io/freebsd/freebsd-notoolchain:15.1 uname -a
FreeBSD c0038de70651 15.1-RELEASE FreeBSD 15.1-RELEASE releng/15.1-n283562-
96841ea08dcf GENERIC amd64
```

As the container image does not include the kernel, freebsd-version(1) shows different results for the running kernel versus the installed userland. In this example, a 15.0-RELEASE container runs on a 15.1-RELEASE host:

```
# podman run -it --rm ghcr.io/freebsd/freebsd-notoolchain:15.0 /bin/sh
# freebsd-version -ru
15.1-RELEASE
15.0-RELEASE
# exit
```

18.8. Useful Variations and Tips

Almost all of the `podman run ...` parameters also apply to `podman build`. Refer to the podman-run(1) and podman-build(1) manual pages for details.

Use the `--rm` flag to have ephemeral containers clean themselves up afterwards.

Use `podman images -a` to show all downloaded images:

```
# podman images -a
REPOSITORY                        TAG            IMAGE ID      CREATED      SIZE
ghcr.io/freebsd/freebsd-runtime  15.1   90c493675429  2 weeks ago  34.1 MB
```

Use `podman ps -a` to see running containers:

```
# podman ps -a
CONTAINER ID  IMAGE                                          COMMAND  CREATED
STATUS        NAMES
3123623bef9b  ghcr.io/freebsd/freebsd-runtime:15.1   /bin/sh  4 minutes ago  Up 4
minutes  admiring_brattain
```

18.8.1. Environment Variables

Podman sets a number of default environment variables in each container:

```
# podman run -it --rm ghcr.io/freebsd/freebsd-notoolchain:15.1 /usr/bin/env
container=podman
TERM=xterm
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOME=/root
HOSTNAME=60bae3343e6b
```

The `HOSTNAME` is set to the container ID by default. It can be overridden with `--hostname`, which is useful when the application inside the container needs a stable or meaningful hostname.

Arbitrary environment variables can be passed in with `--env`:

```
# podman run --env FOO=BAR -it --rm ghcr.io/freebsd/freebsd-runtime:15.1
# echo $FOO
BAR
```

18.8.2. Mounting Files and Volumes

Both files and volumes can be passed into containers using `--volume`. For example, the `runtime` image does not contain which(1):

```
# podman run -it --rm ghcr.io/freebsd/freebsd-runtime:15.1
# which
/bin/sh: which: not found
```

It can be mounted into the container from the host:

```
# podman run -it --rm --volume /usr/bin/which:/usr/bin/which \
    ghcr.io/freebsd/freebsd-runtime:15.1
# which which
/usr/bin/which
```

For a lean image such as `freebsd-static`, mount the existing pkg-static(8) from the host to bootstrap pkg(8) inside the container. Sharing the host's package cache via `--volume /var/cache/pkg` avoids re-downloading packages that have already been fetched, which is particularly useful when building multiple containers. The `--no-hosts` flag prevents mounting `/etc/hosts`, which would otherwise interfere with package upgrades.

Sharing the package cache only works when the host and container run the same FreeBSD major version, as packages are version-specific.

```
# podman run -it --rm \
    --env ASSUME_ALWAYS_YES=true \
    --no-hosts \
    --volume /usr/local/sbin/pkg-static:/bin/pkg-static \
    --volume /var/cache/pkg \
    ghcr.io/freebsd/freebsd-static:15.1
# pkg-static bootstrap -r FreeBSD
Bootstrapping pkg from pkg+https://pkg.FreeBSD.org/FreeBSD:14:amd64/latest, please
wait...
Verifying signature with trusted certificate pkg.freebsd.org.2013102301... done
[fd343e182fa5] Installing pkg-2.1.2...
[fd343e182fa5] Extracting pkg-2.1.2: 100%
```

Files and directories from the host can also be mounted into containers:

```
# podman run -it --rm \
    --volume /etc/fstab:/etc/fstab \
    --volume /tmp:/var/tmp \
    --volume /tmp \
    ghcr.io/freebsd/freebsd-runtime:15.1
```

18.8.3. Inspecting Image Contents

To list all files in any image, including minimal images with no shell or tools, export the container filesystem and list its contents:

```
# podman export $(podman create --rm ghcr.io/freebsd/freebsd-static:15.1 /) | tar tf -
```

The container images do not include pkg(8) itself, but pkg-static(8) can be mounted from the host to query the installed base system packages. Use `pkg info -q` to list package names, or `pkg info -ql` to show all files owned by packages:

```
# podman run --rm \
    -v /usr/local/sbin/pkg-static:/pkg \
    ghcr.io/freebsd/freebsd-runtime:15.1 /pkg info -q
FreeBSD-audit-lib-15.1
FreeBSD-bzip2-lib-15.1
FreeBSD-certctl-15.1
FreeBSD-clibs-15.1
FreeBSD-fetch-15.1
FreeBSD-kerberos-lib-15.1
FreeBSD-libarchive-15.1
FreeBSD-libcasper-15.1
FreeBSD-libexecinfo-15.1
FreeBSD-libucl-15.1
FreeBSD-mtree-15.1
```

```
FreeBSD-ncurses-lib-15.1
FreeBSD-openssl-15.1
FreeBSD-openssl-lib-15.1
FreeBSD-pam-lib-15.1
FreeBSD-pkg-bootstrap-15.1
FreeBSD-rc-15.1
FreeBSD-runtime-15.1
FreeBSD-xz-lib-15.1
FreeBSD-zlib-15.1
FreeBSD-zoneinfo-15.1
FreeBSD-zstd-lib-15.1
```

When the host and container run different FreeBSD major versions, pkg(8) detects the ABI by inspecting `/bin/sh` inside the container, and will warn about the mismatch:

```
# podman run -it --rm \
    --env ASSUME_ALWAYS_YES=true \
    --no-hosts \
    --volume /usr/local/sbin/pkg-static:/usr/sbin/pkg \
    --volume /var/cache/pkg \
    ghcr.io/freebsd/freebsd-runtime:14.4 /bin/sh
# pkg bootstrap
pkg: Warning: Major OS version upgrade detected.  Running "pkg bootstrap -f"
recommended
```

For images without `/bin/sh`, such as `freebsd-dynamic`, pkg-static(8) cannot detect the system ABI at all.

18.9. Networking and Name Resolution

By default, the Podman suite of tools will use mount_nullfs(8) to mount `/etc/resolv.conf` and a modified `/etc/hosts` from the jail host, as well as a `.containerenv` file in `/var/run`.

This hosts file will conflict with base system package upgrades, so it is advised to skip this during `podman build` runs, but leave it enabled during production deployment:

```
# podman run --dns 1.2.3.4 ...
# podman build --no-hosts ...
```

The `--dns` flag specifies a custom DNS server instead of mounting `/etc/resolv.conf` from the host. The `--no-hosts` flag prevents mounting `/etc/hosts` from the host.

18.10. Building Custom Images

Custom images can be built directly from tarballs (for example those from poudriere(8), or official release tarballs), from existing tagged images, or bootstrapped from pkg(8).

18.10.1. Building from Tarballs

A custom image can be made from any FreeBSD root filesystem tarball using podman-import(1). The tarball can be an official release `base.txz`, one produced by poudriere-image(8), or your own custom tarball. The final tagged image can be pushed to registries, modified, and used for further image creation.

```
# podman import --os freebsd \
    --arch amd64 \
    --message 'Import FreeBSD 15.1-RELEASE base.txz' \
    https://download.freebsd.org/releases/amd64/15.1-RELEASE/base.txz
Downloading from "https://download.freebsd.org/releases/amd64/15.1-RELEASE/base.txz"
Getting image source signatures
Copying blob 3768988b151c done   |
Copying config 332bec83a4 done   |
Writing manifest to image destination
sha256:332bec83a428efa6c9f7d1f8b3fbd589c5613d938dca7da5125240c89f4f3dcd

# podman image tag 332bec83a428 localhost/freebsd-base:15.1

# podman image ls
REPOSITORY              TAG   IMAGE ID      CREATED        SIZE
localhost/freebsd-base  15.1  332bec83a428  5 minutes ago  715 MB
```

18.10.2. Building from Containerfiles

Images can be built using a `Containerfile`, similar to a Dockerfile. In most cases, the `freebsd-notoolchain` image is the most convenient base, as it includes nearly all standard system tools. For leaner containers, start from `freebsd-static`, `freebsd-dynamic`, or `freebsd-runtime` as appropriate, and add only what is needed:

```
# Containerfile
FROM ghcr.io/freebsd/freebsd-notoolchain:15.1
RUN env ASSUME_ALWAYS_YES=yes IGNORE_OS_VERSION=yes pkg bootstrap -r FreeBSD
RUN pkg update
RUN pkg upgrade -y
RUN pkg clean -y
ENTRYPOINT /bin/sh
```

Build the image with podman-build(1):

```
# podman build --no-hosts --squash \
    -t localhost/freebsd-base:15.1 \
    -f ./Containerfile
```

Further images can be layered on top:

```
# Containerfile.webserver
FROM localhost/freebsd-base:15.1
RUN pkg install -y lang/python3
RUN pkg clean -y
ENTRYPOINT /usr/local/bin/python3 -m http.server
```

```
# podman build --no-hosts \
    -t localhost/python-www:15.1 \
    -f ./Containerfile.webserver
```

Run the web server in the background with `--detach`, and use `-p` to map a host port to the container port. In this example, the container listens on its default port 8000, which is mapped to port 8888 on the host:

```
# c=$(podman run --detach --rm -p 8888:8000 localhost/python-www:15.1)
```

The port mapping uses pf(4) redirect rules. These rules redirect traffic arriving over the network, so the mapped port must be accessed from another host, or via the host's network-facing IP address:

```
$ fetch -v http://172.16.2.10:8888/
resolving server address: 172.16.2.10:8888
requesting http://172.16.2.10:8888/
remote size / mtime: 784 / 0
fetch.out                                             784  B 3791 kBps    00s
```

To access the container directly from the host, use the container's own IP on its listening port:

```
# podman inspect -f '{{.NetworkSettings.IPAddress}}' $c
10.88.0.91
# fetch -v http://10.88.0.91:8000/
resolving server address: 10.88.0.91:8000
requesting http://10.88.0.91:8000/
remote size / mtime: 784 / 0
fetch.out                                             784  B   25 MBps    00s
```

Port mappings via `-p` are handled by pf(4) redirect rules and only apply to traffic arriving over the network. Traffic originating from the jail host itself to `localhost` or `127.0.0.1` will not be redirected. From the host, use the container's IP address directly instead.

18.11. Advanced Usage

18.11.1. Daemonising Podman

FreeBSD's Podman implementation provides two rc(8) services:

- The `podman` service runs at boot and ensures that all containers marked with `restart-policy=always` are started.
- The `podman_service` service runs the Podman API service, providing a REST API and listening on `/var/run/podman/podman.sock` by default.

Enable them as follows:

```
# sysrc podman_enable=YES
# sysrc podman_service_enable=YES
# sysrc podman_service_flags='--time 0'
# service podman start
# service podman_service start
```

Containers that should restart automatically after a host reboot must be run with the `--restart=always` option:

```
# podman run --detach --restart=always my-image
```

18.11.2. Running Linux Containers

It is possible to run many Linux container images using FreeBSD's Linux emulation:

```
# service linux onestart
# podman run --rm --os=linux docker.io/alpine cat /etc/os-release | head -1
Trying to pull docker.io/library/alpine:latest...
Getting image source signatures
Copying blob 4abcf2066143 done   |
Copying config 05455a0888 done   |
Writing manifest to image destination
NAME="Alpine Linux"
```

Linux containers that depend on systemd, expect to be PID 1, or rely on Linux-specific kernel interfaces may not work correctly under FreeBSD's Linux emulation.

18.11.3. Running a Local Registry

A registry is a web server for storing and sharing OCI images. The sysutils/docker-registry port provides a simple registry suitable for local development or a small organisation. Zot Registry is a more comprehensive modern alternative.

```
# pkg install -r FreeBSD sysutils/docker-registry
# zfs create zroot/var/db/registry
# chown -R www:www /var/db/registry
# sysrc docker_registry_enable=YES
# sysrc docker_registry_user=www
# sed -i '' -E -e 's,rootdirectory:.+,rootdirectory: /var/db/registry,' \
    /usr/local/etc/docker-registry/config.yml
# touch /usr/local/etc/docker-registry/htpasswd
# chown -R root:www /usr/local/etc/docker-registry/config.yml \
    /usr/local/etc/docker-registry/htpasswd
# chmod 0640 /usr/local/etc/docker-registry/config.yml \
    /usr/local/etc/docker-registry/htpasswd
```

A sample configuration for `/usr/local/etc/docker-registry/config.yml`:

```
version: 0.1
log:
  fields:
    service: registry
storage:
  cache:
    blobdescriptor: inmemory
  filesystem:
    rootdirectory: /var/db/registry
http:
  addr: :5000
  secret: <random-secret>
  headers:
    X-Content-Type-Options: [nosniff]
auth:
  htpasswd:
    realm: basic-realm
    path: /usr/local/etc/docker-registry/htpasswd
health:
  storagedriver:
    enabled: true
    interval: 10s
    threshold: 3
```

Start the registry and verify it is running:

```
# service registry start
# curl -u admin:passwd http://localhost:5000/v2/
```

It is recommended to serve the registry over TLS, behind a reverse proxy such as nginx or haproxy.

18.11.4. Synchronising Registries

It is possible to synchronise official FreeBSD images into a local registry using `skopeo`. The following example replicates all FreeBSD-related images from Docker Hub to a local registry:

```
# sync.yml
docker.io:
  images:
    freebsd/freebsd-static: []
    freebsd/freebsd-dynamic: []
    freebsd/freebsd-runtime: []
```

```
$ skopeo sync --src yaml --dest docker --all sync.yml cr.example.net/freebsd
INFO[0000] Processing repo                               repo=freebsd/freebsd-static
INFO[0000] Querying registry for image tags              repo=freebsd/freebsd-static
...
INFO[0771] Synced 42 images from 3 sources
```

So long as the credentials in `/root/.config/containers/auth.json` are up to date, root access is not required to sync images.

18.11.5. Working with Container Registries

Registries generally require authentication for push operations, and optionally for pull. Credentials are saved in `/root/.config/containers/auth.json` by default after login:

```
# podman login ghcr.io -u username
Password:
Login Succeeded!
```

Images can then be pushed to a registry:

```
# podman push localhost/freebsd-runtime:15.1 \
    docker://ghcr.io/example/freebsd-runtime:15.1
Getting image source signatures
Copying blob bbacfdfbe140 done   |
Copying config 786f2592a8 done   |
Writing manifest to image destination
```

Public images can be pulled without authentication:

```
# podman pull ghcr.io/freebsd/freebsd-runtime:15.1
```

18.12. Building Container Images from Sources

The FreeBSD release tooling provides an `oci-release` target to build the necessary images for `podman load` to use. It requires building world, kernel, and base system packages first, so this can take a while.

The following example builds `arm64` images:

```
# cd /usr/src \
    && make TARGET_ARCH=aarch64 TARGET=arm64 -s -j32 buildworld \
    && make TARGET_ARCH=aarch64 TARGET=arm64 KERNCONF=GENERIC -s -j32 buildkernel \
    && make TARGET_ARCH=aarch64 TARGET=arm64 KERNCONF=GENERIC -s -j32 packages \
    && cd ./release \
    && make TARGET_ARCH=aarch64 TARGET=arm64 -j32 -s clean \
    && make KERNCONF=GENERIC TARGET_ARCH=aarch64 TARGET=arm64 WITH_OCIIMAGES=yes \
      -s oci-release
```

And similarly for `amd64`:

```
# cd /usr/src \
    && make TARGET_ARCH=amd64 TARGET=amd64 -s -j32 buildworld \
    && make TARGET_ARCH=amd64 TARGET=amd64 KERNCONF=GENERIC -s -j32 buildkernel \
    && make TARGET_ARCH=amd64 TARGET=amd64 KERNCONF=GENERIC -s -j32 packages \
    && cd ./release \
    && make TARGET_ARCH=amd64 TARGET=amd64 -s -j32 clean \
    && make KERNCONF=GENERIC TARGET_ARCH=amd64 TARGET=amd64 \
      WITH_OCIIMAGES=yes -s oci-release
```

18.13. Cross-Architecture Testing

The `emulators/qemu-user-static` package allows testing container images built for other architectures. After installing the package, enable and start the service to register image activators for all emulated architectures:

```
# pkg install emulators/qemu-user-static
# service qemu_user_static enable
# service qemu_user_static start
```

Then run an `arm64` image on an `amd64` host by mounting the appropriate QEMU binary:

```
# podman run -it --rm --arch=arm64 \
    -v /usr/local/bin/qemu-aarch64-static:/usr/local/bin/qemu-aarch64-static \
    ghcr.io/freebsd/freebsd-runtime:14.snap uname -a
FreeBSD 73d4049d10b2 14.2-RELEASE-p1 FreeBSD 14.2-RELEASE-p1 GENERIC arm64
```

18.14. Troubleshooting

18.14.1. Package Upgrades and Mounted Files

Podman mounts `/etc/resolv.conf` and `/etc/hosts` into containers using mount_nullfs(8). These mounts can cause errors during pkg(8) operations with messages such as `Cross-device link`:

```
pkg: Fail to rename /etc/.pkgtemp.hosts.VeIDrHM0sKN5 -> /etc/hosts:Cross-device link
```

To avoid this, use `--no-hosts` and `--dns` flags:

```
# podman run --no-hosts --dns 1.1.1.1 -it --rm \
    ghcr.io/freebsd/freebsd-runtime:15.1 /bin/sh
```

18.14.2. Checking Networking

Start with the firewall rules from `/usr/local/etc/containers/pf.conf.sample`. Test connectivity step by step from within a container:

```
# podman run --no-hosts --dns 1.1.1.1 \
    --env ASSUME_ALWAYS_YES=1 \
    -it --rm \
    ghcr.io/freebsd/freebsd-runtime:15.1 \
    /bin/sh

# route show 1.1.1.1
   route to: 1.1.1.1
destination: default
    gateway: 10.88.0.1
  interface: eth0

# ping -c 3 10.88.0.1
PING 10.88.0.1 (10.88.0.1): 56 data bytes
64 bytes from 10.88.0.1: icmp_seq=0 ttl=64 time=0.083 ms
...

# ping -c 3 1.1.1.1
PING 1.1.1.1 (1.1.1.1): 56 data bytes
64 bytes from 1.1.1.1: icmp_seq=0 ttl=60 time=0.920 ms
...

# fetch -v http://1.1.1.1/
...

# pkg -dd bootstrap -r FreeBSD
```

If ping(8) fails, check the firewall rules for blocked ICMP. If fetch(1) fails, check the firewall rules for blocked TCP on ports 80 and 443. If pkg(8) fails but ICMP and TCP work, this may be a fragmentation or MTU problem. Add `set reassemble yes no-df` to `/etc/pf.conf` and restart the firewall.

For debugging blocked packets, enable `log` on all block rules in `/etc/pf.conf` and use tcpdump(1) on the pflog(4) device:

```
# service pflog onerestart
# tcpdump -vvveni pflog0 icmp or net 10.88.0.0/16
```

18.14.3. MTU Consistency

Ensure MTU values are consistent across the primary network interface, the container bridge, and the container itself. Mismatched MTU values can cause packet fragmentation and connectivity failures.

In the following example, `vtnet0` has MTU 9000, but the bridge and containers have MTU 1500. In general, align the internal layout to match the external network:

```
# ifconfig vtnet0 mtu 9000
# ifconfig cni-podman0 mtu 9000
```

Inside the container:

```
# ifconfig eth0 mtu 9000
```

The bridges and epairs used by Podman are ephemeral and will need to be configured for each bridge creation and each container. In some cloud environments, adding `set reassemble yes no-df` to `/etc/pf.conf` may also be required.

Глава 19. Принудительный контроль доступа (MAC)

19.1. Обзор

FreeBSD поддерживает расширения безопасности, основанные на документах POSIX®.1e. Эти механизмы безопасности включают списки контроля доступа к файловой системе (“Списки контроля доступа”) и принудительный контроль доступа (MAC). MAC позволяет загружать модули контроля доступа для реализации политик безопасности. Некоторые модули обеспечивают защиту узкого подмножества системы, исиливая защиту определённой службы. Другие предоставляют всеобъемлющую безопасность с метками на всех субъектах и объектах. Обязательная часть определения указывает на то, что применение контроля осуществляется администраторами и операционной системой. Это отличается от механизма безопасности по умолчанию — обычного контроля доступа (Discretionary Access Control, DAC), где применение контроля остаётся на усмотрение пользователей.

Эта глава посвящена инфраструктуре MAC и набору модулей политики безопасности, предоставляемых FreeBSD для включения различных механизмов безопасности.

Прочтите эту главу, чтобы узнать:

- Терминология, связанная с инфраструктурой MAC.
- Возможности модулей политики безопасности MAC, а также разница между политикой с метками и без меток.
- Соображения, которые необходимо учитывать перед настройкой системы для использования инфраструктуры MAC.
- Какие модули политики безопасности MAC включены в FreeBSD и как их настроить.
- Как реализовать более безопасную среду с использованием инфраструктуры MAC.
- Как проверить конфигурацию MAC, чтобы убедиться в правильном использовании инфраструктуры.

Прежде чем читать эту главу, необходимо:

- Понимать основы UNIX® и FreeBSD (Основы FreeBSD).
- Иметь некоторое представление о безопасности и о том, как она относится к FreeBSD (Безопасность).

Неправильная настройка MAC может привести к потере доступа к системе, недовольству пользователей или невозможности использования функций, предоставляемых Xorg. Важно отметить, что нельзя полагаться исключительно на MAC для полной защиты системы. Инфраструктура MAC лишь дополняет существующую политику безопасности. Без грамотных мер безопасности и регулярных проверок система никогда не будет полностью

защищена.

Примеры, приведённые в этой главе, предназначены для демонстрации, и их настройки *не* следует применять в рабочей системе. Внедрение любой политики безопасности требует глубокого понимания, правильного проектирования и тщательного тестирования.

Хотя в этой главе рассматривается широкий спектр вопросов безопасности, связанных с инфраструктурой MAC, разработка новых модулей политики безопасности MAC не будет освещена. Ряд модулей политики безопасности, включённых в фреймворк MAC, обладают специфическими характеристиками, которые предоставляются как для тестирования, так и для разработки новых модулей. Дополнительную информацию об этих модулях политики безопасности и предоставляемых ими механизмах можно найти в mac_test(4), mac_stub(4) и mac_none(4).

19.2. Ключевые термины

В документации FreeBSD при обращении к инфраструктуре MAC используются следующие ключевые термины:

- *компартмент (compartment)*: набор программ и данных, которые должны быть разделены или изолированы, при этом пользователи получают явный доступ к определённым компонентам системы. Компартмент представляет собой группу, такую как рабочая группа, отдел, проект или тема. Компартменты позволяют реализовать политику безопасности на основе принципа "необходимо знать".
- *целостность (integrity)*: уровень доверия, который может быть оказан данным. По мере повышения целостности данных возрастает и возможность доверять этим данным.
- *уровень (level)*: увеличенное или уменьшенное значение атрибута безопасности. По мере повышения уровня его безопасность также считается более высокой.
- *метка (label)*: атрибут безопасности, который может быть применён к файлам, каталогам или другим элементам системы. Его можно рассматривать как штамп конфиденциальности. Когда метка назначается файлу, она описывает его свойства безопасности и разрешает доступ только файлам, пользователям и ресурсам с аналогичными настройками безопасности. Значение и интерпретация меток зависят от конфигурации политики. Некоторые политики рассматривают метку как показатель целостности или секретности объекта, тогда как другие могут использовать метки для хранения правил доступа.
- *множественные метки (multilabel)*: это свойство является параметром файловой системы, которое можно установить в однопользовательском режиме с помощью tunefs(8), во время загрузки с использованием fstab(5) или при создании новой файловой системы. Эта опция позволяет администратору применять различные метки MAC к разным объектам. Данная опция применяется только к модулям политики безопасности, которые поддерживают маркировку метками (далее — маркировку).
- *одиночная метка (single label)*: политика, при которой вся файловая система использует одну метку для контроля доступа к потокам данных. Если параметр `multilabel` не установлен, все файлы будут соответствовать одной и той же настройке метки.

- *объект (object)*: сущность, через которую информация передаётся под управлением *субъекта*. Это включает каталоги, файлы, поля, экраны, клавиатуры, память, магнитные накопители, принтеры или любые другие устройства хранения или передачи данных. Объект является контейнером данных или системным ресурсом. Доступ к объекту фактически означает доступ к его данным.
- *субъект (subject)*: любая активная сущность, вызывающая передачу информации между *объектами*, например, пользователь, пользовательский процесс или системный процесс. В FreeBSD это почти всегда поток, действующий в процессе от имени пользователя.
- *политика (policy)*: набор правил, определяющий, как достичь поставленных целей. Политика обычно документирует, каким образом следует обращаться с определёнными элементами. В этой главе под политикой понимается набор правил, контролирующих поток данных и информации, а также определяющих, кто имеет доступ к этим данным и информации.
- *верхний уровень (high-watermark)*: этот тип политики позволяет повышать уровни безопасности для доступа к информации более высокого уровня. В большинстве случаев исходный уровень восстанавливается после завершения процесса. В настоящее время в инфраструктуре MAC в FreeBSD отсутствует такой тип политики.
- *нижний уровень (low-watermark)*: этот тип политики позволяет понижать уровни безопасности для доступа к информации с более низким уровнем защиты. В большинстве случаев исходный уровень безопасности пользователя восстанавливается после завершения процесса. Единственный модуль политики безопасности в FreeBSD, использующий этот подход, — это mac_lomac(4).
- _ чувствительность (sensivity)_: обычно используется при обсуждении Многоуровневой Безопасности (Multilevel Security, MLS). Уровень чувствительности описывает, насколько важными или секретными должны быть данные. По мере увеличения уровня чувствительности возрастает и важность секретности, или конфиденциальности, данных.

19.3. Метки MAC

Метка MAC — это атрибут безопасности, который может быть применён к субъектам и объектам в системе. При установке метки администратор должен понимать её последствия, чтобы избежать неожиданного или нежелательного поведения системы. Доступные атрибуты объекта зависят от загруженного модуля политики, так как модули политики интерпретируют свои атрибуты по-разному.

Метка безопасности объекта используется как часть решения по контролю доступа в соответствии с политикой. В некоторых политиках метка содержит всю информацию, необходимую для принятия решения. В других политиках метки могут обрабатываться как часть более обширного набора правил.

Существует два типа политик меток: одноуровневые и многоуровневые. По умолчанию система использует одноуровневые метки. Администратор должен учитывать преимущества и недостатки каждого типа, чтобы реализовать политики, соответствующие требованиям модели безопасности системы.

Политика безопасности с одной меткой разрешает использование только одной метки для каждого субъекта или объекта. Поскольку политика с одной меткой применяет единый набор прав доступа во всей системе, это снижает нагрузку на администрирование, но уменьшает гибкость политик, поддерживающих маркировку. Однако во многих средах политика с одной меткой может быть всем, что требуется.

Политика с одной меткой несколько похожа на DAC, поскольку `root` настраивает политики так, чтобы пользователи попадали в соответствующие категории и уровни доступа. Заметное отличие заключается в том, что многие модули политики также могут ограничивать пользователя `root`. Базовый контроль над объектами затем передаётся группе, но `root` может отозвать или изменить настройки в любое время.

Когда это уместно, политику с несколькими метками можно установить на файловой системе UFS, передав `multilabel` в tunefs(8). Политика с несколькими метками позволяет каждому субъекту или объекту иметь свою собственную независимую метку MAC. Решение использовать политику с несколькими метками или одной меткой требуется только для политик, реализующих функцию маркировки, таких как `biba`, `lomac` и `mls`. Некоторые политики, такие как `seeotheruids`, `portacl` и `partition`, вообще не используют метки.

Использование политики с несколькими метками на разделе и установление модели безопасности с несколькими метками может увеличить административную нагрузку, так как всё в этой файловой системе имеет метку. Это включает каталоги, файлы и даже узлы устройств.

Следующая команда установит `multilabel` для указанной файловой системы UFS. Это можно сделать только в однопользовательском режиме и не является обязательным для файловой системы подкачки:

```
# tunefs -l enable /
```

Некоторые пользователи столкнулись с проблемами при установке флага `multilabel` на корневом разделе. Если это ваш случай, ознакомьтесь с Устранение неполадок инфраструктуры MAC.

Поскольку политика с несколькими метками устанавливается для каждой файловой системы, она может не потребоваться, если структура файловых систем хорошо продумана. Рассмотрим пример модели безопасности MAC для веб-сервера FreeBSD. На этой машине используется единая метка `biba/high` для всего в стандартных файловых системах. Если веб-сервер должен работать с `biba/low`, чтобы предотвратить возможность записи вверх, его можно установить в отдельную файловую систему UFS /usr/local с меткой `biba/low`.

19.3.1. Конфигурация меток

Практически все аспекты настройки модуля политики меток будут выполняться с помощью базовых системных утилит. Эти команды предоставляют простой интерфейс для настройки объекта или субъекта, а также для изменения и проверки конфигурации.

Вся настройка может быть выполнена с помощью `setfmac`, который используется для

установки меток MAC на объектах системы, и `setpmac`, который используется для установки меток на субъектах системы. Например, чтобы установить метку MAC `biba` в значение `high` для test:

```
# setfmac biba/high test
```

Если конфигурация выполнена успешно, командная строка вернётся без ошибок. Частая ошибка — `Permission denied`, которая обычно возникает при установке или изменении метки на защищённом объекте. Другие условия могут вызывать иные сбои. Например, файл может не принадлежать пользователю, пытающемуся изменить метку объекта, объект может не существовать или быть доступным только для чтения. Обязательная политика не позволит процессу изменить метку файла, возможно, из-за свойств самого файла, процесса или предлагаемого нового значения метки. Например, если пользователь с низким уровнем целостности попытается изменить метку файла с высоким уровнем целостности или если пользователь с низким уровнем целостности попытается изменить метку файла с низкого уровня на высокий, эти операции завершатся неудачей.

Системный администратор может использовать `setpmac` для переопределения настроек модуля политики, назначив другую метку вызываемому процессу:

```
# setfmac biba/high test
Permission denied
# setpmac biba/low setfmac biba/high test
# getfmac test
test: biba/high
```

Для уже запущенных процессов, таких как sendmail, обычно используется `getpmac` вместо этого. Эта команда принимает идентификатор процесса (PID) вместо имени команды. Если пользователи пытаются работать с файлом, к которому у них нет доступа, в соответствии с правилами загруженных модулей политики, будет отображаться ошибка `Операция не разрешена (Operation not permitted)`.

19.3.2. Предопределённые метки

Несколько модулей политики FreeBSD, поддерживающих функцию меток, предлагают три предопределённых метки: `low`, `equal` и `high`, где:

- `low` считается минимальным уровнем метки, который может быть установлен для объекта или субъекта. Установка этого уровня для объектов или субъектов блокирует их доступ к объектам или субъектам, помеченным как `high`.
- `equal` устанавливает, что субъект или объект отключён или не затронут, и должен использоваться только для объектов, считающихся исключёнными из политики.
- `high` предоставляет объекту или субъекту наивысший уровень доступа, доступный в модулях политик Biba и MLS.

Такие модули политик включают mac_biba(4), mac_mls(4) и mac_lomac(4). Каждый из

предопределённых меток устанавливает различные директивы потока информации. Обратитесь к справочной странице модуля, чтобы определить особенности стандартных конфигураций меток.

19.3.3. Числовые метки

Модули политик Biba и MLS поддерживают числовую метку, которая может быть установлена для указания точного уровня иерархического контроля. Этот числовой уровень используется для разделения или сортировки информации по различным группам классификации, разрешая доступ только к этой группе или к группе более высокого уровня. Например:

```
biba/10:2+3+6(5:2+3-20:2+3+4+5+6)
```

может интерпретироваться как "Метка/уровень целостности (grade) политики Biba 10: компартменты 2, 3 и 6: (уровень 5 ...)"

В этом примере первый уровень целостности будет считаться эффективным с эффективными компартментами, второй уровень — низкий уровень, а последний — высокий. В большинстве конфигураций такие детальные настройки не требуются, так как они считаются расширенными конфигурациями.

Системные объекты имеют только текущий уровень целостности и компартмент. Системные субъекты отражают диапазон доступных прав в системе, а сетевые интерфейсы используются для контроля доступа.

Уровни целостности и компартменты в паре субъект-объект используются для построения отношения, известного как *доминирование*, при котором субъект доминирует над объектом, объект доминирует над субъектом, ни один не доминирует над другим или оба доминируют друг над другом. Случай «оба доминируют» возникает, когда две метки равны. В силу природы информационных потоков в модели Biba пользователь имеет права на набор компартментов, которые могут соответствовать проектам, но объекты также имеют набор компартментов. Пользователям может потребоваться ограничить свои права с помощью `su` или `setpmac`, чтобы получить доступ к объектам в компартментах, от которой они не ограничены.

19.3.4. Пользовательские метки

Пользователи должны иметь метки, чтобы их файлы и процессы корректно взаимодействовали с политикой безопасности, определённой в системе. Это настраивается в /etc/login.conf с использованием классов входа. Каждый модуль политики, использующий метки, будет реализовывать настройку класса пользователя.

Чтобы установить метку класса пользователя по умолчанию, которая будет применяться MAC, добавьте запись `label`. Ниже приведён пример записи `label`, содержащей каждый модуль политики. Обратите внимание, что в реальной конфигурации администратор никогда не включает все модули политики. Рекомендуется ознакомиться с остальной частью этой главы перед внедрением любой конфигурации.

```
default:\
    :welcome=/etc/motd:\
    :setenv=MAIL=/var/mail/$,BLOCKSIZE=K:\
    :path=~/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin:\
    :manpath=/usr/share/man /usr/local/man:\
    :nologin=/usr/sbin/nologin:\
    :cputime=1h30m:\
    :datasize=8M:\
    :vmemoryuse=100M:\
    :stacksize=2M:\
    :memorylocked=4M:\
    :memoryuse=8M:\
    :filesize=8M:\
    :coredumpsize=8M:\
    :openfiles=24:\
    :maxproc=32:\
    :priority=0:\
    :requirehome:\
    :passwordtime=91d:\
    :umask=022:\
    :ignoretime@:\
    :label=partition/13,mls/5,biba/10(5-15),lomac/10[2]:
```

Хотя пользователи не могут изменить значение по умолчанию, они могут изменить свою метку после входа в систему, в соответствии с ограничениями политики. В приведённом выше примере политика Biba указывает, что минимальный уровень целостности процесса — `5`, максимальный — `15`, а эффективная метка по умолчанию — `10`. Процесс будет выполняться с меткой `10`, пока не решит изменить её, например, если пользователь воспользуется `setpmac`, что будет ограничено политикой Biba в рамках заданного диапазона.

После любого изменения в файле login.conf необходимо перестроить базу данных возможностей классов входа с помощью команды `cap_mkdb`.

Многие сайты имеют большое количество пользователей, требующих нескольких различных классов пользователей. Требуется тщательное планирование, так как это может стать сложным в управлении.

19.3.5. Метки сетевых интерфейсов

Метки могут быть установлены на сетевых интерфейсах для помощи в контроле потока данных в сети. Политики, использующие метки сетевых интерфейсов, работают так же, как политики, применяемые к объектам. Например, пользователи с высокими уровнями в Biba не смогут получить доступ к сетевым интерфейсам с меткой `low`.

При установке MAC-метки на сетевых интерфейсах, `maclabel` может быть передан в `ifconfig`:

```
# ifconfig bge0 maclabel biba/equal
```

Этот пример установит метку MAC `biba/equal` на интерфейсе `bge0`. При использовании настройки вида `biba/high(low-high)` всю метку следует заключить в кавычки, чтобы избежать ошибки.

Каждый модуль политики, поддерживающий метки, имеет настраиваемый параметр, который может использоваться для отключения метки MAC на сетевых интерфейсах. Установка метки в значение `equal` даст аналогичный эффект. Для получения дополнительной информации об этих настраиваемых параметрах ознакомьтесь с выводом `sysctl`, справочными страницами политик и информацией в остальной части этой главы.

19.4. Планирование конфигурации безопасности

Прежде чем внедрять какие-либо политики MAC, рекомендуется этап планирования. На этапе планирования администратор должен учесть требования и цели внедрения, такие как:

- Как классифицировать информацию и ресурсы, доступные в целевых системах.
- Какую информацию или ресурсы следует ограничить в доступе, а также тип ограничений, которые должны быть применены.
- Какие модули MAC потребуются для достижения этой цели.

Пробный запуск доверенной системы и её конфигурации должен быть выполнен *до* использования реализации MAC в производственных системах. Поскольку различные среды имеют разные потребности и требования, создание полного профиля безопасности уменьшит необходимость изменений после ввода системы в эксплуатацию.

Рассмотрим, как инфраструктура MAC усиливает безопасность системы в целом. Различные модули политики безопасности, предоставляемые инфраструктурой MAC, могут использоваться для защиты сети и файловых систем или для блокировки доступа пользователей к определённым портам и сокетам. Возможно, наилучшее применение модулей политики — это загрузка нескольких модулей политики безопасности одновременно для создания среды MLS. Такой подход отличается от политики усиления защиты, которая обычно укрепляет элементы системы, используемые только для определённых целей. Недостатком MLS является увеличение административных затрат.

Накладные расходы минимальны по сравнению с долгосрочным эффектом от инфраструктуры, который предоставляет возможность выбирать необходимые политики для конкретной конфигурации и минимизирует снижение производительности. Уменьшение поддержки ненужных политик может повысить общую производительность системы, а также обеспечить гибкость выбора. Хорошая реализация должна учитывать общие требования безопасности и эффективно внедрять различные модули политик безопасности, предоставляемые инфраструктурой.

Система, использующая MAC, гарантирует, что пользователь не сможет по своему желанию изменять атрибуты безопасности. Все пользовательские утилиты, программы и скрипты должны работать в рамках ограничений, установленных правилами доступа выбранных модулей политики безопасности, а управление правилами доступа MAC находится в руках системного администратора.

Обязанностью системного администратора является тщательный выбор подходящих модулей политики безопасности. Для среды, где требуется ограничить контроль доступа по сети, модули политик mac_portacl(4), mac_ifoff(4) и mac_biba(4) могут стать хорошей отправной точкой. В среде, где необходима строгая конфиденциальность объектов файловой системы, следует рассмотреть модули политик mac_bsdextended(4) и mac_mls(4).

Решения о политиках могут приниматься на основе конфигурации сети. Если только определённые пользователи должны иметь доступ к ssh(1), модуль политики mac_portacl(4) является хорошим выбором. В случае файловых систем доступ к объектам может считаться конфиденциальным для одних пользователей, но не для других. Например, большая команда разработчиков может быть разделена на меньшие проекты, где разработчики из проекта A не должны иметь доступа к объектам, созданным разработчиками из проекта B. Однако оба проекта могут нуждаться в доступе к объектам, созданным разработчиками из проекта C. Используя различные модули политик безопасности, предоставляемые MAC-фреймворком, пользователи могут быть разделены на эти группы и затем получить доступ к соответствующим объектам.

Каждый модуль политики безопасности имеет уникальный способ обработки общей безопасности системы. Выбор модуля должен основываться на продуманной политике безопасности, которая может потребовать пересмотра и повторной реализации. Понимание различных модулей политики безопасности, предоставляемых инфраструктурой MAC, поможет администраторам выбрать наилучшие политики для их ситуаций.

Остальная часть этой главы посвящена доступным модулям, описанию их использования и настройки, а в некоторых случаях содержит рекомендации по их применению в различных ситуациях.

Внедрение MAC во многом похоже на настройку межсетевого экрана, так как необходимо соблюдать осторожность, чтобы не оказаться полностью заблокированным в системе. Следует предусмотреть возможность возврата к предыдущей конфигурации, а реализацию MAC через удалённое соединение следует выполнять с особой осторожностью.

19.5. Доступные политики MAC

Стандартное ядро FreeBSD включает `options MAC`. Это означает, что каждый модуль, входящий в состав инфраструктуры MAC, может быть загружен с помощью `kldload` как модуль ядра во время выполнения. После тестирования модуля добавьте его имя в /boot/loader.conf, чтобы он загружался при старте системы. Каждый модуль также предоставляет опцию ядра для администраторов, которые предпочитают компилировать собственное ядро системы.

FreeBSD включает набор политик, которые охватывают большинство требований безопасности. Каждая политика кратко описана ниже. Последние три политики поддерживают целочисленные настройки вместо трёх стандартных меток.

19.5.1. Политика MAC — See Other UIDs

Название модуля: mac_seeotheruids.ko

Строка конфигурации ядра: `options MAC_SEEOTHERUIDS`

Опция загрузки: `mac_seeotheruids_load="YES"`

Модуль mac_seeotheruids(4) расширяет настройки `security.bsd.see_other_uids` и `security.bsd.see_other_gids sysctl`. Эта опция не требует предварительной установки меток для настройки и может работать прозрачно с другими модулями.

После загрузки модуля следующие настройки `sysctl` могут быть использованы для управления его функциями:

- `security.mac.seeotheruids.enabled` включает модуль и реализует настройки по умолчанию, которые запрещают пользователям возможность просматривать процессы и сокеты, принадлежащие другим пользователям.
- `security.mac.seeotheruids.specificgid_enabled` позволяет исключить указанные группы из действия этой политики. Чтобы исключить определённые группы, используйте параметр `security.mac.seeotheruids.specificgid=XXX sysctl`, заменив *XXX* на числовой идентификатор группы, которую нужно исключить.
- `security.mac.seeotheruids.primarygroup_enabled` используется для исключения определённых первичных групп из этой политики. При использовании этого параметра `security.mac.seeotheruids.specificgid_enabled` не может быть установлен.

19.5.2. Политика MAC — BSD Extended

Имя модуля: mac_bsdextended.ko

Строка конфигурации ядра: `options MAC_BSDEXTENDED`

Параметр загрузки: `mac_bsdextended_load="YES"`

Модуль mac_bsdextended(4) обеспечивает файловый межсетевой экран. Он расширяет стандартную модель прав доступа к файловой системе, позволяя администратору создавать набор правил, подобный межсетевому экрану, для защиты файлов, утилит и каталогов в иерархии файловой системы. При попытке доступа к объекту файловой системы происходит перебор списка правил до тех пор, пока не будет найдено соответствующее правило или не будет достигнут конец списка. Это поведение можно изменить с помощью параметра `security.mac.bsdextended.firstmatch_enabled`. Подобно другим модулям межсетевого экрана в FreeBSD, файл с правилами контроля доступа может быть создан и прочитан системой во время загрузки с использованием переменной rc.conf(5).

Список правил может быть введён с помощью ugidfw(8), синтаксис которого похож на ipfw(8). Дополнительные инструменты могут быть написаны с использованием функций из библиотеки libugidfw(3).

После загрузки модуля mac_bsdextended(4) для просмотра текущей конфигурации правил можно использовать следующую команду:

```
# ugidfw list
0 slots, 0 rules
```

По умолчанию никакие правила не определены, и доступ полностью открыт. Чтобы создать правило, которое блокирует доступ для всех пользователей, кроме `root`:

```
# ugidfw add subject not uid root new object not uid root mode n
```

Хотя это правило просто реализовать, это очень плохая идея, так как оно блокирует всех пользователей от выполнения любых команд. Более реалистичный пример запрещает `user1` любой доступ, включая просмотр каталогов, к домашнему каталогу `user2`:

```
# ugidfw set 2 subject uid user1 object uid user2 mode n
# ugidfw set 3 subject uid user1 object gid user2 mode n
```

Вместо `user1` можно использовать `not uid user2`, чтобы применять одинаковые ограничения доступа для всех пользователей. Однако пользователь `root` не подвержен влиянию этих правил.

Следует проявлять крайнюю осторожность при работе с этим модулем, так как неправильное использование может заблокировать доступ к определённым частям файловой системы.

19.5.3. Политика MAC — подавление интерфейсов

Имя модуля: mac_ifoff.ko

Строка конфигурации ядра: `options MAC_IFOFF`

Параметр загрузки: `mac_ifoff_load="YES"`

Модуль mac_ifoff(4) используется для отключения сетевых интерфейсов на лету и предотвращения их включения во время загрузки системы. Он не использует метки и не зависит от других модулей MAC.

Большая часть управления этим модулем осуществляется через следующие настраиваемые параметры `sysctl`:

- `security.mac.ifoff.lo_enabled` включает или отключает весь трафик на интерфейсе loopback, lo(4).
- `security.mac.ifoff.bpfrecv_enabled` включает или отключает весь трафик на интерфейсе Berkeley Packet Filter, bpf(4).
- `security.mac.ifoff.other_enabled` включает или отключает трафик на всех остальных интерфейсах.

Одним из наиболее распространённых применений mac_ifoff(4) является мониторинг сети в среде, где сетевой трафик не должен разрешаться во время последовательности загрузки. Другое применение — написание скрипта, который использует приложение, например security/aide, для автоматической блокировки сетевого трафика при обнаружении новых или изменённых файлов в защищённых каталогах.

19.5.4. Политика MAC - списки управления доступом к портам

Имя модуля: mac_portacl.ko

Строка конфигурации ядра: MAC_PORTACL

Параметр загрузки: mac_portacl_load="YES"

Модуль mac_portacl(4) используется для ограничения привязки к локальным TCP- и UDP-портам, позволяя непривилегированным пользователям (не root) привязываться к указанным привилегированным портам ниже 1024.

После загрузки этот модуль включает политику MAC для всех сокетов. Доступны следующие настраиваемые параметры:

- security.mac.portacl.enabled включает или отключает политику полностью.
- security.mac.portacl.port_high устанавливает наибольший номер порта, который защищает mac_portacl(4).
- security.mac.portacl.suser_exempt при установке в ненулевое значение освобождает пользователя root от действия данной политики.
- security.mac.portacl.rules задаёт политику в виде текстовой строки формата правило[,правило,…], с любым необходимым количеством правил, где каждое правило имеет вид тип_идентификатора:идентификатор:протокол:порт. тип_идентификатора может быть uid или gid. Параметр протокол ` принимает значения `tcp или udp. Параметр порт указывает номер порта, к которому разрешено привязываться указанному пользователю или группе. Для параметров идентификатор пользователя, идентификатор группы и порт можно использовать только числовые значения.

По умолчанию порты ниже 1024 могут использоваться только привилегированными процессами, работающими от имени root. Чтобы разрешить непривилегированным процессам привязываться к портам ниже 1024 с помощью mac_portacl(4), задайте следующие настраиваемые параметры следующим образом:

```
# sysctl security.mac.portacl.port_high=1023
# sysctl net.inet.ip.portrange.reservedlow=0
# sysctl net.inet.ip.portrange.reservedhigh=0
```

Чтобы предотвратить влияние этой политики на пользователя root, установите security.mac.portacl.suser_exempt в ненулевое значение.

```
# sysctl security.mac.portacl.suser_exempt=1
```

Чтобы пользователь `www` с UID 80 мог привязываться к порту 80 без необходимости в привилегиях `root`:

```
# sysctl security.mac.portacl.rules=uid:80:tcp:80
```

Следующий пример разрешает пользователю с UID 1001 привязываться к TCP-портам 110 (POP3) и 995 (POP3s):

```
# sysctl security.mac.portacl.rules=uid:1001:tcp:110,uid:1001:tcp:995
```

19.5.5. Политика MAC — разделы процессов

Имя модуля: mac_partition.ko

Строка конфигурации ядра: `options MAC_PARTITION`

Опция загрузки: `mac_partition_load="YES"`

Политика mac_partition(4) помещает процессы в определённые "разделы" на основе их метки MAC. Большая часть настройки этой политики выполняется с помощью setpmac(8). Для этой политики доступна одна настраиваемая переменная `sysctl`:

- `security.mac.partition.enabled` включает принудительное применение разделов процессов MAC.

Когда эта политика включена, пользователи смогут видеть только свои процессы и процессы в своём разделе, но не смогут работать с утилитами за пределами этого раздела. Например, пользователь из класса `insecure` не сможет получить доступ к `top`, а также ко многим другим командам, которые должны запускать процессы.

В этом примере `top` добавляется к набору меток пользователей в классе `insecure`. Все процессы, запущенные пользователями из класса `insecure`, останутся с меткой `partition/13`.

```
# setpmac partition/13 top
```

Эта команда отображает метку раздела и список процессов:

```
# ps Zax
```

Эта команда отображает метку раздела процессов другого пользователя и его текущие запущенные процессы:

```
# ps -ZU trhodes
```

Пользователи могут видеть процессы с меткой `root`, если не загружена политика mac_seeotheruids(4).

19.5.6. Модуль многоуровневой безопасности MAC

Имя модуля: mac_mls.ko

Строка конфигурации ядра: `options MAC_MLS`

Параметр загрузки: `mac_mls_load="YES"`

Политика mac_mls(4) контролирует доступ между субъектами и объектами в системе, применяя строгую политику управления потоком информации.

В средах MLS в метке каждого субъекта или объекта устанавливается уровень "допуска" вместе с компартментами. Поскольку эти уровни допуска могут достигать значений, превышающих несколько тысяч, тщательная настройка каждого субъекта или объекта была бы сложной задачей. Для снижения административной нагрузки в эту политику включены три метки: `mls/low`, `mls/equal` и `mls/high`, где:

- Все объекты, помеченные меткой `mls/low`, будут иметь низкий уровень доступа и не смогут обращаться к информации более высокого уровня. Эта метка также предотвращает запись или передачу информации от объектов с более высоким уровнем доступа к объектам с более низким уровнем.
- `mls/equal` следует размещать на объектах, которые должны быть освобождены от политики.
- `mls/high` — это наивысший возможный уровень допуска. Объекты с этой меткой будут доминировать над всеми остальными объектами в системе; однако они не допустят утечки информации к объектам более низкого класса.

MLS предоставляет:

- Иерархический уровень безопасности с набором неиерархических категорий.
- Фиксированные правила `нет чтения вверх, нет записи вниз`. Это означает, что субъект может иметь право чтения объектов на своём уровне или ниже, но не выше. Аналогично, субъект может иметь право записи объектов на своём уровне или выше, но не ниже.
- Секретность, или предотвращение несанкционированного раскрытия данных.
- Основы проектирования систем, которые одновременно обрабатывают данные с разными уровнями конфиденциальности, не допуская утечки информации между секретными и конфиденциальными данными.

Доступны следующие настраиваемые параметры `sysctl`:

- `security.mac.mls.enabled` используется для включения или отключения политики MLS.
- `security.mac.mls.ptys_equal` помечает все устройства pty(4) как `mls/equal` при создании.
- `security.mac.mls.revocation_enabled` отзывает доступ к объектам после изменения их метки на метку более низкого уровня целостности.
- `security.mac.mls.max_compartments` устанавливает максимальное количество уровней компартментов, разрешенных в системе.

Для работы с метками MLS используйте setfmac(8). Чтобы назначить метку объекту:

```
# setfmac mls/5 test
```

Чтобы получить метку MLS для файла test:

```
# getfmac test
```

Другой подход заключается в создании основного файла политики в /etc/, который определяет информацию о политике MLS, и передаче этого файла в `setfmac`.

При использовании модуля политики MLS администратор планирует контролировать поток конфиденциальной информации. Значение по умолчанию `block read up block write down` устанавливает всё в состояние low. Вся информация доступна, и администратор постепенно повышает её конфиденциальность.

Помимо трёх основных вариантов меток, администратор может группировать пользователей и группы по мере необходимости, чтобы блокировать поток информации между ними. Возможно, будет проще рассматривать информацию на уровнях допуска, используя описательные слова, такие как классификации `Confidential` (`Конфиденциально`), `Secret` (`Секретно`) и `Top Secret` (`Совершенно секретно`). Некоторые администраторы вместо этого создают разные группы на основе уровней проектов. Независимо от метода классификации, перед внедрением ограничительной политики должен существовать продуманный план.

Некоторые примеры ситуаций для модуля политики MLS включают веб-сервер электронной коммерции, файловый сервер с критически важной информацией компании и среды финансовых учреждений.

19.5.7. Модуль MAC Biba

Имя модуля: mac_biba.ko

Строка конфигурации ядра: `options MAC_BIBA`

Опция загрузки: `mac_biba_load="YES"`

Модуль mac_biba(4) загружает политику MAC Biba. Эта политика похожа на политику MLS, за исключением того, что правила передачи информации слегка изменены в обратном порядке. Это предотвращает поток конфиденциальной информации вниз, тогда как политика MLS предотвращает поток конфиденциальной информации вверх.

В средах Biba для каждого субъекта или объекта устанавливается метка «целостности». Эти метки состоят из иерархических уровней целостности и неиерархических компонентов. По мере повышения уровня увеличивается и его целостность.

Поддерживаемые метки: `biba/low`, `biba/equal` и `biba/high`, где:

- `biba/low` считается самой низкой целостностью, которую может иметь объект или субъект. Установка этого уровня на объекты или субъекты блокирует их запись в объекты или субъекты с меткой `biba/high`, но не предотвращает чтение.
- `biba/equal` следует размещать только на объектах, которые считаются исключёнными из политики.
- `biba/high` разрешает запись в объекты с более низкой меткой, но запрещает чтение этих объектов. Рекомендуется устанавливать эту метку для объектов, которые влияют на целостность всей системы.

Biba обеспечивает:

- Иерархические уровни целостности с набором неиерархических категорий целостности.
- Фиксированные правила — это `нет записи вверх, нет чтения вниз`, что противоположно MLS. Субъект может иметь право записи в объекты на своём уровне или ниже, но не выше. Аналогично, субъект может иметь право чтения объектов на своём уровне или выше, но не ниже.
- Целостность за счет предотвращения нежелательного изменения данных.
- Уровни целостности вместо уровней чувствительности MLS.

Следующие настраиваемые параметры могут быть использованы для управления политикой Biba:

- `security.mac.biba.enabled` используется для включения или отключения принудительного применения политики Biba на целевой машине.
- `security.mac.biba.ptys_equal` используется для отключения политики Biba на устройствах pty(4).
- `security.mac.biba.revocation_enabled` принудительно отзывает доступ к объектам, если их метка изменяется так, чтобы доминировать над субъектом.

Для доступа к настройкам политики Biba для системных объектов используйте `setfmac` и `getfmac`:

```
# setfmac biba/low test
# getfmac test
test: biba/low
```

Целостность, которая отличается от конфиденциальности, используется для гарантии того, что информация не будет изменена ненадёжными сторонами. Это включает информацию, передаваемую между субъектами и объектами. Она обеспечивает пользователям возможность изменять или получать доступ только к той информации, к которой у них

есть явный доступ. Модуль политики безопасности mac_biba(4) позволяет администратору настроить, какие файлы и программы пользователь может просматривать и запускать, гарантируя, что эти программы и файлы считаются системой доверенными для данного пользователя.

В ходе начального этапа планирования администратор должен быть готов разделить пользователей по уровням целостности (grade), уровням объектов (level) и областям. После включения этого модуля политики система по умолчанию перейдет на высокий уровень метки, и администратору потребуется настроить различные уровни целостности и уровни объектов для пользователей. Вместо использования уровней доступа хорошим методом планирования может стать использование тематик. Например, разрешить разработчикам доступ на изменение только к репозиторию исходного кода, компилятору исходного кода и другим инструментам разработки. Остальные пользователи будут распределены по другим категориям, таким как тестировщики, дизайнеры или конечные пользователи, и им будет разрешен только доступ на чтение.

Субъект с более низким уровнем целостности не может записывать данные в субъект с более высоким уровнем целостности, а субъект с более высоким уровнем целостности не может просматривать или читать объект с более низким уровнем целостности. Установка метки на минимально возможном уровне может сделать объект недоступным для субъектов. Перспективными средами для использования этого модуля политики безопасности могут быть ограниченный веб-сервер, машина для разработки и тестирования, а также репозиторий исходного кода. Менее полезной реализацией будет персональная рабочая станция, машина, используемая в качестве маршрутизатора, или сетевой межсетевой экран.

19.5.8. Модуль MAC Low-watermark (нижний порог)

Имя модуля: mac_lomac.ko

Строка конфигурации ядра: `options MAC_LOMAC`

Параметр загрузки: `mac_lomac_load="YES"`

В отличие от политики MAC Biba, политика mac_lomac(4) разрешает доступ к объектам с более низким уровнем целостности только после понижения уровня целостности, чтобы не нарушать правила целостности.

Политика целостности Low-watermark работает почти идентично Biba, за исключением использования плавающих меток для поддержки понижения уровня субъекта через компартмент с вспомогательным уровнем целостности. Этот вторичный компартмент имеет вид `[auxgrade]`. При назначении политики с вспомогательным уровнем целостности используйте синтаксис `lomac/10[2]`, где `2` — это вспомогательный уровень целостности.

Данная политика основывается на повсеместной маркировке всех системных объектов метками целостности, позволяя субъектам читать из объектов с низкой целостностью, а затем понижая уровень метки на субъекте с помощью `[auxgrade]`, чтобы предотвратить последующие записи в объекты с высокой целостностью. Эта политика может обеспечить большую совместимость и потребовать меньше начальной настройки по сравнению с Biba.

Как и в политиках Biba и MLS, `setfmac` и `setpmac` используются для назначения меток объектам системы:

```
# setfmac /usr/home/trhodes lomac/high[low]
# getfmac /usr/home/trhodes lomac/high[low]
```

Вспомогательный уровень целостности `low` — это функция, предоставляемая только политикой MACLOMAC.

19.6. Блокировка пользователя

Этот пример рассматривает относительно небольшую систему хранения данных с менее чем пятьюдесятью пользователями. Пользователи будут иметь возможность входа в систему и могут хранить данные и получать доступ к ресурсам.

Для данного сценария модули политик mac_bsdextended(4) и mac_seeotheruids(4) могут сосуществовать и блокировать доступ к системным объектам, скрывая при этом пользовательские процессы.

Начните с добавления следующей строки в /boot/loader.conf:

```
mac_seeotheruids_load="YES"
```

Модуль политики безопасности mac_bsdextended(4) может быть активирован добавлением следующей строки в /etc/rc.conf:

```
ugidfw_enable="YES"
```

Файл с правилами по умолчанию, хранящийся в /etc/rc.bsdextended, будет загружен при инициализации системы. Однако стандартные записи могут потребовать изменения. Поскольку предполагается, что данная машина будет обслуживать только пользователей, все строки можно оставить закомментированными, за исключением последних двух, чтобы обеспечить принудительную загрузку системных объектов, принадлежащих пользователям, по умолчанию.

Добавьте необходимых пользователей на эту машину и перезагрузитесь. Для тестирования попробуйте войти в систему под разными пользователями на двух консолях. Выполните `ps -aux`, чтобы проверить, видны ли процессы других пользователей. Убедитесь, что выполнение ls(1) для домашнего каталога другого пользователя завершается ошибкой.

Не пытайтесь проводить тестирование от пользователя `root`, если специальные параметры `sysctl` не были изменены для блокировки доступа суперпользователя.

При добавлении нового пользователя его правило mac_bsdextended(4) не будет в списке набора правил. Чтобы быстро обновить набор правил, выгрузите модуль политики безопасности и загрузите его снова с помощью

kldunload(8) и kldload(8).

19.7. Nagios в MAC клетке

В этом разделе показаны шаги, необходимые для внедрения системы мониторинга сети Nagios в среде MAC. Это пример, который требует от администратора проверки соответствия реализованной политики требованиям безопасности сети перед использованием в рабочей среде.

Этот пример требует установки `multilabel` на каждой файловой системе. Также предполагается, что net-mgmt/nagios-plugins, net-mgmt/nagios и www/apache22 установлены, настроены и корректно работают до попытки интеграции в инфраструктуре MAC.

19.7.1. Создайте небезопасный класс пользователя

Начните процедуру, добавив следующий класс пользователя в /etc/login.conf:

```
insecure:\
:welcome=/etc/motd:\
:setenv=MAIL=/var/mail/$,BLOCKSIZE=K:\
:path=~/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin
:manpath=/usr/share/man /usr/local/man:\
:nologin=/usr/sbin/nologin:\
:cputime=1h30m:\
:datasize=8M:\
:vmemoryuse=100M:\
:stacksize=2M:\
:memorylocked=4M:\
:memoryuse=8M:\
:filesize=8M:\
:coredumpsize=8M:\
:openfiles=24:\
:maxproc=32:\
:priority=0:\
:requirehome:\
:passwordtime=91d:\
:umask=022:\
:ignoretime@:\
:label=biba/10(10-10):
```

Затем добавьте следующую строку в раздел класса пользователя по умолчанию:

```
:label=biba/high:
```

Сохраните изменения и выполните следующую команду для перестроения базы данных:

```
# cap_mkdb /etc/login.conf
```

19.7.2. Настройте пользователей

Установите пользователя `root` в класс по умолчанию с помощью:

```
# pw usermod root -L default
```

Все пользовательские учётные записи, кроме `root`, теперь требуют указания класса входа. Класс входа обязателен, в противном случае пользователям будет отказано в доступе к распространённым командам. Следующий скрипт на `sh` должен помочь:

```
# for x in `awk -F: '($3 >= 1001) && ($3 != 65534) { print $1 }' \
    /etc/passwd`; do pw usermod $x -L default; done;
```

Затем добавьте учётные записи `nagios` и `www` в класс insecure:

```
# pw usermod nagios -L insecure
# pw usermod www -L insecure
```

19.7.3. Создайте файл контекстов

Файл контекстов теперь должен быть создан как /etc/policy.contexts:

```
# This is the default BIBA policy for this system.

# System:
/var/run(/.*)?          biba/equal

/dev/(/.*)?         biba/equal

/var                biba/equal
/var/spool(/.*)?        biba/equal

/var/log(/.*)?          biba/equal

/tmp(/.*)?          biba/equal
/var/tmp(/.*)?          biba/equal

/var/spool/mqueue       biba/equal
/var/spool/clientmqueue     biba/equal

# For Nagios:
/usr/local/etc/nagios(/.*)? biba/10
```

```
/var/spool/nagios(/.*)?     biba/10

# For apache
/usr/local/etc/apache(/.*)? biba/10
```

Эта политика обеспечивает безопасность, устанавливая ограничения на поток информации. В данной конкретной конфигурации пользователям, включая `root`, никогда не должно быть разрешено обращаться к Nagios. Конфигурационные файлы и процессы, являющиеся частью Nagios, будут полностью самодостаточными или изолированными.

Этот файл будет прочитан после выполнения `setfsmac` для каждой файловой системы. В этом примере устанавливается политика для корневой файловой системы:

```
# setfsmac -ef /etc/policy.contexts /
```

Далее добавьте эти изменения в основной раздел файла /etc/mac.conf:

```
default_labels file ?biba
default_labels ifnet ?biba
default_labels process ?biba
default_labels socket ?biba
```

19.7.4. Конфигурация загрузчика

Для завершения настройки добавьте следующие строки в /boot/loader.conf:

```
mac_biba_load="YES"
mac_seeotheruids_load="YES"
security.mac.biba.trust_all_interfaces=1
```

Добавьте следующую строку в конфигурацию сетевой карты, хранящуюся в /etc/rc.conf. Если основная настройка сети выполняется через DHCP, это может потребовать ручной настройки после каждой загрузки системы:

```
maclabel biba/equal
```

19.7.5. Проверка конфигурации

Сначала убедитесь, что веб-сервер и Nagios не будут запускаться при инициализации системы и перезагрузке. Убедитесь, что `root` не имеет доступа к любым файлам в конфигурационном каталоге Nagios. Если `root` может просматривать содержимое /var/spool/nagios, значит что-то не так. Вместо этого должна возвращаться ошибка "permission denied".

Если все выглядит нормально, можно запустить Nagios, Apache и Sendmail:

```
# cd /etc/mail && make stop && \
setpmac biba/equal make start && setpmac biba/10\(10-10\) apachectl start && \
setpmac biba/10\(10-10\) /usr/local/etc/rc.d/nagios.sh forcestart
```

Тщательно проверьте, чтобы всё работало правильно. Если нет, проверьте файлы журналов на наличие сообщений об ошибках. При необходимости используйте sysctl(8) для отключения модуля политики безопасности mac_biba(4) и попробуйте запустить всё снова как обычно.

Пользователь root всё ещё может изменять параметры безопасности и редактировать конфигурационные файлы. Следующая команда разрешит понижение уровня целостности политики безопасности для нового запущенного shell:

```
# setpmac biba/10 csh
```

Чтобы предотвратить это, принудительно ограничьте пользователя диапазоном с помощью login.conf(5). Если setpmac(8) попытается выполнить команду вне пределов компартмента, будет возвращена ошибка и команда не выполнится. В данном случае установите root в biba/high(high-high).

19.8. Устранение проблем с инфраструктурой MAC

Этот раздел посвящён распространённым ошибкам конфигурации и способам их устранения.

Флаг multilabel не сохраняется на корневом (/) разделе

Следующие действия могут помочь устранить эту временную ошибку:

1. Отредактируйте файл /etc/fstab и установите корневой раздел в ro для режима только для чтения.
2. Перезагрузитесь в однопользовательском режиме.
3. Выполните команду tunefs -l enable для раздела /.
4. Перезагрузите систему.
5. Выполните mount -urw/, измените ro обратно на rw в /etc/fstab и перезагрузите систему снова.
6. Перепроверьте вывод команды mount, чтобы убедиться, что опция multilabel корректно установлена для корневой файловой системы.

После настройки безопасной среды с MAC, Xorg больше не запускается

Это может быть вызвано политикой MAC partition или ошибкой маркировки в одной из политик маркировки MAC. Для диагностики попробуйте следующее:

1. Проверьте сообщение об ошибке. Если пользователь находится в классе `insecure`, проблема может быть в политике `partition`. Попробуйте вернуть пользователя в класс `default` и пересобрать базу данных с помощью `cap_mkdb`. Если это не решит проблему, перейдите ко второму шагу.
2. Перепроверьте, что политики меток правильно установлены для пользователя, Xorg и записей в /dev.
3. Если ни один из этих способов не решит проблему, отправьте сообщение об ошибке и описание окружения на Список рассылки, посвящённый вопросам и ответам пользователей FreeBSD.

Появляется ошибка `_secure_path: unable to stat .login_conf`

Этот ошибка может возникать, когда пользователь пытается переключиться с пользователя `root` на другого пользователя в системе. Это сообщение обычно появляется, когда у пользователя установлена более высокая метка, чем у пользователя, в которого он пытается переключиться. Например, если у `joe` метка по умолчанию `biba/low`, а у `root` — `biba/high`, `root` не сможет просмотреть домашний каталог `joe`. Это произойдет независимо от того, использовал ли `root` команду `su` для переключения на `joe`, так как модель целостности Biba не позволяет `root` просматривать объекты с более низким уровнем целостности.

Система больше не распознает `root`

Когда это происходит, `whoami` возвращает `0`, а `su` выводит `who are you?`.

Это может произойти, если политика меток была отключена через sysctl(8) или модуль политики был выгружен. Если политика отключена, необходимо перенастроить базу данных возможностей входа. Проверьте файл /etc/login.conf, чтобы убедиться, что все опции `label` удалены, и перестройте базу данных с помощью `cap_mkdb`.

Это также может произойти, если политика ограничивает доступ к master.passwd. Обычно это происходит, когда администратор изменяет файл под меткой, которая конфликтует с общей политикой, используемой системой. В таких случаях система прочитает информацию о пользователе, но доступ будет заблокирован, так как файл унаследовал новую метку. Отключите политику с помощью sysctl(8), и всё должно вернуться в норму.

Глава 20. Аудит событий безопасности

20.1. Обзор

Операционная система FreeBSD включает поддержку аудита событий безопасности. Аудит событий обеспечивает надёжное, детализированное и настраиваемое журналирование различных системных событий, связанных с безопасностью, включая входы в систему, изменения конфигурации, доступ к файлам и сети. Эти записи журнала могут быть неоценимыми для мониторинга системы в реальном времени, обнаружения вторжений и "посмертного" анализа после краха системы. FreeBSD реализует опубликованный Sun™ программный интерфейс Basic Security Module (BSM) и формат файлов, и также совместима с реализациями аудита Solaris™ и Mac OS® X.

Эта глава посвящена установке и настройке аудита событий. В ней объясняются политики аудита и приводится пример конфигурации аудита.

Прочтите эту главу, чтобы узнать:

- Что такое аудит событий и как он работает.
- Как настроить аудит событий в FreeBSD для пользователей и процессов.
- Как просмотреть журнал аудита с использованием инструментов сокращения и просмотра аудита.

Прежде чем читать эту главу, необходимо:

- Понимать основы UNIX® и FreeBSD (Основы FreeBSD).
- Быть знакомым с основами настройки и компиляции ядра (Настройка ядра FreeBSD).
- Иметь некоторое представление о безопасности и о том, как она относится к FreeBSD (Безопасность).

У системы аудита есть несколько известных ограничений. Не все связанные с безопасностью системные события подлежат аудиту, а некоторые механизмы входа, такие как дисплейные менеджеры на основе Xorg и сторонние демоны, не настраивают аудит для сеансов пользовательского входа должным образом.

Система аудита событий безопасности способна создавать очень подробные журналы активности системы. На загруженной системе данные файлов журналов могут быть очень большими при настройке высокой детализации, в некоторых конфигурациях превышая гигабайты в неделю. Администраторы должны учитывать требования к дисковому пространству, связанные с конфигурациями аудита с высоким объёмом данных. Например, может быть целесообразно выделить отдельную файловую систему для /var/audit, чтобы другие файловые системы не пострадали, если файловая система аудита переполнится.

20.2. Ключевые термины

Следующие термины связаны с аудитом событий безопасности:

- *событие (event)*: аудируемое событие — это любое событие, которое может быть зарегистрировано с помощью подсистемы аудита. Примерами событий, связанных с безопасностью, являются создание файла, установка сетевого соединения или вход пользователя в систему. События могут быть "приписываемые", то есть их можно отследить до аутентифицированного пользователя, или "неприписываемые". Примерами неприписываемых событий являются любые события, происходящие до аутентификации в процессе входа в систему, например, неудачные попытки ввода пароля.
- *класс (class)*: именованный набор связанных событий, используемых в выражениях выбора. Распространённые классы событий включают "создание файла" (fc), "выполнение" (ex) и "вход/выход" (lo).
- *запись (record)*: запись в журнале аудита, описывающая событие безопасности. Записи содержат тип события, информацию о субъекте (пользователе), выполняющем действие, данные о дате и времени, информацию об объектах или аргументах, а также указание на успешность или неудачу выполнения.
- *журнал (trail)*: файл журнала, состоящий из серии записей аудита, описывающих события безопасности. Записи в журнале расположены в приблизительном хронологическом порядке относительно времени завершения событий. Только авторизованные процессы имеют право добавлять записи в журнал аудита.
- *выражение выбора (selection expression)*: строка, содержащая список префиксов и имён классов событий аудита, используемых для сопоставления событий.
- *предварительный выбор (preselection)*: процесс, в ходе которого система определяет, какие события представляют интерес для администратора. Конфигурация предварительного отбора использует ряд выражений выбора для определения классов событий, подлежащих аудиту для конкретных пользователей, а также глобальные настройки, применяемые как к авторизованным, так и к неавторизованным процессам.
- *фильтрация (reduction)*: процесс выбора записей из существующих журналов аудита для сохранения, печати или анализа. Аналогично, процесс удаления нежелательных записей аудита из журнала. Используя сокращение, администраторы могут реализовать политики сохранения данных аудита. Например, детальные журналы аудита могут храниться в течение одного месяца, но после этого они могут быть сокращены, чтобы сохранить только информацию о входах в систему для архивных целей.

20.3. Настройка аудита

Поддержка аудита событий в пользовательском пространстве устанавливается как часть базовой операционной системы FreeBSD. Поддержка на уровне ядра доступна в ядре GENERIC по умолчанию, и auditd(8) может быть включен добавлением следующей строки в /etc/rc.conf:

```
auditd_enable="YES"
```

Затем запустите демон аудита:

```
# service auditd start
```

Пользователи, предпочитающие компилировать собственное ядро, должны включить следующую строку в файл конфигурации своего ядра:

```
options AUDIT
```

20.3.1. Выражения выбора событий

Выражения выбора используются в нескольких местах конфигурации аудита для определения, какие события должны аудироваться. Выражения содержат список классов событий для сопоставления. Выражения выбора вычисляются слева направо, а два выражения объединяются путем добавления одного к другому.

Классы событий аудита по умолчанию содержит сводку классов событий аудита по умолчанию:

Таблица 34. Классы событий аудита по умолчанию

Имя класса	Описание	Действие
all	all	Совпадает со всеми классами событий.
aa	authentication and authorization	
ad	administrative	Административные действия, выполняемые с системой в целом.
ap	application	Определённое приложением действие.
cl	file close	Аудит вызовов системного вызова `close`.
ex	exec	Аудит выполнения программ. Аудит аргументов командной строки и переменных окружения контролируется через audit_control(5) с использованием параметров `argv` и `envv` в настройке `policy`.

Имя класса	Описание	Действие
fa	file attribute access	Аудит доступа к атрибутам объектов, таким как stat(1) и pathconf(2).
fc	file create	События аудита, в результате которых создаётся файл.
fd	file delete	Аудит событий, в которых происходит удаление файлов.
fm	file attribute modify	События аудита, связанные с изменением атрибутов файлов, например, с помощью chown(8), chflags(1) и flock(2).
fr	file read	События аудита, в которых данные читаются или файлы открываются для чтения.
fw	file write	События аудита, в которых данные записываются или файлы создаются либо изменяются.
io	ioctl	Контроль использования системного вызова `ioctl`.
ip	ipc	Аудит различных форм межпроцессного взаимодействия, включая POSIX-каналы и операции System V IPC.
lo	login_logout	Audit login(1) и logout(1) события.
na	non attributable	Аудит неприписываемых событий.
no	ошибочный класс	Не соответствует ни одному событию аудита.
nt	network	События аудита, связанные с сетевыми действиями, такими как connect(2) и accept(2).
ot	other	Аудит различных событий.
pc	process	Аудит операций процессов, таких как exec(3) и exit(3).

Эти классы событий аудита могут быть настроены путем изменения конфигурационных

файлов audit_class и audit_event.

Каждый класс событий аудита может быть объединён с префиксом, указывающим, соответствуют ли успешные/неудачные операции, и добавляется или удаляется запись для соответствия классу и типу. В Префиксы для классов событий аудита приведены доступные префиксы:

Таблица 35. Префиксы для классов событий аудита

Префикс	Действие
+	Аудит успешных событий в этом классе.
-	Аудит неудачных событий в этом классе.
^	Не аудировать ни успешные, ни неудачные события в этом классе.
^+	Не аудировать успешные события в данном классе.
^-	Не аудировать неудачные события в этом классе.

Если префикс отсутствует, будут аудироваться как успешные, так и неудачные экземпляры события.

Следующая строка выбора выбирает как успешные, так и неудачные события входа/выхода, но только успешные события выполнения:

```
lo,+ex
```

20.3.2. Файлы конфигурации

В каталоге /etc/security находятся следующие файлы конфигурации для аудита событий безопасности:

- audit_class: содержит определения классов аудита.
- audit_control: управляет аспектами подсистемы аудита, такими как классы аудита по умолчанию, минимальное свободное место на томе с журналом аудита и максимальный размер журнала аудита.
- audit_event: текстовые названия и описания системных событий аудита, а также список классов, к которым относится каждое событие.
- audit_user: пользовательские требования аудита, которые объединяются с глобальными настройками по умолчанию при входе в систему.
- audit_warn: настраиваемый сценарий оболочки, используемый auditd(8) для генерации предупреждающих сообщений в исключительных ситуациях, например, когда заканчивается место для записей аудита или когда файл журнала аудита был перезаписан.

Файлы конфигурации аудита следует редактировать и поддерживать тщательно, так как ошибки в конфигурации могут привести к некорректной записи событий.

В большинстве случаев администраторам потребуется изменить только файлы audit_control и audit_user. Первый файл управляет системными настройками и политиками аудита, а второй может использоваться для тонкой настройки аудита по пользователям.

Файл audit_control

Некоторые параметры подсистемы аудита по умолчанию указаны в файле audit_control:

```
dir:/var/audit
dist:off
flags:lo,aa
minfree:5
naflags:lo,aa
policy:cnt,argv
filesz:2M
expire-after:10M
```

Запись `dir` используется для указания одного или нескольких каталогов, в которых будут храниться журналы аудита. Если указано несколько каталогов, они будут использоваться по очереди по мере заполнения. Обычно настраивают аудит так, чтобы журналы хранились на выделенной файловой системе — это предотвращает конфликты между подсистемой аудита и другими подсистемами при заполнении файловой системы.

Если поле `dist` установлено в `on` или `yes`, жёсткие ссылки будут созданы для всех файлов журнала в /var/audit/dist.

Поле `flags` устанавливает системную маску предварительного выбора по умолчанию для учитываемых событий. В приведённом примере аудиту подлежат успешные и неудачные события входа/выхода, а также аутентификация и авторизация для всех пользователей.

Запись `minfree` определяет минимальный процент свободного места в файловой системе, где хранится журнал аудита.

Запись `naflags` определяет классы аудита для событий без атрибутов, таких как процесс входа/выхода, аутентификация и авторизация.

Запись `policy` определяет список флагов политики, разделённых запятыми, которые контролируют различные аспекты поведения аудита. Флаг `cnt` указывает, что система должна продолжать работу, несмотря на сбой аудита (этот флаг настоятельно рекомендуется). Другой флаг, `argv`, приводит к аудиту аргументов командной строки системного вызова execve(2) как части выполнения команды.

`filesz` указывает максимальный размер файла аудита перед автоматическим завершением и ротацией файла. Значение `0` отключает автоматическую ротацию логов. Если запрашиваемый размер файла меньше минимального значения в 512k, он будет

проигнорирован, и будет сгенерировано сообщение в логе.

Поле `expire-after` указывает, когда файлы журналов аудита устареют и будут удалены.

Файл audit_user

Администратор может указать дополнительные требования аудита для конкретных пользователей в файле audit_user. Каждая строка настраивает аудит для пользователя с помощью двух полей: поле `alwaysaudit` определяет набор событий, которые всегда должны аудироваться для пользователя, а поле `neveraudit` определяет набор событий, которые никогда не должны аудироваться для пользователя.

Следующие примеры записей аудита фиксируют события входа/выхода и успешного выполнения команд для пользователя `root`, а также создание файлов и успешное выполнение команд для пользователя `www`. Если используется файл audit_control по умолчанию, запись `lo` для `root` избыточна, и события входа/выхода также будут фиксироваться для `www`.

```
root:lo,+ex:no
www:fc,+ex:no
```

20.4. Работа с журналами аудита

Поскольку записи аудита хранятся в двоичном формате BSM, для их изменения или преобразования в текстовый формат доступны встроенные инструменты. Для преобразования файлов записей в простой текстовый формат используйте `praudit`. Для сокращения файла записей аудита с целью анализа, архивирования или печати используйте `auditreduce`. Эта утилита поддерживает различные параметры выбора, включая тип события, класс события, пользователя, дату или время события, а также путь к файлу или объект, над которым выполнялось действие.

Например, чтобы вывести всё содержимое указанного журнала аудита в виде обычного текста:

```
# praudit /var/audit/AUDITFILE
```

Где *AUDITFILE* — файл журнала аудита для дампа.

Журналы аудита состоят из последовательности записей аудита, сформированных из токенов, которые `praudit` выводит последовательно, по одному на строку. Каждый токен имеет определённый тип, например, `header` (заголовок записи аудита) или `path` (путь к файлу из поиска по имени). Ниже приведён пример события `execve`:

```
header,133,10,execve(2),0,Mon Sep 25 15:58:03 2006, + 384 msec
exec arg,finger,doug
path,/usr/bin/finger
attribute,555,root,wheel,90,24918,104944
```

```
subject,robert,root,wheel,root,wheel,38439,38032,42086,128.232.9.100
return,success,0
trailer,133
```

Этот аудит представляет успешный вызов `execve`, в котором была выполнена команда `finger doug`. Токен `exec arg` содержит обработанную командную строку, переданную оболочкой ядру. Токен `path` содержит путь к исполняемому файлу, найденный ядром. Токен `attribute` описывает бинарный файл и включает режим файла. Токен `subject` хранит идентификатор пользователя для аудита, эффективные идентификаторы пользователя и группы, реальные идентификаторы пользователя и группы, идентификатор процесса, идентификатор сессии, идентификатор порта и адрес входа. Обратите внимание, что аудитный идентификатор пользователя и реальный идентификатор пользователя различаются, так как пользователь `robert` переключился на учётную запись `root` перед выполнением этой команды, но аудит ведется с использованием исходного аутентифицированного пользователя. Токен `return` указывает на успешное выполнение, а `trailer` завершает запись.

Также поддерживается формат вывода XML, и он может быть выбран добавлением опции `-x`.

Поскольку журналы аудита могут быть очень большими, можно выбрать подмножество записей с помощью `auditreduce`. В этом примере выбираются все записи аудита, созданные для пользователя `trhodes`, хранящиеся в AUDITFILE:

```
# auditreduce -u trhodes /var/audit/AUDITFILE | praudit
```

Участники группы `audit` имеют право читать журналы аудита в /var/audit. По умолчанию эта группа пуста, поэтому только пользователь `root` может читать журналы аудита. Пользователи могут быть добавлены в группу `audit` для делегирования прав просмотра аудита. Поскольку возможность отслеживать содержимое журналов аудита даёт значительное представление о поведении пользователей и процессов, рекомендуется делегировать права просмотра аудита с осторожностью.

20.4.1. Мониторинг в реальном времени с использованием потоков аудита

Каналы аудита (audit pipe) являются клонируемыми псевдоустройствами, которые позволяют приложениям получать доступ к потоку записей аудита в реальном времени. В первую очередь это интересно разработчикам систем обнаружения вторжений и мониторинга. Однако устройство канала аудита — это удобный способ для администратора организовать мониторинг в реальном времени без проблем с правами владения файлами аудита или прерывания потока событий из-за ротации логов. Для отслеживания живого потока событий аудита:

```
# praudit /dev/auditpipe
```

По умолчанию узлы устройств каналов аудита доступны только пользователю `root`. Чтобы

сделать их доступными для членов группы `audit`, добавьте правило `devfs` в /etc/devfs.rules:

```
add path 'auditpipe*' mode 0440 group audit
```

За дополнительной информацией о настройке файловой системы devfs см. devfs.rules(5).

Легко создать циклы обратной связи с событиями аудита, когда просмотр каждого события аудита приводит к генерации новых событий аудита. Например, если аудируется весь сетевой ввод-вывод, и `praudit` запускается из сессии SSH, будет создаваться непрерывный поток событий аудита с высокой скоростью, так как каждое выводимое событие будет генерировать новое событие. По этой причине рекомендуется запускать `praudit` на устройстве аудит-канала из сеансов без детального аудита ввода-вывода.

20.4.2. Ротация и сжатие файлов журнала аудита

Журналы аудита создаются ядром и управляются демоном аудита auditd(8). Администраторам не следует пытаться использовать newsyslog.conf(5) или другие инструменты для непосредственной ротации журналов аудита. Вместо этого следует использовать `audit` для остановки аудита, переконфигурации системы аудита и выполнения ротации журналов. Следующая команда заставляет демон аудита создать новый журнал аудита и передать ядру сигнал о переходе на использование нового журнала. Старый журнал будет завершен и переименован, после чего администратор может выполнить с ним необходимые действия:

```
# audit -n
```

Если auditd(8) в данный момент не запущен, эта команда завершится ошибкой и будет выведено сообщение об ошибке.

Добавление следующей строки в /etc/crontab позволит выполнять эту ротацию каждые двенадцать часов:

```
0     */12      *       *       *       root    /usr/sbin/audit -n
```

Изменение вступит в силу после сохранения файла /etc/crontab.

Автоматическая ротация файла журнала аудита на основе размера файла возможна с использованием `filesz` в audit_control, как описано в Файл audit_control.

Поскольку файлы журналов аудита могут становиться очень большими, часто возникает необходимость сжимать или архивировать их после закрытия демоном аудита. Скрипт audit_warn можно использовать для выполнения пользовательских операций при различных событиях, связанных с аудитом, включая корректное завершение журналов при их ротации. Например, следующее можно добавить в /etc/security/audit_warn для сжатия журналов аудита после закрытия:

```
#
# Compress audit trail files on close.
#
if [ "$1" = closefile ]; then
        gzip -9 $2
fi
```

Другие действия по архивированию могут включать копирование файлов журналов на централизованный сервер, удаление старых файлов журналов или сокращение журнала аудита для удаления ненужных записей. Этот скрипт будет выполняться только тогда, когда файлы журналов аудита корректно завершены. Он не будет выполняться для журналов, оставшихся незавершёнными после некорректного завершения работы.

Глава 21. Устройства хранения

21.1. Обзор

Эта глава посвящена использованию дисков и носителей данных в FreeBSD. Сюда входят SCSI- и IDE-диски, CD- и DVD-носители, диски в памяти и USB-устройства хранения данных.

Прочтите эту главу, чтобы узнать:

- Как добавить дополнительные жёсткие диски в систему FreeBSD.
- Как увеличить размер раздела диска в FreeBSD.
- Как настроить FreeBSD для использования USB-накопителей.
- Как использовать CD и DVD носители в системе FreeBSD.
- Как использовать программы резервного копирования, доступные в FreeBSD.
- Как настроить диски в памяти.
- Что такое снимки файловой системы и как их эффективно использовать.
- Как использовать квоты для ограничения использования дискового пространства.
- Как зашифровать диски и раздел подкачки для защиты от злоумышленников.
- Как настроить сеть хранения данных с высокой доступностью.

Прежде чем читать эту главу, необходимо:

- Знать, как конфигурировать и устанавливать новое ядро FreeBSD.

21.2. Добавление дисков

В этом разделе описывается, как добавить новый диск SATA к компьютеру, в котором в настоящее время установлен только один накопитель. Сначала выключите компьютер и установите диск, следуя инструкциям производителей компьютера, контроллера и диска. Перезагрузите систему и войдите в систему как `root`.

Проверьте /var/run/dmesg.boot, чтобы убедиться, что новый диск обнаружен. В этом примере новый SATA-диск будет отображаться как ada1.

Для этого примера на новом диске будет создан один большой раздел. Схема разделов GPT будет использована вместо более старой и менее универсальной схемы MBR.

Если добавляемый диск не пуст, старую информацию о разделах можно удалить с помощью `gpart delete`. Подробности см. в gpart(8).

Создаётся схема разделов, а затем добавляется единственный раздел. Для повышения производительности на новых дисках с большими размерами аппаратных блоков раздел выравнивается по границам одного мегабайта:

```
# gpart create -s GPT ada1
# gpart add -t freebsd-ufs -a 1M ada1
```

В зависимости от использования может потребоваться несколько небольших разделов. См. gpart(8) для вариантов создания разделов меньше целого диска.

Информацию о разделах диска можно просмотреть с помощью `gpart show`:

```
% gpart show ada1
=>        34  1465146988  ada1  GPT  (699G)
          34        2014        - free -  (1.0M)
        2048  1465143296     1  freebsd-ufs  (699G)
  1465145344        1678        - free -  (839K)
```

Создаётся файловая система в новом разделе на новом диске:

```
# newfs -U /dev/ada1p1
```

Создаётся пустой каталог как *точка монтирования* — место для подключения нового диска в файловой системе исходного диска:

```
# mkdir /newdisk
```

Наконец, в файл /etc/fstab добавляется запись, чтобы новый диск автоматически монтировался при загрузке:

```
/dev/ada1p1 /newdisk    ufs rw  2   2
```

Новый диск можно подключить вручную без перезагрузки системы:

```
# mount /newdisk
```

21.3. Изменение размера и увеличение дисков

Емкость диска может быть увеличена без изменения уже имеющихся данных. Это часто происходит с виртуальными машинами, когда виртуальный диск оказывается слишком маленьким и его расширяют. Иногда образ диска записывается на USB-накопитель, но не использует его полную емкость. Здесь мы описываем, как изменить размер или *расширить* содержимое диска, чтобы использовать увеличенную емкость.

Определите имя устройства диска, который нужно изменить, просмотрев /var/run/dmesg.boot. В этом примере в системе только один SATA-диск, поэтому диск будет

отображаться как ada0.

Перечислите разделы на диске, чтобы увидеть текущую конфигурацию:

```
# gpart show ada0
=>      34  83886013  ada0  GPT  (48G) [CORRUPT]
        34       128     1  freebsd-boot  (64k)
       162  79691648     2  freebsd-ufs  (38G)
  79691810   4194236     3  freebsd-swap  (2G)
  83886046         1        - free -  (512B)
```

Если диск был отформатирован с использованием схемы разделов GPT, он может отображаться как "повреждённый", поскольку резервная таблица разделов GPT больше не находится в конце диска. Восстановите резервную таблицу разделов с помощью `gpart`:

```
# gpart recover ada0
ada0 recovered
```

Теперь дополнительное пространство на диске доступно для использования новым разделом или для расширения существующего раздела:

```
# gpart show ada0
=>       34  102399933  ada0  GPT  (48G)
         34        128     1  freebsd-boot  (64k)
        162   79691648     2  freebsd-ufs  (38G)
   79691810    4194236     3  freebsd-swap  (2G)
   83886046   18513921        - free -  (8.8G)
```

Разделы можно изменять в размере только в пределах непрерывного свободного пространства. В данном случае последним разделом на диске является раздел подкачки, но требуется изменить размер второго раздела. Поскольку разделы подкачки содержат только временные данные, их можно безопасно отмонтировать, удалить, а затем заново создать третий раздел после изменения размера второго раздела.

Отключить раздел подкачки:

```
# swapoff /dev/ada0p3
```

Удалите третий раздел, указанный флагом `-i`, с диска *ada0*.

```
# gpart delete -i 3 ada0
ada0p3 deleted
# gpart show ada0
=>       34  102399933  ada0  GPT  (48G)
```

```
        34        128     1  freebsd-boot  (64k)
       162   79691648     2  freebsd-ufs  (38G)
  79691810   22708157        - free -  (10G)
```

Существует риск потери данных при изменении таблицы разделов смонтированной файловой системы. Наилучшим вариантом будет выполнение следующих шагов на размонтированной файловой системе, загрузившись с Live CD-ROM или USB-устройства. Однако, если это крайне необходимо, смонтированную файловую систему можно изменить, отключив защитные механизмы GEOM:

```
# sysctl kern.geom.debugflags=16
```

Измените размер раздела, оставив место для создания раздела подкачки нужного размера. Раздел, который нужно изменить, указывается с помощью `-i`, а новый желаемый размер — с помощью `-s`. Дополнительно выравнивание раздела контролируется с помощью `-a`. Это изменяет только размер раздела. Файловая система в разделе будет расширена в отдельном шаге.

```
# gpart resize -i 2 -s 47G -a 4k ada0
ada0p2 resized
# gpart show ada0
=>       34  102399933  ada0  GPT  (48G)
         34        128     1  freebsd-boot  (64k)
        162   98566144     2  freebsd-ufs  (47G)
   98566306    3833661        - free -  (1.8G)
```

Воссоздайте раздел подкачки и активируйте его. Если размер не указан с помощью `-s`, используется все оставшееся пространство:

```
# gpart add -t freebsd-swap -a 4k ada0
ada0p3 added
# gpart show ada0
=>       34  102399933  ada0  GPT  (48G)
         34        128     1  freebsd-boot  (64k)
        162   98566144     2  freebsd-ufs  (47G)
   98566306    3833661     3  freebsd-swap  (1.8G)
# swapon /dev/ada0p3
```

Увеличьте файловую систему UFS, чтобы использовать новую ёмкость изменённого раздела:

```
# growfs /dev/ada0p2
Device is mounted read-write; resizing will result in temporary write suspension for
/.
```

```
It's strongly recommended to make a backup before growing the file system.
OK to grow file system on /dev/ada0p2, mounted on /, from 38GB to 47GB? [Yes/No] Yes
super-block backups (for fsck -b #) at:
 80781312, 82063552, 83345792, 84628032, 85910272, 87192512, 88474752,
 89756992, 91039232, 92321472, 93603712, 94885952, 96168192, 97450432
```

Если файловая система ZFS, изменение размера запускается выполнением подкоманды `online` с ключом `-e`:

```
# zpool online -e zroot /dev/ada0p2
```

Как раздел, так и файловая система на нём теперь изменены в размере для использования нового доступного пространства на диске.

21.4. USB-накопители

Многие внешние устройства хранения данных, такие как жёсткие диски, USB-флешки, а также устройства для записи CD и DVD, используют универсальную последовательную шину (USB). FreeBSD поддерживает устройства USB 1.x, 2.0 и 3.0.

Поддержка USB 3.0 несовместима с некоторым оборудованием, включая чипсеты Haswell (Lynx point). Если FreeBSD загружается с сообщением `failed with error 19`, отключите xHCI/USB3 в BIOS системы.

Поддержка USB-накопителей встроена в ядро GENERIC. Для собственной сборки ядра убедитесь, что следующие строки присутствуют в конфигурационном файле ядра:

```
device scbus    # SCSI bus (required for ATA/SCSI)
device da   # Direct Access (disks)
device pass # Passthrough device (direct ATA/SCSI access)
device uhci # provides USB 1.x support
device ohci # provides USB 1.x support
device ehci # provides USB 2.0 support
device xhci # provides USB 3.0 support
device usb  # USB Bus (required)
device umass    # Disks/Mass storage - Requires scbus and da
device cd   # needed for CD and DVD burners
```

FreeBSD использует драйвер umass(4), который задействует подсистему SCSI для доступа к USB-устройствам хранения данных. Поскольку любое USB-устройство будет распознаваться системой как SCSI-устройство, если USB-устройство является записывающим CD- или DVD-приводом, *не* включайте `device atapicam` в конфигурационный файл пользовательского ядра.

Оставшаяся часть этого раздела демонстрирует, как убедиться, что USB-накопитель распознаётся FreeBSD, и как настроить устройство для использования.

21.4.1. Настройка устройств

Для проверки конфигурации USB подключите USB-устройство. Используйте `dmesg`, чтобы убедиться, что устройство появилось в системном буфере сообщений. Результат должен выглядеть примерно так:

```
umass0: <STECH Simple Drive, class 0/0, rev 2.00/1.04, addr 3> on usbus0
umass0:  SCSI over Bulk-Only; quirks = 0x0100
umass0:4:0:-1: Attached to scbus4
da0 at umass-sim0 bus 0 scbus4 target 0 lun 0
da0: <STECH Simple Drive 1.04> Fixed Direct Access SCSI-4 device
da0: Serial Number WD-WXE508CAN263
da0: 40.000MB/s transfers
da0: 152627MB (312581808 512 byte sectors: 255H 63S/T 19457C)
da0: quirks=0x2<NO_6_BYTE>
```

Марка, файл устройства (da0), скорость и размер будут отличаться в зависимости от устройства.

Поскольку USB-устройство распознаётся как SCSI, для вывода списка USB-накопителей, подключённых к системе, можно использовать `camcontrol`:

```
# camcontrol devlist
<STECH Simple Drive 1.04>          at scbus4 target 0 lun 0 (pass3,da0)
```

Или можно использовать `usbconfig` для вывода списка устройств. Дополнительную информацию об этой команде смотрите в usbconfig(8).

```
# usbconfig
ugen0.3: <Simple Drive STECH> at usbus0, cfg=0 md=HOST spd=HIGH (480Mbps) pwr=ON (2mA)
```

Если устройство не было отформатировано, обратитесь к разделу Добавление дисков для получения инструкций по форматированию и созданию разделов на USB-накопителе. Если накопитель поставляется с файловой системой, он может быть смонтирован пользователем `root` с помощью инструкций из «Монтирование и размонтирование файловых систем».

Разрешение непривилегированным пользователям монтировать произвольные носители путем включения `vfs.usermount`, как описано ниже, не должно считаться безопасным с точки зрения защиты. Большинство файловых систем не предназначены для защиты от вредоносных устройств.

Чтобы устройство можно было монтировать обычным пользователем, одним из решений является добавление всех пользователей устройства в группу `operator` с помощью pw(8). Затем убедитесь, что группа `operator` имеет права на чтение и запись устройства, добавив следующие строки в /etc/devfs.rules:

```
[localrules=5]
add path 'da*' mode 0660 group operator
```

Если в системе также установлены внутренние SCSI-диски, измените вторую строку следующим образом:

```
add path 'da[3-9]*' mode 0660 group operator
```

Это исключит первые три SCSI-диска (da0 — da2) из принадлежности к группе `operator`. Замените *3* на количество внутренних SCSI-дисков. Дополнительную информацию об этом файле смотрите в devfs.rules(5).

Затем включите набор правил в /etc/rc.conf:

```
devfs_system_ruleset="localrules"
```

Затем настройте систему для разрешения обычным пользователям монтировать файловые системы, добавив следующую строку в /etc/sysctl.conf:

```
vfs.usermount=1
```

Поскольку это вступит в силу только после следующей перезагрузки, используйте `sysctl`, чтобы установить эту переменную сейчас:

```
# sysctl vfs.usermount=1
vfs.usermount: 0 -> 1
```

Последним шагом является создание каталога, в который будет монтироваться файловая система. Этот каталог должен принадлежать пользователю, который будет монтировать файловую систему. Один из способов сделать это — создать подкаталог от имени `root`, принадлежащий этому пользователю, например /mnt/username. В следующем примере замените *username* на имя пользователя, а *usergroup* на основную группу пользователя:

```
# mkdir /mnt/username
# chown username:usergroup /mnt/username
```

Предположим, подключена USB-флешка, и появилось устройство /dev/da0s1. Если устройство отформатировано с файловой системой FAT, пользователь может смонтировать его с помощью:

```
% mount -t msdosfs -o -m=644,-M=755 /dev/da0s1 /mnt/username
```

Прежде чем устройство можно будет отключить, его *необходимо* размонтировать:

```
% umount /mnt/username
```

После удаления устройства в системном буфере сообщений будут отображены сообщения, аналогичные следующим:

```
umass0: at uhub3, port 2, addr 3 (disconnected)
da0 at umass-sim0 bus 0 scbus4 target 0 lun 0
da0: <STECH Simple Drive 1.04> s/n WD-WXE508CAN263          detached
(da0:umass-sim0:0:0:0): Periph destroyed
```

21.4.2. Автомонтирование съемных носителей

USB-устройства могут автоматически монтироваться при раскомментировании этой строки в /etc/auto_master:

```
/media      -media      -nosuid
```

Затем добавьте следующие строки в /etc/devd.conf:

```
notify 100 {
    match "system" "GEOM";
    match "subsystem" "DEV";
    action "/usr/sbin/automount -c";
};
```

Перезагрузите конфигурацию, если autofs(5) и devd(8) уже запущены:

```
# service automount restart
# service devd restart
```

autofs(5) можно настроить для запуска при загрузке, добавив следующую строку в /etc/rc.conf:

```
autofs_enable="YES"
```

autofs(5) требует, чтобы devd(8) был включён, как это и настроено по умолчанию.

Запустите службы немедленно с помощью:

```
# service automount start
# service automountd start
```

```
# service autounmountd start
# service devd start
```

Каждая файловая система, которая может быть автоматически смонтирована, отображается как каталог в /media/. Каталог именуется в соответствии с меткой файловой системы. Если метка отсутствует, каталог именуется в соответствии с устройством.

Файловая система автоматически монтируется при первом доступе и размонтируется после периода неактивности. Автомонтируемые диски также можно размонтировать вручную:

```
# automount -fu
```

Этот механизм обычно используется для карт памяти и USB-флешек. Он может применяться с любыми блочными устройствами, включая оптические приводы или iSCSI LUN.

21.5. Создание и использование CD-носителей

Компакт-диски (CD) обладают рядом особенностей, которые отличают их от обычных дисков. Они спроектированы так, чтобы их можно было читать непрерывно без задержек на перемещение головки между дорожками. Хотя на CD действительно есть дорожки, они обозначают участки данных, предназначенные для непрерывного чтения, а не физическое свойство диска. Файловая система ISO 9660 была разработана для работы с этими различиями.

Коллекция портов FreeBSD предоставляет несколько утилит для записи и копирования аудио- и данных на CD. В этой главе демонстрируется использование нескольких утилит командной строки. Для записи CD с графическим интерфейсом можно установить пакеты или порты sysutils/xcdroast или sysutils/k3b.

21.5.1. Поддерживаемые устройства

Ядро GENERIC обеспечивает поддержку SCSI, USB и устройств чтения и записи ATAPICD. Если используется собственное ядро, параметры, которые должны присутствовать в конфигурационном файле ядра, зависят от типа устройства.

Для SCSI-устройства, записывающего CD или DVD диски, убедитесь, что присутствуют следующие параметры:

```
device scbus    # SCSI bus (required for ATA/SCSI)
device da   # Direct Access (disks)
device pass # Passthrough device (direct ATA/SCSI access)
device cd   # needed for CD and DVD burners
```

Для USB-привода убедитесь, что указаны следующие параметры:

```
device scbus    # SCSI bus (required for ATA/SCSI)
```

```
device da   # Direct Access (disks)
device pass # Passthrough device (direct ATA/SCSI access)
device cd   # needed for CD and DVD burners
device uhci # provides USB 1.x support
device ohci # provides USB 1.x support
device ehci # provides USB 2.0 support
device xhci # provides USB 3.0 support
device usb  # USB Bus (required)
device umass    # Disks/Mass storage - Requires scbus and da
```

Для ATAPI устройств, записывающих CD или DVD диски, убедитесь, что указаны следующие параметры:

```
device ata  # Legacy ATA/SATA controllers
device scbus    # SCSI bus (required for ATA/SCSI)
device pass # Passthrough device (direct ATA/SCSI access)
device cd   # needed for CD and DVD burners
```

В версиях FreeBSD до 10.x эта строка также необходима в конфигурационном файле ядра, если устройство записи является ATAPI-устройством:

```
device atapicam
```

Или этот драйвер можно загрузить при загрузке, добавив следующую строку в файл /boot/loader.conf:

```
atapicam_load="YES"
```

Это потребует перезагрузки системы, так как этот драйвер может быть загружен только во время загрузки.

Чтобы убедиться, что FreeBSD распознает устройство, выполните команду `dmesg` и найдите запись об этом устройстве. В системах до версии 10.x имя устройства в первой строке вывода будет acd0 вместо cd0.

```
% dmesg | grep cd
cd0 at ahcich1 bus 0 scbus1 target 0 lun 0
cd0: <HL-DT-ST DVDRAM GU70N LT20> Removable CD-ROM SCSI-0 device
cd0: Serial Number M3OD3S34152
cd0: 150.000MB/s transfers (SATA 1.x, UDMA6, ATAPI 12bytes, PIO 8192bytes)
cd0: Attempt to query device size failed: NOT READY, Medium not present - tray closed
```

21.5.2. Запись компакт-диска

В FreeBSD для записи компакт-дисков можно использовать `cdrecord`. Эта команда устанавливается с пакетом или портом sysutils/cdrtools.

Хотя `cdrecord` имеет множество опций, базовое использование просто. Укажите имя ISO-файла для записи и, если в системе несколько устройств для записи, укажите имя используемого устройства:

```
# cdrecord dev=device imagefile.iso
```

Чтобы определить имя устройства записывающего привода, используйте `-scanbus`, что может дать результат, подобный следующему:

```
# cdrecord -scanbus
ProDVD-ProBD-Clone 3.00 (amd64-unknown-freebsd10.0) Copyright (C) 1995-2010 Jörg
Schilling
Using libscg version 'schily-0.9'
scsibus0:
        0,0,0     0) 'SEAGATE ' 'ST39236LW       ' '0004' Disk
        0,1,0     1) 'SEAGATE ' 'ST39173W        ' '5958' Disk
        0,2,0     2) *
        0,3,0     3) 'iomega  ' 'jaz 1GB         ' 'J.86' Removable Disk
        0,4,0     4) 'NEC     ' 'CD-ROM DRIVE:466' '1.26' Removable CD-ROM
        0,5,0     5) *
        0,6,0     6) *
        0,7,0     7) *
scsibus1:
        1,0,0   100) *
        1,1,0   101) *
        1,2,0   102) *
        1,3,0   103) *
        1,4,0   104) *
        1,5,0   105) 'YAMAHA  ' 'CRW4260        ' '1.0q' Removable CD-ROM
        1,6,0   106) 'ARTEC   ' 'AM12S          ' '1.06' Scanner
        1,7,0   107) *
```

Найдите запись устройство для записи CD и используйте три числа, разделённые запятыми, в качестве значения для `dev`. В данном случае устройство Yamaha имеет значение `1,5,0`, поэтому правильный ввод для указания этого устройства — `dev=1,5,0`. Обратитесь к руководству `cdrecord` для других способов указания этого значения, а также для получения информации о записи аудиодорожек и управлении скоростью записи.

Или выполните следующую команду, чтобы получить адрес записывающего устройства:

```
# camcontrol devlist
<MATSHITA CDRW/DVD UJDA740 1.00>   at scbus1 target 0 lun 0 (cd0,pass0)
```

Используйте числовые значения для `scbus`, `target` и `lun`. В этом примере `1,0,0` — это имя устройства, которое следует использовать.

21.5.3. Запись данных в файловую систему ISO

Для создания компакт-диска с данными файлы, которые будут составлять дорожки на диске, необходимо подготовить перед записью на CD. В FreeBSD пакет sysutils/cdrtools устанавливает `mkisofs`, который можно использовать для создания файловой системы ISO 9660, представляющей образ дерева каталогов в UNIX® файловой системе. Простейший способ использования — указать имя создаваемого ISO-файла и путь к файлам, которые нужно поместить в файловую систему ISO 9660:

```
# mkisofs -o imagefile.iso /path/to/tree
```

Эта команда сопоставляет имена файлов в указанном пути с именами, соответствующими ограничениям стандартной файловой системы ISO 9660, и исключает файлы, не соответствующие стандарту для файловых систем ISO.

Для преодоления ограничений стандарта доступен ряд опций. В частности, `-R` включает расширения Rock Ridge, распространённые в системах UNIX®, а `-J` включает расширения Joliet, используемые в системах Microsoft®.

Для компакт-дисков, которые будут использоваться только в системах FreeBSD, можно применить `-U` для отключения всех ограничений на имена файлов. При использовании вместе с `-R` создаётся образ файловой системы, идентичный указанному дереву FreeBSD, даже если он нарушает стандарт ISO 9660.

Последняя опция общего назначения — `-b`. Она используется для указания местоположения загрузочного образа при создании загрузочного CD в формате "El Torito". Эта опция принимает аргумент — путь к загрузочному образу относительно корня дерева, записываемого на CD. По умолчанию `mkisofs` создаёт образ ISO в режиме "эмуляции флоппи-диска", поэтому ожидает, что загрузочный образ будет иметь размер ровно 1200, 1440 или 2880 КБ. Некоторые загрузчики, например, используемые на дистрибутивных носителях FreeBSD, не применяют режим эмуляции. В этом случае следует использовать опцию `-no-emul-boot`. Таким образом, если /tmp/myboot содержит загрузочную систему FreeBSD с загрузочным образом в /tmp/myboot/boot/cdboot, то следующая команда создаст /tmp/bootable.iso:

```
# mkisofs -R -no-emul-boot -b boot/cdboot -o /tmp/bootable.iso /tmp/myboot
```

Полученный образ ISO можно подключить как диск в памяти с помощью:

```
# mdconfig -a -t vnode -f /tmp/bootable.iso -u 0
# mount -t cd9660 /dev/md0 /mnt
```

Затем можно убедиться, что /mnt и /tmp/myboot идентичны.

Доступно множество других опций для `mkisofs`, позволяющих точно настроить его поведение. Подробности смотрите в mkisofs(8).

Возможно скопировать компакт-диск с данными в файл образа, функционально эквивалентный файлу образа, созданному с помощью `mkisofs`. Для этого используйте dd, указав имя устройства в качестве входного файла и имя создаваемого ISO в качестве выходного файла:

```
# dd if=/dev/cd0 of=file.iso bs=2048
```

Полученный файл образа можно записать на компакт-диск, как описано в Запись компакт-диска.

21.5.4. Использование компакт-дисков с данными

После записи ISO-образа на компакт-диск его можно смонтировать, указав тип файловой системы, имя устройства с компакт-диском и существующую точку монтирования:

```
# mount -t cd9660 /dev/cd0 /mnt
```

Поскольку `mount` предполагает, что файловая система имеет тип `ufs`, ошибка `Incorrect super block` возникнет, если не указать `-t cd9660` при монтировании компакт-диска с данными.

В то время как любой компакт-диск с данными можно смонтировать таким образом, диски с определёнными расширениями ISO 9660 могут работать неожиданно. Например, диски Joliet хранят все имена файлов в двухбайтовых символах Unicode. Если некоторые неанглийские символы отображаются как знаки вопроса, укажите локальную кодировку с помощью `-C`. Для получения дополнительной информации обратитесь к mount_cd9660(8).

Для выполнения этого преобразования символов с помощью опции `-C` необходимо загрузить модуль ядра cd9660_iconv.ko. Это можно сделать, добавив следующую строку в loader.conf:

```
cd9660_iconv_load="YES"
```

и затем перезагрузить машину, или напрямую загрузить модуль с помощью `kldload`.

Изредка при попытке смонтировать компакт-диск с данными может отображаться сообщение `Device not configured`. Обычно это означает, что привод не обнаружил диск в лотке или что привод не виден на шине. Обнаружение носителя может занять несколько секунд, поэтому следует набраться терпения.

Иногда привод SCSICD может быть пропущен, потому что у него не хватило времени ответить на сброс шины. Чтобы решить эту проблему, можно создать пользовательское

ядро с увеличенной задержкой SCSI по умолчанию. Добавьте следующую опцию в конфигурационный файл собственного ядра и пересоберите ядро, следуя инструкциям в "Сборка и установка собственного ядра":

```
options SCSI_DELAY=15000
```

Это указывает шине SCSI сделать паузу в 15 секунд во время загрузки, чтобы дать CD-приводу максимальный шанс ответить на сброс шины.

Возможно записать файл непосредственно на CD без создания файловой системы ISO 9660. Это называется записью сырых данных на CD, и некоторые люди делают это для целей резервного копирования.

Такой диск нельзя смонтировать как обычный CD с данными. Чтобы извлечь данные, записанные на такой диск, их необходимо прочитать непосредственно с устройства. Например, следующая команда извлечёт сжатый tar-архив со второго CD-устройства в текущий рабочий каталог:

```
# tar xzvf /dev/cd1
```

Для монтирования компакт-диска с данными они должны быть записаны с использованием `mkisofs`.

21.5.5. Копирование аудио-CD

Для копирования аудио-CD извлеките аудиоданные с диска в виде набора файлов, затем запишите эти файлы на чистый CD.

В Копирование аудио-CD описано, как дублировать и записывать аудио-CD. Если версия FreeBSD меньше 10.0 и устройство является ATAPI, необходимо сначала загрузить модуль `atapicam`, следуя инструкциям в Поддерживаемые устройства.

Процедура: Копирование аудио-CD

1. Пакет или порт sysutils/cdrtools устанавливает `cdda2wav`. Эту команду можно использовать для извлечения всех аудиодорожек, при этом каждая дорожка записывается в отдельный WAV-файл в текущем рабочем каталоге:

   ```
   % cdda2wav -vall -B -Owav
   ```

   Имя устройства не нужно указывать, если в системе только одно устройство CD. Обратитесь к руководству `cdda2wav` для получения инструкций по указанию устройства и дополнительной информации о других параметрах этой команды.

2. Используйте `cdrecord` для записи файлов .wav:

```
% cdrecord -v dev=2,0 -dao -useinfo  *.wav
```

Убедитесь, что *2,0* установлено правильно, как описано в Запись CD.

21.6. Создание и использование DVD-носителей

По сравнению с компакт-диском, DVD представляет собой следующее поколение технологии хранения данных на оптических носителях. DVD может вмещать больше данных, чем любой компакт-диск, и является стандартом для издания видео.

Для записываемого DVD можно определить пять физических форматов записи:

- DVD-R: Это первый доступный формат записываемых DVD. Стандарт DVD-R определен DVD Forum. Этот формат поддерживает однократную запись.
- DVD-RW: Это перезаписываемая версия стандарта DVD-R. DVD-RW можно перезаписывать около 1000 раз.
- DVD-RAM: Это перезаписываемый формат, который можно рассматривать как съемный жёсткий диск. Однако, этот носитель не совместим с большинством приводов DVD-ROM и DVD-видеоплееров, так как лишь немногие DVD-рекордеры поддерживают формат DVD-RAM. Дополнительную информацию об использовании DVD-RAM см. в Использование DVD-RAM.
- DVD+RW: Это перезаписываемый формат, определённый альянсом DVD+RW Alliance. DVD+RW можно перезаписывать около 1000 раз.
- DVD+R: Этот формат является однократно записываемой разновидностью формата DVD+RW.

Однослойный записываемый DVD может вместить до 4 700 000 000 байт, что фактически составляет 4,38 ГБ или 4485 МБ, так как 1 килобайт равен 1024 байтам.

Необходимо различать физический носитель и приложение. Например, DVD-Video — это определённая структура файлов, которую можно записать на любой перезаписываемый DVD-носитель, такой как DVD-R, DVD+R или DVD-RW. Перед выбором типа носителя убедитесь, что и записывающее устройство, и проигрыватель DVD-Video поддерживают рассматриваемый носитель.

21.6.1. Конфигурация

Для записи DVD используйте growisofs(1). Эта команда входит в набор утилит sysutils/dvd+rw-tools, которые поддерживают все типы DVD-носителей.

Эти инструменты используют подсистему SCSI для доступа к устройствам, поэтому поддержка ATAPI/CAM должна быть загружена или статически собрана в ядре. Эта поддержка не требуется, если устройство записи использует интерфейс USB. Подробнее о настройке USB-устройств см. в разделе USB-накопители.

Доступ DMA также должен быть включен для устройств ATAPI, добавив следующую строку в /boot/loader.conf:

```
hw.ata.atapi_dma="1"
```

Перед попыткой использования dvd+rw-tools ознакомьтесь с примечаниями о совместимости оборудования.

Для графического интерфейса можно использовать пакет sysutils/k3b, который предоставляет удобный интерфейс к growisofs(1) и многим другим инструментам записи.

21.6.2. Запись данных на DVD

Поскольку growisofs(1) является интерфейсом для mkisofs, он вызывает mkisofs(8) для создания структуры файловой системы и записи на DVD. Это означает, что нет необходимости создавать образ данных перед процессом записи.

Для записи данных из /path/to/data на DVD+R или DVD-R используйте следующую команду:

```
# growisofs -dvd-compat -Z /dev/cd0 -J -R /path/to/data
```

В этом примере `-J -R` передаются в mkisofs(8) для создания файловой системы ISO 9660 с расширениями Joliet и Rock Ridge. Подробности см. в mkisofs(8).

Для начальной записи сессии используется параметр `-Z` как для одиночных, так и для множественных сессий. Замените */dev/cd0* на имя устройства DVD. Использование `-dvd-compat` указывает, что диск будет закрыт и запись нельзя будет дополнять. Это также обеспечивает лучшую совместимость носителя с приводами DVD-ROM.

Для записи предварительно созданного образа, например *imagefile.iso*, используйте:

```
# growisofs -dvd-compat -Z /dev/cd0=imagefile.iso
```

Скорость записи должна определяться и автоматически устанавливаться в зависимости от носителя и используемого привода. Для принудительного задания скорости записи используйте `-speed=`. Примеры использования см. в growisofs(1).

Для поддержки файлов размером более 4,38 ГБ необходимо создать гибридную файловую систему UDF/ISO-9660, передав параметры `-udf -iso-level 3` в mkisofs(8) и все связанные программы, например growisofs(1). Это требуется только при создании ISO-образа или записи файлов непосредственно на диск. Поскольку диск, созданный таким образом, должен монтироваться как файловая система UDF с помощью mount_udf(8), он будет доступен только в операционных системах с поддержкой UDF. В противном случае файлы на диске будут выглядеть повреждёнными.

Чтобы создать ISO-файл такого типа:

```
% mkisofs -R -J -udf -iso-level 3 -o imagefile.iso /path/to/data
```

Для записи файлов непосредственно на диск:

```
# growisofs -dvd-compat -udf -iso-level 3 -Z /dev/cd0 -J -R
/path/to/data
```

Когда ISO-образ уже содержит большие файлы, для записи этого образа на диск с помощью `growisofs` не требуется дополнительных параметров.

Убедитесь, что используется актуальная версия sysutils/cdrtools, которая содержит mkisofs(8), так как более старая версия может не поддерживать большие файлы. Если последняя версия не работает, установите sysutils/cdrtools-devel и ознакомьтесь с его mkisofs(8).

21.6.3. Запись DVD-Video

DVD-Video — это определённая структура файлов, основанная на спецификациях ISO 9660 и микро-UDF (M-UDF). Поскольку DVD-Video представляет собой конкретную иерархию структуры данных, для создания DVD требуется специальная программа, например multimedia/dvdauthor.

Если уже существует образ файловой системы DVD-Video, его можно записать так же, как и любой другой образ. Если для создания DVD использовался `dvdauthor` и результат находится в /path/to/video, то для записи DVD-Video следует использовать следующую команду:

```
# growisofs -Z /dev/cd0 -dvd-video /path/to/video
```

`-dvd-video` передаётся в mkisofs(8), чтобы указать создать файловую систему в формате DVD-Video. Эта опция подразумевает использование опции `-dvd-compat` growisofs(1).

21.6.4. Использование DVD+RW

В отличие от CD-RW, новая DVD+RW требует форматирования перед первым использованием. *Рекомендуется* позволить growisofs(1) автоматически выполнить это, когда это уместно. Однако можно использовать `dvd+rw-format` для форматирования DVD+RW:

```
# dvd+rw-format /dev/cd0
```

Выполняйте эту операцию только один раз и помните, что форматировать нужно только чистые носители DVD+RW. После форматирования DVD+RW можно записывать как обычно.

Для записи совершенно новой файловой системы, а не просто добавления данных на

DVD+RW, не требуется предварительно очищать носитель. Вместо этого можно перезаписать предыдущую запись следующим образом:

```
# growisofs -Z /dev/cd0 -J -R /path/to/newdata
```

Формат DVD+RW поддерживает добавление данных к предыдущей записи. Эта операция заключается в объединении нового сеанса с существующим, так как это не считается многосессионной записью. growisofs(1) будет *расширять* файловую систему ISO 9660, присутствующую на носителе.

Например, чтобы добавить данные на DVD+RW, используйте следующую команду:

```
# growisofs -M /dev/cd0 -J -R /path/to/nextdata
```

Те же параметры `mkisofs(8)`, которые использовались для записи начальной сессии, следует применять при последующих записях.

Используйте `-dvd-compat` для лучшей совместимости носителя с приводами DVD-ROM. При использовании DVD+RW эта опция не предотвращает добавление данных.

Для очистки диска запустите:

```
# growisofs -Z /dev/cd0=/dev/zero
```

21.6.5. Использование DVD-RW

DVD-RW поддерживает два формата дисков: инкрементальный последовательный и с ограниченной перезаписью. По умолчанию диски DVD-RW имеют последовательный формат.

Чистый DVD-RW можно записывать напрямую без форматирования. Однако DVD-RW в последовательном формате, который уже использовался, необходимо очистить перед записью нового начального сеанса.

Для очистки DVD-RW в последовательном режиме:

```
# dvd+rw-format -blank=full /dev/cd0
```

Полное стирание с использованием `-blank=full` займёт около одного часа для однократной записи (1x). Быстрое стирание можно выполнить с помощью `-blank`, если DVD-RW будет записываться в режиме Disk-At-Once (DAO). Для записи DVD-RW в режиме DAO используйте команду:

```
# growisofs -use-the-force-luke=dao -Z /dev/cd0=imagefile.iso
```

Поскольку growisofs(1) автоматически определяет быстро очищенные носители и использует запись DAO, параметр `-use-the-force-luke=dao` не требуется.

Следует использовать режим ограниченной перезаписи для любых DVD-RW, так как этот формат более гибкий по сравнению со стандартным инкрементным последовательным.

Для записи данных на последовательную DVD-RW используйте те же инструкции, что и для других форматов DVD:

```
# growisofs -Z /dev/cd0 -J -R /path/to/data
```

Чтобы добавить данные к предыдущей записи, используйте `-M` с growisofs(1). Однако если данные добавляются на DVD-RW в инкрементальном последовательном режиме, на диске будет создана новая сессия, и в результате получится многосессионный диск.

DVD-RW в формате с ограниченной перезаписью не требует очистки перед созданием новой начальной сессии. Вместо этого перезапишите диск с помощью `-Z`. Также можно расширить существующую файловую систему ISO 9660, записанную на диск, с помощью `-M`. Результатом будет односессионный DVD.

Чтобы перевести DVD-RW в формат ограниченной перезаписи, необходимо использовать следующую команду:

```
# dvd+rw-format /dev/cd0
```

Для возврата к последовательному формату используйте:

```
# dvd+rw-format -blank=full /dev/cd0
```

21.6.6. Многосеансовые диски

Немногие приводы DVD-ROM поддерживают многосеансовые DVD и в большинстве случаев читают только первый сеанс. DVD+R, DVD-R и DVD-RW в последовательном формате могут поддерживать несколько сеансов. Понятие множественных сеансов отсутствует для форматов DVD+RW и DVD-RW с ограниченной перезаписью.

Используя следующую команду после начального незакрытого сеанса на DVD+R, DVD-R или DVD-RW в последовательном формате, можно добавить новый сеанс на диск:

```
# growisofs -M /dev/cd0 -J -R /path/to/nextdata
```

Использование этой команды с DVD+RW или DVD-RW в режиме ограниченной перезаписи приведет к добавлению данных с объединением нового сеанса с существующим. В результате получится диск с одним сеансом. Используйте этот метод для добавления данных после первоначальной записи на таких типах носителей.

Поскольку некоторое пространство на носителе используется между каждым сеансом для отметки конца и начала сеансов, следует добавлять сеансы с большим объёмом данных для оптимизации пространства на носителе. Количество сеансов ограничено 154 для DVD+R, около 2000 для DVD-R и 127 для DVD+R Double Layer.

21.6.7. Для получения дополнительной информации

Для получения дополнительной информации о DVD используйте команду `dvd+rw-mediainfo /dev/cd0`, когда диск находится в указанном приводе.

Дополнительная информация о dvd+rw-tools доступна в growisofs(1), на веб-сайте dvd+rw-tools и в архивах почтовой рассылки cdwrite.

При создании отчёта о проблеме, связанной с использованием dvd+rw-tools, всегда прилагайте вывод команды `dvd+rw-mediainfo`.

21.6.8. Использование DVD-RAM

Записывающие устройства DVD-RAM могут использовать интерфейс SCSI или ATAPI. Для устройств ATAPI необходимо включить доступ DMA, добавив следующую строку в /boot/loader.conf:

```
hw.ata.atapi_dma="1"
```

DVD-RAM можно рассматривать как съемный жёсткий диск. Как и любой другой жёсткий диск, DVD-RAM необходимо отформатировать перед использованием. В этом примере все дисковое пространство будет отформатировано под стандартную файловую систему UFS2:

```
# dd if=/dev/zero of=/dev/acd0 bs=2k count=1
# bsdlabel -Bw acd0
# newfs /dev/acd0
```

Устройство DVD, acd0, должно быть изменено в соответствии с конфигурацией.

После форматирования DVD-RAM его можно подключить как обычный жёсткий диск:

```
# mount /dev/acd0 /mnt
```

После монтирования DVD-RAM будет доступен как для чтения, так и для записи.

21.7. Создание и использование дискет

Этот раздел объясняет, как отформатировать 3,5-дюймовую дискету в FreeBSD.

Процедура: Шаги для форматирования дискеты

Дискету необходимо отформатировать на низком уровне перед использованием. Обычно это делается производителем, но форматирование — хороший способ проверить целостность носителя. Для низкоуровневого форматирования дискеты в FreeBSD используйте fdformat(8). При работе с этой утилитой обратите внимание на сообщения об ошибках, так как они помогают определить, исправен диск или нет.

1. Для форматирования дискеты вставьте новую дискету размером 3,5 дюйма в первый дисковод и выполните команду:

   ```
   # /usr/sbin/fdformat -f 1440 /dev/fd0
   ```

2. После низкоуровневого форматирования диска создайте метку диска, так как она необходима системе для определения размера диска и его геометрии. Поддерживаемые значения геометрии перечислены в /etc/disktab.

   Для записи метки диска используйте bsdlabel(8):

   ```
   # /sbin/bsdlabel -B -w /dev/fd0 fd1440
   ```

3. Дискета теперь готова для высокоуровневого форматирования с файловой системой. Файловая система дискеты может быть UFS или FAT, причём FAT, как правило, является лучшим выбором для дискет.

   Чтобы отформатировать дискету в FAT, выполните:

   ```
   # /sbin/newfs_msdos /dev/fd0
   ```

Диск готов к использованию. Чтобы использовать дискету, смонтируйте её с помощью mount_msdosfs(8). Также можно установить и использовать emulators/mtools из Коллекции портов.

21.8. Основы резервного копирования

Реализация плана резервного копирования необходима для возможности восстановления после выхода диска из строя, случайного удаления файлов, повреждения данных или полного уничтожения машины, включая уничтожение локальных резервных копий.

Тип резервного копирования и его расписание будут варьироваться в зависимости от

важности данных, необходимой детализации для восстановления файлов и допустимого времени простоя. Некоторые возможные методы резервного копирования включают:

- Архивы всей системы, сохранённые на постоянных носителях вне площадки. Это обеспечивает защиту от всех перечисленных выше проблем, но восстановление происходит медленно и неудобно, особенно для непривилегированных пользователей.
- Снимки файловой системы, полезные для восстановления удалённых файлов или предыдущих версий файлов.
- Копии целых файловых систем или дисков, которые синхронизируются с другой системой в сети с использованием пакета net/rsync, запускаемого по расписанию.
- Аппаратный или программный RAID, который минимизирует или исключает простои при отказе диска.

Обычно используется комбинация методов резервного копирования. Например, можно создать расписание для автоматического еженедельного полного резервного копирования системы, которое хранится за пределами основной площадки, и дополнить его ежечасными снимками ZFS. Кроме того, можно вручную создавать резервные копии отдельных каталогов или файлов перед их редактированием или удалением.

В этом разделе описаны некоторые утилиты, которые можно использовать для создания резервных копий и управления ими в системе FreeBSD.

21.8.1. Резервное копирование файловой системы

Традиционные программы UNIX® для резервного копирования файловой системы — это dump(8), который создаёт резервную копию, и restore(8), который восстанавливает данные из резервной копии. Эти утилиты работают на уровне блоков диска, ниже абстракций файлов, ссылок и каталогов, создаваемых файловыми системами. В отличие от другого программного обеспечения для резервного копирования, `dump` создаёт резервную копию всей файловой системы и не может сохранить только часть файловой системы или дерево каталогов, расположенное на нескольких файловых системах. Вместо записи файлов и каталогов `dump` записывает непосредственно блоки данных, из которых состоят файлы и каталоги.

Если `dump` используется для корневого каталога, он не будет создавать резервные копии для /home, /usr и многих других каталогов, так как обычно они являются точками монтирования других файловых систем или символическими ссылками на них.

При восстановлении данных `restore` по умолчанию сохраняет временные файлы в /tmp/. Если используется диск восстановления с малым объёмом /tmp, следует установить переменную `TMPDIR` в каталог с большим свободным пространством для успешного выполнения восстановления.

При использовании `dump` следует учитывать, что некоторые особенности остались с ранних времен версии 6 AT&T UNIX®, примерно 1975 года. Параметры по умолчанию предполагают резервное копирование на 9-дорожечную магнитную ленту, а не на другой тип носителя

или на современные высокоплотные ленты. Эти значения по умолчанию необходимо переопределять в командной строке.

Возможно выполнить резервное копирование файловой системы через сеть на другую систему или на ленточный накопитель, подключённый к другому компьютеру. Хотя для этого можно использовать утилиты rdump(8) и rrestore(8), они не считаются безопасными.

Вместо этого можно более безопасно использовать `dump` и `restore` через SSH-соединение. Этот пример создаёт полную сжатую резервную копию /usr и отправляет её на указанный хост через SSH-соединение.

Пример 26. Использование `dump` через ssh

```
# /sbin/dump -0uan -f - /usr | gzip -2 | ssh -c blowfish \
          targetuser@targetmachine.example.com dd of=/mybigfiles/dump-usr-l0.gz
```

Этот пример устанавливает переменную окружения `RSH` для записи резервной копии на ленточный накопитель в удалённой системе через SSH-соединение:

Пример 27. Использование `dump` через ssh с установленной переменной `RSH`

```
# env RSH=/usr/bin/ssh /sbin/dump -0uan -f
targetuser@targetmachine.example.com:/dev/sa0 /usr
```

Системы, использующие файловую систему Z (ZFS), могут использовать zfs(8) для создания снимков, а также их отправки и получения на удалённые системы или с них.

21.8.2. Резервное копирование каталогов

Некоторые встроенные утилиты могут делать резервное копирование и восстановление указанных файлов и каталогов по мере необходимости.

Хорошим выбором для создания резервной копии всех файлов в каталоге является tar(1). Эта утилита появилась ещё в шестой версии AT&T UNIX® и по умолчанию предполагает рекурсивное резервное копирование на локальное ленточное устройство. С помощью ключей можно указать имя файла для резервной копии.

Этот пример создаёт сжатую резервную копию текущего каталога и сохраняет её в /tmp/mybackup.tgz. При создании резервной копии убедитесь, что она не сохраняется в тот же каталог, который резервируется.

Пример 28. Резервное копирование текущего каталога с помощью `tar`

```
# tar czvf /tmp/mybackup.tgz .
```

Для восстановления всей резервной копии перейдите в каталог, в который нужно восстановить данные, и укажите имя резервной копии. Обратите внимание, что это перезапишет более новые версии файлов в каталоге восстановления. Если есть сомнения, восстановите данные во временный каталог или укажите имя файла внутри резервной копии для восстановления.

Пример 29. Восстановление текущего каталога с помощью `tar`

```
# tar xzvf /tmp/mybackup.tgz
```

Существуют десятки доступных параметров, описанных в tar(1). Эта утилита также поддерживает использование шаблонов исключения для указания, какие файлы не должны включаться при резервном копировании указанного каталога или восстановлении файлов из резервной копии.

Для создания резервной копии с использованием указанного списка файлов и каталогов подходит утилита cpio(1). В отличие от `tar`, `cpio` не умеет обходить дерево каталогов и требует предоставления списка файлов для резервирования.

Например, список файлов можно создать с помощью `ls` или `find`. Этот пример создаёт рекурсивный список текущего каталога, который затем передаётся в `cpio` для создания резервной копии с именем /tmp/mybackup.cpio.

Пример 30. Использование `ls` и `cpio` для создания рекурсивной резервной копии текущего каталога

```
# ls -R | cpio -ovF /tmp/mybackup.cpio
```

Утилита для резервного копирования, которая пытается объединить возможности, предоставляемые `tar` и `cpio`, — это pax(1). С течением времени различные версии `tar` и `cpio` стали немного несовместимыми. POSIX® создал `pax`, который пытается читать и записывать многие из различных форматов `cpio` и `tar`, а также новые собственные форматы.

Эквивалент `pax` для предыдущих примеров будет:

Пример 31. Резервное копирование текущего каталога с помощью `pax`

```
# pax -wf /tmp/mybackup.pax .
```

21.8.3. Использование магнитных лент для резервного копирования

В то время как технология ленточных накопителей продолжает развиваться, современные системы резервного копирования обычно сочетают удалённое резервное копирование с локальными съёмными носителями. FreeBSD поддерживает любые ленточные накопители, использующие SCSI, такие как LTO или DAT. Поддержка SATA и USB ленточных накопителей ограничена.

Для SCSI-ленточных устройств FreeBSD использует драйвер sa(4) и устройства /dev/sa0, /dev/nsa0 и /dev/esa0. Физическое имя устройства — /dev/sa0. При использовании /dev/nsa0 программа резервного копирования не перематывает ленту после записи файла, что позволяет записывать несколько файлов на одну ленту. Использование /dev/esa0 приводит к извлечению ленты после закрытия устройства.

В FreeBSD `mt` используется для управления операциями ленточного накопителя, например, для поиска файлов на ленте или записи управляющих меток на ленту. Например, первые три файла на ленте можно сохранить, пропустив их перед записью нового файла:

```
# mt -f /dev/nsa0 fsf 3
```

Эта утилита поддерживает множество операций. Подробности смотрите в mt(1).

Для записи одного файла на ленту с помощью `tar` укажите имя устройства ленты и файл для резервного копирования:

```
# tar cvf /dev/sa0 file
```

Для восстановления файлов из архива `tar` на ленте в текущий каталог:

```
# tar xvf /dev/sa0
```

Для резервного копирования файловой системы UFS используйте `dump`. В этом примере выполняется резервное копирование /usr без перемотки ленты по завершении:

```
# dump -0aL -b64 -f /dev/nsa0 /usr
```

Для интерактивного восстановления файлов из файла `dump` на ленте в текущий каталог:

```
# restore -i -f /dev/nsa0
```

21.8.4. Сторонние утилиты резервного копирования

Коллекция портов FreeBSD предоставляет множество сторонних утилит, которые можно использовать для планирования создания резервных копий, упрощения резервного

копирования на ленточные накопители, а также для повышения удобства и простоты этого процесса. Многие из этих приложений работают по принципу клиент-сервер и позволяют автоматизировать резервное копирование как отдельной системы, так и всех компьютеров в сети.

Популярные утилиты включают:

- Amanda (misc/amanda-server и misc/amanda-client),
- Bacula (sysutils/bacula13-server и sysutils/bacula13-client),
- Bareos (sysutils/bareos-server и sysutils/bareos-client),
- net/rsync,
- sysutils/duply и
- sysutils/duplicity.

21.8.5. Процедура восстановления при сбое

В дополнение к регулярному резервному копированию рекомендуется выполнить следующие шаги в рамках плана подготовки к чрезвычайным ситуациям.

Создайте печатную копию вывода следующих команд:

- `gpart show`
- `more /etc/fstab`
- `pkg prime-list`
- `dmesg`

Сохраните эту распечатку и копию установочного носителя в надёжном месте. В случае необходимости аварийного восстановления загрузитесь с установочного носителя и выберите `Live CD` для доступа к оболочке-спасателю — режиму аварийного восстановления. Этот режим восстановления можно использовать для просмотра текущего состояния системы и, при необходимости, для переразметки дисков и восстановления данных из резервных копий.

Затем протестируйте аварийную оболочку и резервные копии. Задокументируйте процедуру. Храните эти записи вместе с носителями, распечатками и резервными копиями. Эти заметки могут предотвратить случайное уничтожение резервных копий в стрессовой ситуации во время аварийного восстановления.

Для дополнительной безопасности храните последнюю резервную копию в удалённом месте, физически отделённом от компьютеров и дисков на значительное расстояние.

21.9. Диски в памяти

В дополнение к физическим дискам FreeBSD также поддерживает создание и использование RAM-дисков. Один из возможных вариантов применения RAM-диска — доступ к содержимому файловой системы ISO без необходимости предварительной записи

на CD или DVD с последующим монтированием CD/DVD-носителя.

В FreeBSD драйвер md(4) используется для поддержки дисков в памяти. Ядро GENERIC включает этот драйвер. При использовании пользовательского конфигурационного файла ядра убедитесь, что он содержит следующую строку:

```
device md
```

21.9.1. Присоединение и отсоединение существующих образов

Для подключения существующего образа файловой системы используйте `mdconfig`, указав имя файла ISO и свободный номер устройства. Затем, используя этот номер устройства, подключите его к существующей точке монтирования. После подключения файлы из ISO будут доступны в точке монтирования. В этом примере *diskimage.iso* подключается к устройству в памяти /dev/md0, которое затем монтируется в /mnt:

```
# mdconfig -f diskimage.iso -u 0
# mount -t cd9660 /dev/md0 /mnt
```

Обратите внимание, что `-t cd9660` был использован для монтирования формата ISO. Если номер устройства не указан с помощью `-u`, `mdconfig` автоматически выделит неиспользуемый диск в памяти и выведет имя выделенного устройства, например, md4. Дополнительные сведения о данной команде и её параметрах можно найти в mdconfig(8).

Когда диск в памяти больше не используется, его ресурсы должны быть возвращены обратно системе. Сначала размонтируйте файловую систему, затем используйте `mdconfig` для отключения диска от системы и освобождения его ресурсов. Чтобы продолжить этот пример:

```
# umount /mnt
# mdconfig -d -u 0
```

Чтобы определить, подключены ли к системе какие-либо диски в памяти, введите `mdconfig -l`.

21.9.2. Создание диска в памяти на основе файла или памяти

FreeBSD также поддерживает диски в памяти, где хранилище выделяется либо с жёсткого диска, либо из области памяти. Первый метод обычно называют файловой системой на основе файла, а второй — файловой системой на основе памяти. Оба типа можно создать с помощью `mdconfig`.

Для создания новой файловой системы в памяти укажите тип `swap` и размер создаваемого диска в памяти. Затем отформатируйте диск в памяти файловой системой и смонтируйте его как обычно. В этом примере создаётся диск в памяти размером 5M на устройстве 1. Этот диск в памяти затем форматируется файловой системой UFS перед монтированием:

```
# mdconfig -a -t swap -s 5m -u 1
# newfs -U md1
/dev/md1: 5.0MB (10240 sectors) block size 16384, fragment size 2048
        using 4 cylinder groups of 1.27MB, 81 blks, 192 inodes.
        with soft updates
super-block backups (for fsck -b #) at:
 160, 2752, 5344, 7936
# mount /dev/md1 /mnt
# df /mnt
Filesystem 1K-blocks Used Avail Capacity  Mounted on
/dev/md1        4718    4  4338     0%    /mnt
```

Чтобы создать новый файловый диск в памяти, сначала выделите область на диске для использования. В этом примере создаётся пустой файл размером 5 МБ с именем newimage:

```
# dd if=/dev/zero of=newimage bs=1k count=5k
5120+0 records in
5120+0 records out
```

Затем подключите этот файл к диску в памяти, создайте метку диска и отформатируйте его с файловой системой UFS, смонтируйте диск в памяти и проверьте размер диска на основе файла:

```
# mdconfig -f newimage -u 0
# bsdlabel -w md0 auto
# newfs -U md0a
/dev/md0a: 5.0MB (10224 sectors) block size 16384, fragment size 2048
        using 4 cylinder groups of 1.25MB, 80 blks, 192 inodes.
super-block backups (for fsck -b #) at:
 160, 2720, 5280, 7840
# mount /dev/md0a /mnt
# df /mnt
Filesystem 1K-blocks Used Avail Capacity  Mounted on
/dev/md0a       4710    4  4330     0%    /mnt
```

Для создания файловой системы на основе файла или оперативной памяти с помощью `mdconfig` требуется выполнить несколько команд. В FreeBSD также доступна утилита `mdmfs`, которая автоматически настраивает диск в памяти, форматирует его с файловой системой UFS и монтирует. Например, после создания образа *newimage* с помощью `dd`, следующая команда эквивалентна выполнению команд `bsdlabel`, `newfs` и `mount`, приведённых выше:

```
# mdmfs -F newimage -s 5m md0 /mnt
```

Чтобы вместо этого создать новый диск в памяти на основе оперативной памяти с помощью `mdmfs`, используйте следующую команду:

```
# mdmfs -s 5m md1 /mnt
```

Если номер устройства не указан, `mdmfs` автоматически выберет неиспользуемое устройство памяти. Подробнее о `mdmfs` см. в mdmfs(8).

21.10. Снимки файловой системы

FreeBSD предлагает функцию в сочетании с мягкими обновлениями: создание снимков файловой системы.

Снимки UFS позволяют пользователю создавать образы указанных файловых систем и работать с ними как с файлами. Если вы используете файловую систему Z (ZFS), обратитесь к разделу Управление снимками для получения информации об использовании снимков.

Файлы снимков должны быть созданы в той файловой системе, над которой выполняется действие, и пользователь может создать не более 20 снимков для каждой файловой системы. Активные снимки записываются в суперблок, поэтому они сохраняются после размонтирования и повторного монтирования, а также после перезагрузки системы. Когда снимок больше не нужен, его можно удалить с помощью rm(1). Хотя снимки можно удалять в любом порядке, не все освобождаемое пространство может быть использовано, так как другой снимок может претендовать на часть освобожденных блоков.

Неизменяемый флаг файла `snapshot` устанавливается mksnap_ffs(8) после первоначального создания файла снимка. unlink(1) делает исключение для файлов снимков, так как позволяет их удалять.

Снимки создаются с помощью mount(8). Чтобы создать снимок /var в файле /var/snapshot/snap, используйте следующую команду:

```
# mount -u -o snapshot /var/snapshot/snap /var
```

Или используйте mksnap_ffs(8) для создания снимка:

```
# mksnap_ffs /var /var/snapshot/snap
```

Файлы снимков можно найти в файловой системе, например, в /var, с помощью find(1):

```
# find /var -flags snapshot
```

После создания моментального снимка он может быть использован несколькими способами:

- Некоторые администраторы используют файл снимка для резервного копирования, так как снимок можно перенести на компакт-диски или магнитные ленты.

- Проверка целостности файловой системы, fsck(8), может быть запущена на снимке. При условии, что файловая система была чистой на момент монтирования, результат всегда должен быть чистым и неизменным.
- Запуск dump(8) на снимке создаст дамп-файл, согласованный с файловой системой и временной меткой снимка. dump(8) также может создать снимок, создать образ дампа и затем удалить снимок одной командой, используя опцию `-L`.
- Снимок может быть смонтирован как замороженный образ файловой системы. Для монтирования снимка /var/snapshot/snap выполните команду mount(8):

```
# mdconfig -a -t vnode -o readonly -f /var/snapshot/snap -u 4
# mount -r /dev/md4 /mnt
```

Замороженный /var теперь доступен через /mnt. Все изначально будет находиться в том же состоянии, в котором было на момент создания снимка. Единственное исключение — любые предыдущие снимки будут отображаться как файлы нулевой длины. Чтобы отмонтировать снимок, используйте:

```
# umount /mnt
# mdconfig -d -u 4
```

Для получения дополнительной информации о `softupdates` и снимках файловых систем, включая технические документы, посетите веб-сайт Маршалла Кирка МакКузика по адресу http://www.mckusick.com/.

21.11. Квоты на диске

Дисковые квоты могут использоваться для ограничения объёма дискового пространства или количества файлов, которые пользователь или члены группы могут выделить в рамках одной файловой системы. Это предотвращает ситуацию, когда один пользователь или группа пользователей потребляет всё доступное дисковое пространство.

Этот раздел описывает, как настроить квоты дисков для файловой системы UFS. Для настройки квот в файловой системе ZFS обратитесь к разделу Квоты наборов данных, пользователей и групп

21.11.1. Включение квот на диске

Чтобы определить, поддерживает ли ядро FreeBSD квоты дискового пространства:

```
% sysctl kern.features.ufs_quota
kern.features.ufs_quota: 1
```

В этом примере `1` указывает на поддержку квот. Если значение равно `0`, добавьте следующую строку в файл конфигурации собственного ядра и пересоберите ядро, используя инструкции из Настройка ядра FreeBSD:

```
options QUOTA
```

Далее включите квоты на диски в /etc/rc.conf:

```
quota_enable="YES"
```

Обычно при загрузке проверяется целостность квот для каждой файловой системы с помощью quotacheck(8). Эта программа гарантирует, что данные в базе квот соответствуют данным в файловой системе. Это трудоёмкий процесс, который может значительно увеличить время загрузки системы. Чтобы пропустить этот шаг, добавьте следующую переменную в /etc/rc.conf:

```
check_quotas="NO"
```

Наконец, отредактируйте /etc/fstab, чтобы включить квоты диска для каждой файловой системы. Чтобы включить квоты для пользователей в файловой системе, добавьте `userquota` в поле опций записи /etc/fstab для файловой системы, на которой нужно включить квоты. Например:

```
/dev/da1s2g   /home    ufs rw,userquota 1 2
```

Для включения квот групп используйте `groupquota` вместо этого. Чтобы включить квоты и для пользователей, и для групп, разделите параметры запятой:

```
/dev/da1s2g    /home    ufs rw,userquota,groupquota 1 2
```

По умолчанию файлы квот хранятся в корневом каталоге файловой системы как quota.user и quota.group. Дополнительную информацию можно найти в fstab(5). Указание альтернативного расположения для файлов квот не рекомендуется.

После завершения настройки перезагрузите систему, и /etc/rc автоматически выполнит соответствующие команды для создания начальных файлов квот для всех включённых квот в /etc/fstab.

В обычном режиме работы нет необходимости вручную запускать quotacheck(8), quotaon(8) или quotaoff(8). Однако рекомендуется ознакомиться с их руководствами, чтобы понимать принцип работы.

21.11.2. Установка ограничений квот

Для проверки включения квот выполните:

```
# quota -v
```

Должна быть однострочная сводка об использовании диска и текущих лимитах квот для каждой файловой системы, на которой включены квоты.

Система готова к назначению квот с помощью `edquota`.

Доступно несколько вариантов для установки ограничений на объём дискового пространства, который может быть выделен пользователю или группе, а также на количество создаваемых ими файлов. Ограничения могут быть установлены на основе объёма дискового пространства (блочные квоты), количества файлов (квоты inode) или их комбинации. Каждое ограничение дополнительно разделяется на две категории: жёсткие и мягкие лимиты.

Жёсткий лимит не может быть превышен. Как только пользователь достигает жёсткого лимита, он не может выделить дополнительные ресурсы на этой файловой системе. Например, если у пользователя жёсткий лимит в 500 КБ на файловой системе и он уже использует 490 КБ, он может выделить только дополнительные 10 КБ. Попытка выделить дополнительные 11 КБ завершится неудачей.

Мягкие ограничения могут быть превышены на ограниченное время, известное как льготный период, который по умолчанию составляет одну неделю. Если пользователь превышает своё ограничение дольше льготного периода, мягкое ограничение становится жёстким, и дальнейшие выделения ресурсов запрещаются. Когда пользователь снова опускается ниже мягкого ограничения, льготный период сбрасывается.

В следующем примере редактируется квота для учётной записи `test`. При запуске `edquota` открывается редактор, указанный в переменной `EDITOR`, для изменения ограничений квоты. Редактор по умолчанию установлен в vi.

```
# edquota -u test
Quotas for user test:
/usr: kbytes in use: 65, limits (soft = 50, hard = 75)
        inodes in use: 7, limits (soft = 50, hard = 60)
/usr/var: kbytes in use: 0, limits (soft = 50, hard = 75)
        inodes in use: 0, limits (soft = 50, hard = 60)
```

Обычно для каждой файловой системы с включёнными квотами есть две строки. Одна строка представляет ограничения на блоки, а другая — ограничения на файлы. Измените значение, чтобы изменить лимит квоты. Например, чтобы увеличить лимит блоков для /usr до мягкого лимита `500` и жёсткого лимита `600`, измените значения в этой строке следующим образом:

```
/usr: kbytes in use: 65, limits (soft = 500, hard = 600)
```

Новые ограничения квот вступают в силу после выхода из редактора.

Иногда требуется установить квоты для диапазона пользователей. Это можно сделать, сначала назначив желаемую квоту для одного пользователя, а затем используя опцию `-p` для копирования этой квоты в указанный диапазон идентификаторов пользователей (UID).

Следующая команда скопирует эти квоты для UID с 10,000 по 19,999:

```
# edquota -p test 10000-19999
```

Для получения дополнительной информации обратитесь к edquota(8).

21.11.3. Проверка ограничений квот и использования диска

Для проверки квот и использования диска отдельными пользователями или группами используйте quota(1). Пользователь может просматривать только свою собственную квоту и квоту группы, в которой он состоит. Только суперпользователь может просматривать все квоты пользователей и групп. Чтобы получить сводку по всем квотам и использованию диска для файловых систем с включёнными квотами, используйте repquota(8).

Обычно файловые системы, на которых пользователь не занимает места, не отображаются в выводе команды quota, даже если для пользователя установлено ограничение квоты для этой файловой системы. Используйте -v, чтобы отобразить эти файловые системы. Ниже приведён пример вывода quota -v для пользователя, у которого установлены ограничения квоты на двух файловых системах.

```
Disk quotas for user test (uid 1002):
     Filesystem  usage    quota   limit   grace   files   quota   limit   grace
           /usr      65*     50      75   5days       7      50      60
       /usr/var       0      50      75               0      50      60
```

В этом примере пользователь превысил мягкое ограничение в 50 Кб на /usr на 15 Кб, и у него осталось 5 дней льготного периода. Звёздочка * указывает, что пользователь в настоящее время превысил ограничение квоты.

21.11.4. Квоты по NFS

Квоты применяются подсистемой квот на NFS-сервере. Демон rpc.rquotad(8) предоставляет информацию о квотах для команды quota на NFS-клиентах, позволяя пользователям на этих машинах просматривать свою статистику по квотам.

На NFS-сервере включите rpc.rquotad, удалив # из этой строки в файле **/etc/inetd.conf**:

```
rquotad/1      dgram rpc/udp wait root /usr/libexec/rpc.rquotad rpc.rquotad
```

Затем перезапустите inetd:

```
# service inetd restart
```

21.12. Шифрование разделов диска

FreeBSD обеспечивает отличную защиту от несанкционированного доступа к данным в режиме онлайн. Права доступа к файлам и Принудительный контроль доступа (MAC) помогают предотвратить доступ к данным неавторизованных пользователей, пока операционная система активна и компьютер включен. Однако принудительно устанавливаемые операционной системой права доступа не имеют значения, если злоумышленник получит физический доступ к компьютеру и сможет переместить его жёсткий диск в другую систему для копирования и анализа данных.

Независимо от того, как злоумышленник получил доступ к жёсткому диску или выключённому компьютеру, криптографические подсистемы на основе GEOM, встроенные в FreeBSD, способны защитить данные в файловых системах компьютера даже от высокомотивированных злоумышленников с значительными ресурсами. В отличие от методов шифрования, которые шифруют отдельные файлы, встроенные утилиты `gbde` и `geli` могут использоваться для прозрачного шифрования целых файловых систем. Ни один открытый текст никогда не попадает на пластину жёсткого диска.

Эта глава демонстрирует, как создать зашифрованную файловую систему в FreeBSD. Сначала показан процесс с использованием `gbde`, а затем приведён тот же пример с использованием `geli`.

Подсистема шифрования дисков GBDE устарела и удалена в FreeBSD 15.0.

21.12.1. Шифрование диска с gbde

Целью средства gbde(4) является создание серьёзного препятствия для злоумышленника, пытающегося получить доступ к содержимому *отключённого* устройства хранения данных. Однако, если компьютер скомпрометирован во время работы и устройство хранения активно подключено, или злоумышленник имеет доступ к корректной парольной фразе, это средство не обеспечивает защиты содержимого устройства хранения. Таким образом, важно обеспечивать физическую безопасность системы во время её работы и защищать парольную фразу, используемую механизмом шифрования.

Это средство обеспечивает несколько уровней защиты данных, хранящихся в каждом секторе диска. Оно шифрует содержимое сектора диска с использованием 128-битного AES в режиме CBC. Каждый сектор на диске шифруется с использованием уникального ключа AES. Для получения дополнительной информации о криптографической схеме, включая способ получения ключей секторов из предоставленной пользователем парольной фразы, обратитесь к gbde(4).

FreeBSD предоставляет модуль ядра для gbde, который можно загрузить следующей командой:

```
# kldload geom_bde
```

Если используется пользовательский конфигурационный файл ядра, убедитесь, что он

содержит следующую строку:

```
options GEOM_BDE
```

Следующий пример демонстрирует добавление нового жёсткого диска в систему, который будет содержать единственный зашифрованный раздел, монтируемый в /private.

Процедура: Шифрование раздела с помощью gbde

1. Добавьте новый жёсткий диск

   Установите новый диск в систему, как описано в Добавление дисков. Для целей данного примера новый раздел жёсткого диска добавлен как /dev/ad4s1c, а /dev/ad0s1* представляет существующие стандартные разделы FreeBSD.

   ```
   # ls /dev/ad*
   /dev/ad0        /dev/ad0s1b     /dev/ad0s1e     /dev/ad4s1
   /dev/ad0s1      /dev/ad0s1c     /dev/ad0s1f     /dev/ad4s1c
   /dev/ad0s1a     /dev/ad0s1d     /dev/ad4
   ```

2. Создайте каталог для хранения файлов блокировок gbde

   ```
   # mkdir /etc/gbde
   ```

   Файл блокировки gbde содержит информацию, необходимую gbde для доступа к зашифрованным разделам. Без доступа к файлу блокировки gbde не сможет расшифровать данные, содержащиеся в зашифрованном разделе, без значительного ручного вмешательства, которое не поддерживается программным обеспечением. Каждый зашифрованный раздел использует отдельный файл блокировки.

3. Инициализируйте раздел gbde

   Раздел gbde необходимо инициализировать перед использованием. Эта инициализация выполняется только один раз. Данная команда откроет редактор по умолчанию для настройки различных параметров конфигурации в шаблоне. Для использования с файловой системой UFS установите sector_size в значение 2048:

   ```
   # gbde init /dev/ad4s1c -i -L /etc/gbde/ad4s1c.lock
   #
   # Sector size is the smallest unit of data which can be read or written.
   # Making it too small decreases performance and decreases available space.
   # Making it too large may prevent filesystems from working.  512 is the
   # minimum and always safe.  For UFS, use the fragment size
   #
   sector_size =   2048
   [...]
   ```

   После сохранения изменений пользователю будет предложено дважды ввести

парольную фразу, используемую для защиты данных. Парольная фраза должна быть одинаковой в обоих случаях. Способность gbde защищать данные полностью зависит от качества парольной фразы. Советы по выбору безопасной парольной фразы, которую легко запомнить, можно найти по ссылке http://world.std.com/~reinhold/diceware.htm.

Такая инициализация создаёт файл блокировки для раздела gbde. В данном примере он сохраняется как /etc/gbde/ad4s1c.lock. Файлы блокировки должны иметь расширение ".lock", чтобы корректно определяться скриптом запуска /etc/rc.d/gbde.

Файлы блокировок *обязательно* должны быть включены в резервную копию вместе с содержимым зашифрованных разделов. Без файла блокировки законный владелец не сможет получить доступ к данным на зашифрованном разделе.

4. Присоедините зашифрованный раздел к ядру

```
# gbde attach /dev/ad4s1c -l /etc/gbde/ad4s1c.lock
```

Эта команда запросит ввод парольной фразы, выбранной при инициализации зашифрованного раздела. Новое зашифрованное устройство появится в /dev под именем /dev/имя_устройства.bde:

```
# ls /dev/ad*
/dev/ad0        /dev/ad0s1b     /dev/ad0s1e     /dev/ad4s1
/dev/ad0s1      /dev/ad0s1c     /dev/ad0s1f     /dev/ad4s1c
/dev/ad0s1a     /dev/ad0s1d     /dev/ad4        /dev/ad4s1c.bde
```

5. Создайте файловую систему на зашифрованном устройстве

После подключения зашифрованного устройства к ядру на нём можно создать файловую систему. В этом примере создаётся файловая система UFS с включёнными мягкими обновлениями. Убедитесь, что указан раздел с расширением *.bde:

```
# newfs -U /dev/ad4s1c.bde
```

6. Смонтируйте зашифрованный раздел

Создайте точку монтирования и подключите зашифрованную файловую систему:

```
# mkdir /private
# mount /dev/ad4s1c.bde /private
```

7. Проверьте, что зашифрованная файловая система доступна

Зашифрованная файловая система теперь должна быть видна и доступна для

использования:

```
% df -H
Filesystem         Size   Used  Avail Capacity  Mounted on
/dev/ad0s1a       1037M    72M   883M     8%    /
/devfs             1.0K   1.0K     0B   100%    /dev
/dev/ad0s1f        8.1G    55K   7.5G     0%    /home
/dev/ad0s1e       1037M   1.1M   953M     0%    /tmp
/dev/ad0s1d        6.1G   1.9G   3.7G    35%    /usr
/dev/ad4s1c.bde    150G   4.1K   138G     0%    /private
```

После каждой загрузки все зашифрованные файловые системы должны быть вручную повторно подключены к ядру, проверены на ошибки и смонтированы, прежде чем их можно будет использовать. Чтобы настроить эти шаги, добавьте следующие строки в /etc/rc.conf:

```
gbde_autoattach_all="YES"
gbde_devices="ad4s1c"
gbde_lockdir="/etc/gbde"
```

Для этого необходимо ввести пароль на консоли во время загрузки. После ввода правильного пароля зашифрованный раздел будет автоматически подключен. Дополнительные параметры загрузки gbde доступны и перечислены в rc.conf(5).

`sysinstall` несовместим с устройствами, зашифрованными `gbde`. Все устройства с именами `*.bde` должны быть отключены от ядра перед запуском `sysinstall`, иначе он завершится аварией во время начального сканирования устройств. Чтобы отключить зашифрованное устройство, использованное в примере, выполните следующую команду:

```
# gbde detach /dev/ad4s1c
```

21.12.2. Шифрование диска с помощью `geli`

Альтернативный криптографический класс GEOM доступен с использованием `geli`. Эта утилита управления предоставляет дополнительные возможности и использует другую схему для выполнения криптографических операций. Она обеспечивает следующие функции:

- Использует фреймворк crypto(9) и автоматически задействует криптографическое оборудование, когда оно доступно.
- Поддерживает несколько криптографических алгоритмов, таких как AES-XTS, AES-CBC и Camellia-CBC.
- Позволяет зашифровать корневой раздел. Пароль для доступа к зашифрованному корневому разделу будет запрашиваться при загрузке системы.

- Позволяет использование двух независимых ключей.
- Она быстрая, так как выполняет простое поблочное шифрование.
- Позволяет создавать резервные копии и восстанавливать мастер-ключи. Если пользователь уничтожит свои ключи, доступ к данным всё ещё можно получить, восстановив ключи из резервной копии.
- Позволяет подключить диск с одноразовым случайным ключом, что полезно для разделов подкачки и временных файловых систем.

Дополнительные возможности и примеры использования приведены в geli(8).

Следующий пример описывает, как сгенерировать ключевой файл, который будет использоваться как часть мастер-ключа для зашифрованного провайдера, монтируемого в /private. Ключевой файл предоставит случайные данные, используемые для шифрования мастер-ключа. Мастер-ключ также будет защищён парольной фразой. Размер сектора провайдера составит 4 КБ. В примере описано, как подключиться к провайдеру `geli`, создать на нём файловую систему, смонтировать её, работать с ней и, наконец, отключить её.

Процедура: Шифрование раздела с помощью `geli`

1. Загрузите поддержку `geli`

   Поддержка `geli` доступна в виде загружаемого модуля ядра. Чтобы настроить систему для автоматической загрузки модуля при загрузке, добавьте следующую строку в файл /boot/loader.conf:

   ```
   geom_eli_load="YES"
   ```

   Чтобы загрузить модуль ядра сейчас:

   ```
   # kldload geom_eli
   ```

   Для собственного ядра убедитесь, что файл конфигурации ядра содержит следующие строки:

   ```
   options GEOM_ELI
   device crypto
   ```

2. Сгенерируйте мастер-ключа

   Следующие команды создают мастер-ключ, которым будут зашифрованы все данные. Этот ключ нельзя изменить. Вместо его прямого использования, он шифруется одним или несколькими пользовательскими ключами. Пользовательские ключи формируются из опциональной комбинации случайных байтов из файла /root/da2.key и/или парольной фразы. В данном случае источником данных для ключевого файла является /dev/random. Эта команда также устанавливает размер сектора провайдера (/dev/da2.eli) равным 4 КБ для улучшения производительности:

```
# dd if=/dev/random of=/root/da2.key bs=64 count=1
# geli init -K /root/da2.key -s 4096 /dev/da2
Enter new passphrase:
Reenter new passphrase:
```

Не обязательно использовать и парольную фразу, и файл ключа, так как каждый из этих методов защиты главного ключа может применяться отдельно.

Если файл ключа указан как "-", будет использован стандартный ввод. Например, следующая команда генерирует три файла ключей:

```
# cat keyfile1 keyfile2 keyfile3 | geli init -K - /dev/da2
```

3. Присоедините поставщика с сгенерированным Ключом

   Для подключения провайдера укажите файл ключа, имя диска и парольную фразу:

   ```
   # geli attach -k /root/da2.key /dev/da2
   Enter passphrase:
   ```

   Это создаёт новое устройство с расширением .eli:

   ```
   # ls /dev/da2*
   /dev/da2  /dev/da2.eli
   ```

4. Создайте новую файловую систему

   Далее отформатируйте устройство с файловой системой UFS и смонтируйте его в существующей точке монтирования:

   ```
   # dd if=/dev/random of=/dev/da2.eli bs=1m
   # newfs /dev/da2.eli
   # mount /dev/da2.eli /private
   ```

   Зашифрованная файловая система теперь должна быть доступна для использования:

   ```
   # df -H
   Filesystem     Size   Used  Avail Capacity  Mounted on
   /dev/ad0s1a    248M    89M   139M    38%    /
   /devfs         1.0K   1.0K     0B   100%    /dev
   /dev/ad0s1f    7.7G   2.3G   4.9G    32%    /usr
   /dev/ad0s1d    989M   1.5M   909M     0%    /tmp
   /dev/ad0s1e    3.9G   1.3G   2.3G    35%    /var
   /dev/da2.eli   150G   4.1K   138G     0%    /private
   ```

После завершения работы с зашифрованным разделом и когда раздел /private больше не нужен, рекомендуется перевести устройство в холодное хранилище, размонтировав и отключив зашифрованный раздел `geli` от ядра:

```
# umount /private
# geli detach da2.eli
```

Для упрощения монтирования зашифрованных устройств `geli` во время загрузки предоставляется скрипт rc.d. Для данного примера добавьте следующие строки в /etc/rc.conf:

```
geli_devices="da2"
geli_da2_flags="-k /root/da2.key"
```

В этом примере /dev/da2 настраивается как провайдер `geli` с мастер-ключом /root/da2.key. Система автоматически отключит провайдер от ядра перед завершением работы. Во время загрузки скрипт запросит парольную фразу перед подключением провайдера. До или после запроса пароля могут отображаться другие сообщения ядра. Если процесс загрузки кажется зависшим, внимательно поищите запрос пароля среди других сообщений. После ввода правильной парольной фразы провайдер будет подключен. Файловая система затем монтируется, обычно с помощью записи в /etc/fstab. Инструкции по настройке автоматического монтирования файловой системы при загрузке можно найти в “Монтирование и размонтирование файловых систем”.

21.13. Шифрование раздела подкачки

Как и шифрование разделов диска, шифрование раздела подкачки используется для защиты конфиденциальной информации. Рассмотрим приложение, работающее с паролями. Пока пароли находятся в физической памяти, они не записываются на диск и будут удалены после перезагрузки. Однако если FreeBSD начнёт выгружать страницы памяти для освобождения места, пароли могут быть записаны на диск в незашифрованном виде. Решением в этом случае может быть шифрование раздела подкачки.

В этой части показано, как настроить зашифрованный раздел подкачки с использованием шифрования gbde(8) или geli(8). Предполагается, что раздел подкачки — это /dev/ada0s1b.

21.13.1. Настройка зашифрованного раздела подкачки

Разделы подкачки по умолчанию не шифруются, и перед продолжением работы следует удалить из них все конфиденциальные данные. Чтобы перезаписать текущий раздел подкачки случайными данными, выполните следующую команду:

```
# dd if=/dev/random of=/dev/ada0s1b bs=1m
```

Для шифрования раздела подкачки с помощью gbde(8) добавьте суффикс `.bde` к строке раздела подкачки в /etc/fstab:

```
# Device        Mountpoint  FStype  Options     Dump    Pass#
/dev/ada0s1b.bde    none        swap    sw      0   0
```

Для шифрования раздела подкачки с помощью geli(8) инициализируйте устройство с одноразовым ключём:

```
# geom eli onetime -d /dev/ada0s1b
```

После этого добавьте в /etc/fstab строку для устройства с суффиксом `.eli`:

```
# Device        Mountpoint  FStype  Options     Dump    Pass#
/dev/ada0s1b.eli    none        swap    sw      0   0
```

По умолчанию geli(8) использует алгоритм AES с длиной ключа 128 бит. Обычно стандартных настроек достаточно. При необходимости эти значения по умолчанию можно изменить в поле options файла /etc/fstab. Доступные флаги:

aalgo

Алгоритм проверки целостности данных, используемый для обеспечения отсутствия изменений в зашифрованных данных. Список поддерживаемых алгоритмов приведён в geli(8).

ealgo

Алгоритм шифрования, используемый для защиты данных. Список поддерживаемых алгоритмов приведён в geli(8).

keylen

Длина ключа, используемого для алгоритма шифрования. Подробнее о поддерживаемых длинах ключей для каждого алгоритма шифрования смотрите в geli(8).

sectorsize

Размер блоков, на которые разбиваются данные перед шифрованием. Увеличение размера секторов повышает производительность за счет большего расхода пространства. Рекомендуемый размер — 4096 байт.

В этом примере настраивается зашифрованный раздел подкачки с использованием алгоритма AES-XTS с длиной ключа 128 бит и размером сектора 4 килобайта:

```
# Device        Mountpoint  FStype  Options             Dump    Pass#
/dev/ada0s1b.eli    none        swap    sw,ealgo=AES-XTS,keylen=128,sectorsize=4096 0
0
```

21.13.2. Проверка зашифрованного раздела подкачки

После перезагрузки системы корректную работу зашифрованного раздела подкачки можно проверить с помощью `swapinfo`.

Если используется gbde(8):

```
% swapinfo
Device          1K-blocks     Used    Avail Capacity
/dev/ada0s1b.bde   542720        0   542720     0
```

Если используется geli(8):

```
% swapinfo
Device          1K-blocks     Used    Avail Capacity
/dev/ada0s1b.eli   542720        0   542720     0
```

21.14. Высокодоступное хранилище (HAST)

Высокая доступность — одно из основных требований для серьёзных бизнес-приложений, а высокодоступное хранилище является ключевым компонентом в таких средах. В FreeBSD фреймворк Highly Available STorage (HAST) обеспечивает прозрачное хранение одних и тех же данных на нескольких физически разделённых машинах, соединённых через сеть TCP/IP. HAST можно рассматривать как сетевой RAID1 (зеркало), аналогичный системе хранения DRBD®, используемой на платформе GNU/Linux®. В сочетании с другими функциями высокой доступности FreeBSD, такими как CARP, HAST позволяет создавать высокодоступные кластеры хранения, устойчивые к аппаратным сбоям.

Основные возможности HAST:

- Может использоваться для маскировки ошибок ввода-вывода на локальных жёстких дисках.
- Файлово-системно агностичен, так как работает с любой файловой системой, поддерживаемой FreeBSD.
- Эффективная и быстрая повторная синхронизация, так как синхронизируются только блоки, изменённые во время простоя узла.
- Может использоваться в уже развернутой среде для добавления дополнительной избыточности.
- Вместе с CARP, Heartbeat или другими инструментами он может использоваться для создания надёжной и отказоустойчивой системы хранения данных.

Прочтите эту главу, чтобы узнать:

- Что такое HAST, как он работает и какие возможности предоставляет.
- Как настроить и использовать HAST в FreeBSD.

- Как интегрировать CARP и devd(8) для создания надёжной системы хранения данных.

Прежде чем читать этот раздел, вы должны:

- Понимать основы UNIX® и FreeBSD (Основы FreeBSD).
- Знать, как настраивать сетевые интерфейсы и другие основные подсистемы FreeBSD (Настройка и оптимизация).
- Хорошо разбираться в сетевых возможностях FreeBSD (Сетевое взаимодействие).

Проект HAST был поддержан Фондом FreeBSD при участии http://www.omc.net/ и http://www.transip.nl/.

21.14.1. Работа HAST

HAST обеспечивает синхронную репликацию на блочном уровне между двумя физическими машинами: *primary* (основной) узел и *secondary* (вторичный) узел. Вместе эти две машины называются кластером.

Поскольку HAST работает в конфигурации "основной-вторичный", он позволяет только одному узлу кластера быть активным в любой момент времени. Основной узел, также называемый *активным*, обрабатывает все запросы ввода-вывода для устройств, управляемых HAST. Вторичный узел автоматически синхронизируется с основным.

Физические компоненты системы HAST включают локальный диск на основном узле и диск на удалённом, резервном узле.

HAST работает синхронно на блочном уровне, что делает его прозрачным для файловых систем и приложений. HAST предоставляет обычные GEOM-провайдеры в /dev/hast/ для использования другими инструментами или приложениями. Нет разницы между использованием устройств, предоставляемых HAST, и использованием обычных дисков или разделов.

Каждая операция записи, удаления или сброса данных отправляется как на локальный диск, так и на удалённый диск через TCP/IP. Каждая операция чтения выполняется с локального диска, если только локальный диск не содержит актуальных данных или не возникает ошибка ввода-вывода. В таких случаях операция чтения отправляется на вторичный узел.

HAST стремится обеспечить быстрое восстановление после сбоев. По этой причине важно сократить время синхронизации после отказа узла. Для быстрой синхронизации HAST использует битовую карту грязных экстентов на диске и синхронизирует только их в процессе обычной синхронизации, за исключением начальной синхронизации.

Существует множество способов обработки синхронизации. HAST реализует несколько режимов репликации для работы с различными методами синхронизации:

- *memsync*: В этом режиме операция записи считается завершённой, когда локальная операция записи завершена и когда удалённый узел подтверждает получение данных, но до фактического сохранения данных. Данные на удалённом узле будут сохранены

сразу после отправки подтверждения. Этот режим предназначен для уменьшения задержки, но при этом обеспечивает хорошую надёжность. Этот режим используется по умолчанию.

- *fullsync*: В этом режиме операция записи считается завершённой, когда завершается как локальная, так и удалённая запись. Это самый безопасный и самый медленный режим репликации.
- *async*: В этом режиме операция записи считается завершённой, как только завершается локальная запись. Это самый быстрый и самый опасный режим репликации. Он должен использоваться только при репликации на удалённый узел, где задержка слишком высока для других режимов.

21.14.2. Конфигурация HAST

Фреймворк HAST состоит из нескольких компонентов:

- Демон hastd(8), который обеспечивает синхронизацию данных. При запуске этого демона он автоматически загружает модуль `geom_gate.ko`.
- Служебная программа управления пользовательским пространством hastctl(8).
- Файл конфигурации hast.conf(5). Этот файл должен существовать до запуска hastd.

Пользователи, которые предпочитают статически встраивать поддержку `GEOM_GATE` в ядро, должны добавить следующую строку в файл конфигурации собственного ядра, а затем пересобрать ядро, следуя инструкциям в Настройка ядра FreeBSD:

```
options GEOM_GATE
```

Следующий пример описывает настройку двух узлов в режиме первичный-вторичный с использованием HAST для репликации данных между ними. Узлы будут называться `hasta` с IP-адресом `172.16.0.1` и `hastb` с IP-адресом `172.16.0.2`. Оба узла будут иметь выделенный жёсткий диск /dev/ad6 одинакового размера для работы с HAST. Пул HAST, иногда называемый ресурсом или провайдером GEOM в /dev/hast/, будет называться `test`.

Настройка HAST выполняется с помощью файла /etc/hast.conf. Этот файл должен быть идентичным на обоих узлах. Простейшая конфигурация выглядит следующим образом:

```
resource test {
    on hasta {
        local /dev/ad6
        remote 172.16.0.2
    }
    on hastb {
        local /dev/ad6
        remote 172.16.0.1
    }
}
```

Для более сложной настройки обратитесь к hast.conf(5).

Также можно использовать имена хостов в операторах `remote`, если хосты разрешаемы и определены либо в /etc/hosts, либо в локальном DNS.

После создания конфигурации на обоих узлах можно создать пул HAST. Выполните следующие команды на обоих узлах, чтобы разместить начальные метаданные на локальном диске и запустить hastd(8):

```
# hastctl create test
# service hastd onestart
```

Невозможно использовать провайдеры GEOM с существующей файловой системой или преобразовать существующее хранилище в пул под управлением HAST. Эта процедура требует хранения некоторых метаданных на провайдере, и на существующем провайдере не будет достаточно необходимого пространства.

Роль `primary` или `secondary` узла HAST выбирается администратором или программным обеспечением, таким как Heartbeat, с помощью hastctl(8). На основном узле `hasta` выполните следующую команду:

```
# hastctl role primary test
```

Выполните эту команду на дополнительном узле, `hastb`:

```
# hastctl role secondary test
```

Проверьте результат, выполнив `hastctl` на каждом узле:

```
# hastctl status test
```

Проверьте строку `status` в выводе. Если там указано `degraded`, значит, с файлом конфигурации что-то не так. На каждом узле должно быть указано `complete`, что означает начало синхронизации между узлами. Синхронизация завершается, когда `hastctl status` сообщает о 0 байтах в `dirty` экстентах.

Следующий шаг — создать файловую систему на провайдере GEOM и смонтировать её. Это должно быть выполнено на узле `primary`. Создание файловой системы может занять несколько минут в зависимости от размера жёсткого диска. В этом примере создаётся файловая система UFS на /dev/hast/test:

```
# newfs -U /dev/hast/test
# mkdir /hast/test
```

```
# mount /dev/hast/test /hast/test
```

После правильной настройки структуры HAST последним шагом является обеспечение автоматического запуска HAST во время загрузки системы. Добавьте следующую строку в /etc/rc.conf:

```
hastd_enable="YES"
```

Конфигурация отказоустойчивости

Цель данного примера — создать надёжную систему хранения, устойчивую к отказу любого узла. Если основной узел выходит из строя, резервный узел готов взять на себя управление без перерывов, проверить и смонтировать файловую систему, продолжив работу без потери данных.

Для выполнения этой задачи используется Протокол избыточности общих адресов (CARP — Common Address Redundancy Protocol), который обеспечивает автоматическое переключение на резервный узел на IP-уровне. CARP позволяет нескольким узлам в одном сетевом сегменте совместно использовать один IP-адрес. Настройте CARP на обоих узлах кластера в соответствии с документацией, доступной в “Common Address Redundancy Protocol (CARP)”. В этом примере каждый узел будет иметь свой собственный управляющий IP-адрес и общий IP-адрес *172.16.0.254*. Основной узел HAST в кластере должен быть основным узлом CARP.

Созданный в предыдущем разделе пул HAST теперь готов к экспорту на другие узлы в сети. Это можно осуществить, экспортировав его через NFS или Samba, используя общий IP-адрес *172.16.0.254*. Единственная оставшаяся нерешенной проблема — это автоматический переход на резервный узел в случае отказа основного.

В случае перехода интерфейсов CARP в состояние "включен" или "выключен", операционная система FreeBSD генерирует событие devd(8), что позволяет отслеживать изменения состояния интерфейсов CARP. Изменение состояния интерфейса CARP указывает на то, что один из узлов вышел из строя или вернулся в онлайн. Эти события изменения состояния позволяют запускать скрипт для автоматической обработки переключения при отказе в HAST.

Для отслеживания изменений состояния на интерфейсах CARP добавьте следующую конфигурацию в /etc/devd.conf на каждом узле, заменив `<vhid>` на идентификатор виртуального хоста и `<ifname>` на имя соответствующего интерфейса:

```
notify 30 {
    match "system" "CARP";
    match "subsystem" "<vhid>@<ifname>";
    match "type" "MASTER";
    action "/usr/local/sbin/carp-hast-switch primary";
};
```

```
notify 30 {
    match "system" "CARP";
    match "subsystem" "<vhid>@<ifname>";
    match "type" "BACKUP";
    action "/usr/local/sbin/carp-hast-switch secondary";
};
```

Перезапустите devd(8) на обоих узлах, чтобы новая конфигурация вступила в силу:

```
# service devd restart
```

Когда состояние указанного интерфейса изменяется (переход вверх или вниз), система генерирует уведомление, позволяющее подсистеме devd(8) запустить указанный скрипт автоматического переключения /usr/local/sbin/carp-hast-switch. Для дополнительных пояснений о данной конфигурации обратитесь к devd.conf(5).

Вот пример скрипта автоматического переключения при отказе:

```
#!/bin/sh

# Original script by Freddie Cash <fjwcash@gmail.com>
# Modified by Michael W. Lucas <mwlucas@BlackHelicopters.org>
# and Viktor Petersson <vpetersson@wireload.net>

# The names of the HAST resources, as listed in /etc/hast.conf
resources="test"

# delay in mounting HAST resource after becoming primary
# make your best guess
delay=3

# logging
log="local0.debug"
name="carp-hast"

# end of user configurable stuff

case "$1" in
    primary)
        logger -p $log -t $name "Switching to primary provider for ${resources}."
        sleep ${delay}

        # Wait for any "hastd secondary" processes to stop
        for disk in ${resources}; do
            while $( pgrep -lf "hastd: ${disk} \(secondary\)" > /dev/null 2>&1 ); do
                sleep 1
            done

            # Switch role for each disk
```

```
            hastctl role primary ${disk}
            if [ $? -ne 0 ]; then
                logger -p $log -t $name "Unable to change role to primary for resource
${disk}."
                exit 1
            fi
        done

        # Wait for the /dev/hast/* devices to appear
        for disk in ${resources}; do
            for I in $( jot 60 ); do
                [ -c "/dev/hast/${disk}" ] && break
                sleep 0.5
            done

            if [ ! -c "/dev/hast/${disk}" ]; then
                logger -p $log -t $name "GEOM provider /dev/hast/${disk} did not
appear."
                exit 1
            fi
        done

        logger -p $log -t $name "Role for HAST resources ${resources} switched to
primary."

        logger -p $log -t $name "Mounting disks."
        for disk in ${resources}; do
            mkdir -p /hast/${disk}
            fsck -p -y -t ufs /dev/hast/${disk}
            mount /dev/hast/${disk} /hast/${disk}
        done

    ;;

    secondary)
        logger -p $log -t $name "Switching to secondary provider for ${resources}."

        # Switch roles for the HAST resources
        for disk in ${resources}; do
            if ! mount | grep -q "^/dev/hast/${disk} on "
            then
            else
                umount -f /hast/${disk}
            fi
            sleep $delay
            hastctl role secondary ${disk} 2>&1
            if [ $? -ne 0 ]; then
                logger -p $log -t $name "Unable to switch role to secondary for
resource ${disk}."
                exit 1
            fi
```

```
            logger -p $log -t $name "Role switched to secondary for resource ${disk}."
        done
    ;;
esac
```

В двух словах, скрипт выполняет следующие действия, когда узел становится основным:

- Переводит пул HAST в primary на другом узле.
- Проверяет файловую систему в пуле HAST.
- Подключает пул.

Когда узел становится вторичным:

- Размонтирует пул HAST.
- Переводит пул HAST в состояние secondary.

Это просто пример скрипта, который служит доказательством концепции. Он не обрабатывает все возможные сценарии и может быть расширен или изменён любым способом, например, для запуска или остановки необходимых служб.

Для этого примера использовалась стандартная файловая система UFS. Чтобы сократить время, необходимое для восстановления, можно использовать журналируемую UFS или файловую систему ZFS.

Вместо использования высокодоступного хранилища локально, его также можно предоставить в общее пользование другим компьютерам в сети через NFS, iSCSI, sshfs(1) или программы из портов (например, net/samba419).

Более подробная информация с дополнительными примерами доступна по адресу http://wiki.FreeBSD.org/HAST.

21.14.3. Устранение неполадок

HAST, как правило, должен работать без проблем. Однако, как и с любым другим программным продуктом, могут возникнуть ситуации, когда он работает не так, как предполагается. Источники проблем могут быть разными, но главное правило — обеспечить синхронизацию времени между узлами кластера.

При устранении неполадок HAST уровень отладки hastd(8) следует повысить, запустив `hastd` с параметром `-d`. Этот аргумент можно указать несколько раз для дальнейшего повышения уровня отладки. Также рекомендуется использовать `-F`, что запускает `hastd` в foreground.

Восстановление после раскола кластера

Раскол (split-brain) возникает, когда узлы кластера не могут связаться друг с другом, и оба настроены как первичные. Это опасная ситуация, так как она позволяет обоим узлам вносить противоречивые изменения в данные. Данная проблема должна быть устранена

вручную системным администратором.

Администратор должен либо определить, на каком узле находятся более важные изменения, либо выполнить слияние вручную. Затем следует позволить HAST выполнить полную синхронизацию узла с повреждёнными данными. Для этого выполните следующие команды на узле, который требует повторной синхронизации:

```
# hastctl role init test
# hastctl create test
# hastctl role secondary test
```

Глава 22. GEOM: Модульная инфраструктура трансформации дискового пространства

22.1. Обзор

В FreeBSD система GEOM обеспечивает доступ и управление классами, такими как главная загрузочная запись (MBR) и метки BSD, через использование провайдеров или дисковых устройств в /dev. Поддерживая различные конфигурации программного RAID, GEOM прозрачно предоставляет доступ операционной системе и системным утилитам.

В этой главе рассматривается использование дисков в рамках системы GEOM в FreeBSD. Это включает основные утилиты управления RAID, которые используют данную систему для настройки. Данная глава не является исчерпывающим руководством по конфигурациям RAID и рассматривает только поддерживаемые GEOM классификации RAID.

Прочтите эту главу, чтобы узнать:

- Какой тип поддержки RAID доступен через GEOM.
- Как использовать базовые утилиты для настройки, обслуживания и управления различными уровнями RAID.
- Как зеркалировать, чередовать, шифровать и удалённо подключать дисковые устройства с помощью GEOM.
- Как устранять неполадки с дисками, подключёнными к системе GEOM.

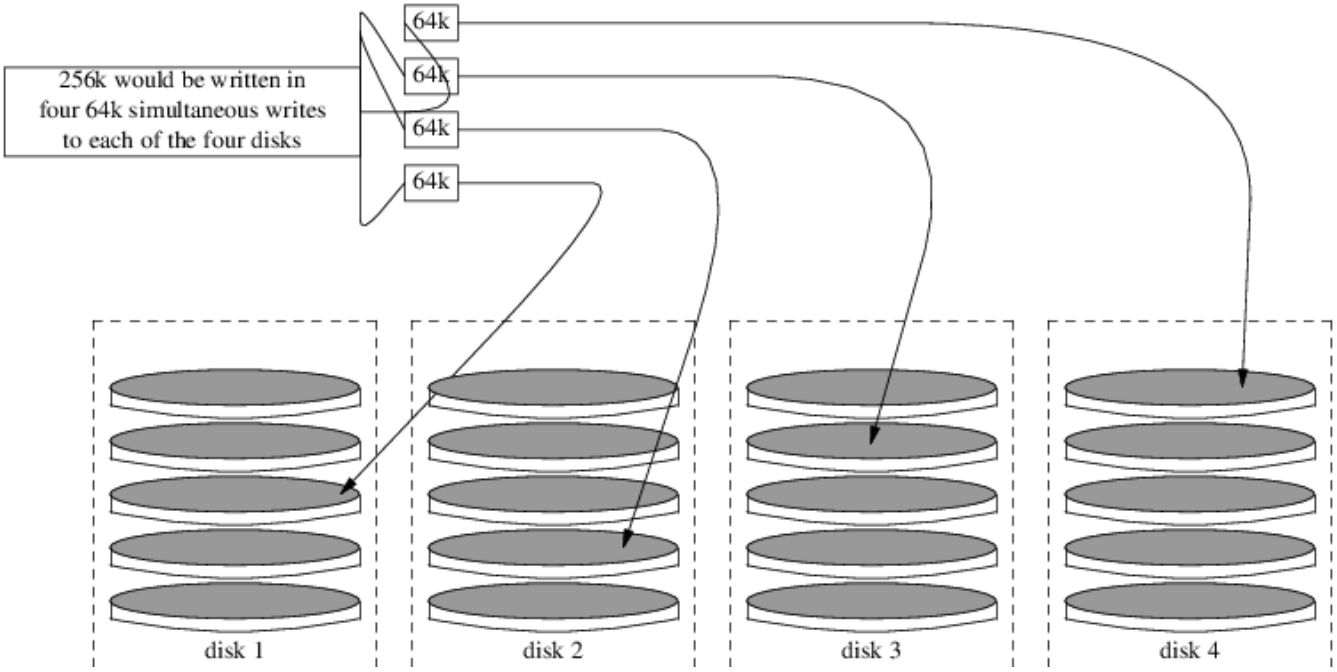
Прежде чем читать эту главу, необходимо:

- Понимание того, как FreeBSD работает с дисковыми устройствами (Хранение данных).
- Знать, как настроить и установить новый ядро (Настройка ядра FreeBSD).

22.2. RAID0 - Striping

Чередование объединяет несколько дисков в один том. Чередование может быть выполнено с использованием аппаратных RAID-контроллеров. Подсистема GEOM предоставляет программную поддержку чередования дисков, также известного как RAID0, без необходимости в RAID-контроллере.

В RAID0 данные разбиваются на блоки, которые записываются на все диски массива. Как показано на следующей иллюстрации, вместо ожидания записи 256 КБ на один диск, RAID0 может одновременно записать по 64 КБ на каждый из четырёх дисков массива, обеспечивая высокую производительность ввода-вывода. Эту производительность можно дополнительно повысить, используя несколько контроллеров дисков.



Каждый диск в дисковой последовательности RAID0 должен быть одинакового размера, поскольку запросы ввода-вывода чередуются для чтения или записи на несколько дисков параллельно.

RAID0 *не* обеспечивает избыточности. Это означает, что если один диск в массиве выйдет из строя, все данные на дисках будут потеряны. Если данные важны, реализуйте стратегию резервного копирования, которая регулярно сохраняет резервные копии на удалённую систему или устройство.

Процесс создания программного RAID0 на основе GEOM в системе FreeBSD с использованием обычных дисков выглядит следующим образом. После создания дисковой последовательности обратитесь к gstripe(8) для получения дополнительной информации о том, как управлять существующей дисковой последовательностью.

Процедура: Создание последовательности неформатированных ATA-дисков

1. Загрузите модуль geom_stripe.ko:

```
# kldload geom_stripe
```

2. Убедитесь, что существует подходящая точка монтирования. Если этот том будет использоваться как корневой раздел, временно используйте другую точку монтирования, например, /mnt.
3. Определите имена устройств для дисков, которые будут объединены в дисковую последовательность, и создайте новое устройство для последовательности. Например, для объединения в последовательность двух неиспользуемых и неразмеченных ATA-дисков с именами устройств /dev/ad2 и /dev/ad3:

```
# gstripe label -v st0 /dev/ad2 /dev/ad3
Metadata value stored on /dev/ad2.
Metadata value stored on /dev/ad3.
Done.
```

4. Запишите стандартную метку, также известную как таблица разделов, на новый том и установите код начальной загрузки по умолчанию:

```
# bsdlabel -wB /dev/stripe/st0
```

5. Этот процесс должен создать два дополнительных устройства в /dev/stripe помимо st0 — st0a и st0c. На этом этапе можно создать файловую систему UFS на st0a с помощью команды `newfs`:

```
# newfs -U /dev/stripe/st0a
```

   По экрану пройдёт множество чисел, и через несколько секунд процесс завершится. Том создан и готов к монтированию.

6. Чтобы вручную смонтировать созданную дисковую последовательность:

```
# mount /dev/stripe/st0a /mnt
```

7. Для автоматического монтирования этой чередующейся файловой системы во время загрузки добавьте информацию о томе в /etc/fstab. В этом примере создаётся постоянная точка монтирования с именем stripe:

```
# mkdir /stripe
# echo "/dev/stripe/st0a /stripe ufs rw 2 2" \
>> /etc/fstab
```

8. Модуль geom_stripe.ko также должен автоматически загружаться при инициализации системы, добавив строку в /boot/loader.conf:

```
# echo 'geom_stripe_load="YES"' >> /boot/loader.conf
```

22.3. RAID1 - Зеркалирование

RAID1, или *зеркалирование*, — это техника записи одних и тех же данных на несколько дисковых накопителей. Зеркала обычно используются для защиты от потери данных из-за выхода диска из строя. Каждый диск в зеркале содержит идентичную копию данных. При

отказе одного из дисков зеркало продолжает работать, предоставляя данные с оставшихся исправных дисков. Компьютер продолжает функционировать, а администратор имеет возможность заменить вышедший из строя диск без прерывания работы пользователей.

В этих примерах показаны две распространённые ситуации. Первый пример создаёт зеркало из двух новых дисков и использует его для замены существующего одиночного диска. Второй пример создаёт зеркало на одном новом диске, копирует на него данные со старого диска, а затем добавляет старый диск в зеркало. Хотя эта процедура немного сложнее, она требует только одного нового диска.

Традиционно два диска в зеркале имеют одинаковую модель и объём, но gmirror(8) не требует этого. Зеркала, созданные с разными дисками, будут иметь объём, равный объёму наименьшего диска в зеркале. Дополнительное место на более крупных дисках останется неиспользуемым. Диски, добавляемые в зеркало позже, должны иметь объём не меньше, чем у наименьшего диска, уже находящегося в зеркале.

Приведённые здесь процедуры зеркалирования являются неразрушающими, но, как и при любых операциях с дисками, сначала создайте полную резервную копию.

В этих процедурах используется dump(8) для копирования файловых систем, однако он не работает с файловыми системами, использующими журналирование мягких обновлений. Информацию о том, как обнаружить и отключить журналирование мягких обновлений, смотрите в tunefs(8).

22.3.1. Проблемы с метаданными

Многие дисковые системы хранят метаданные в конце каждого диска. Старые метаданные следует стереть перед повторным использованием диска для зеркала. Большинство проблем вызвано двумя конкретными типами оставшихся метаданных: таблицами разделов GPT и старыми метаданными от предыдущего зеркала.

Метаданные GPT можно удалить с помощью gpart(8). В этом примере удаляются как основная, так и резервная таблицы разделов GPT с диска ada8:

```
# gpart destroy -F ada8
```

Диск может быть удален из активного зеркала, а его метаданные стерты за один шаг с помощью gmirror(8). В этом примере диск ada8 удаляется из активного зеркала gm4:

```
# gmirror remove gm4 ada8
```

Если зеркало не запущено, но старые метаданные зеркала остались на диске, используйте `gmirror clear` для их удаления:

```
# gmirror clear ada8
```

gmirror(8) хранит один блок метаданных в конце диска. Поскольку схемы разделов GPT также хранят метаданные в конце диска, зеркалирование целых GPT-дисков с помощью gmirror(8) не рекомендуется. Здесь используется разделение MBR, так как оно хранит таблицу разделов только в начале диска и не конфликтует с метаданными зеркала.

22.3.2. Создание зеркала с двумя новыми дисками

В этом примере FreeBSD уже установлена на одном диске ada0. К системе подключены два новых диска — ada1 и ada2. На этих дисках будет создано новое зеркало, которое заменит старый одиночный диск.

Модуль ядра geom_mirror.ko должен быть либо встроен в ядро, либо загружен при загрузке или во время работы. Вручную загрузите модуль ядра сейчас:

```
# gmirror load
```

Создайте зеркало с двумя новыми дисками:

```
# gmirror label -v gm0 /dev/ada1 /dev/ada2
```

gm0 — это выбранное пользователем имя устройства, назначенное новому зеркалу. После запуска зеркала это имя устройства появляется в /dev/mirror/.

Теперь таблицы разделов MBR и bsdlabel можно создавать на зеркале с помощью gpart(8). В этом примере используется традиционная структура файловой системы с разделами для /, swap, /var, /tmp и /usr. Также подойдёт одиночный раздел / и раздел подкачки.

Разделы на зеркале не обязательно должны быть такого же размера, как и на существующем диске, но они должны быть достаточно большими, чтобы вместить все данные, уже находящиеся на ada0.

```
# gpart create -s MBR mirror/gm0
# gpart add -t freebsd -a 4k mirror/gm0
# gpart show mirror/gm0
=>       63  156301423  mirror/gm0  MBR  (74G)
         63         63                  - free -  (31k)
        126  156301299                1  freebsd  (74G)
  156301425         61                  - free -  (30k)
```

```
# gpart create -s BSD mirror/gm0s1
# gpart add -t freebsd-ufs  -a 4k -s 2g mirror/gm0s1
# gpart add -t freebsd-swap -a 4k -s 4g mirror/gm0s1
# gpart add -t freebsd-ufs  -a 4k -s 2g mirror/gm0s1
```

```
# gpart add -t freebsd-ufs  -a 4k -s 1g mirror/gm0s1
# gpart add -t freebsd-ufs  -a 4k mirror/gm0s1
# gpart show mirror/gm0s1
=>        0  156301299  mirror/gm0s1  BSD  (74G)
          0          2                     - free -  (1.0k)
          2    4194304                   1  freebsd-ufs  (2.0G)
    4194306    8388608                   2  freebsd-swap (4.0G)
   12582914    4194304                   4  freebsd-ufs  (2.0G)
   16777218    2097152                   5  freebsd-ufs  (1.0G)
   18874370  137426928                   6  freebsd-ufs  (65G)
  156301298          1                     - free -  (512B)
```

Сделайте зеркало загружаемым, установив загрузочный код в MBR и bsdlabel, а также настроив активный раздел:

```
# gpart bootcode -b /boot/mbr mirror/gm0
# gpart set -a active -i 1 mirror/gm0
# gpart bootcode -b /boot/boot mirror/gm0s1
```

Отформатируйте файловые системы на новом зеркале, включив мягкие обновления.

```
# newfs -U /dev/mirror/gm0s1a
# newfs -U /dev/mirror/gm0s1d
# newfs -U /dev/mirror/gm0s1e
# newfs -U /dev/mirror/gm0s1f
```

Файловые системы с исходного диска ada0 теперь можно скопировать на зеркало с помощью dump(8) и restore(8).

```
# mount /dev/mirror/gm0s1a /mnt
# dump -C16 -b64 -0aL -f - / | (cd /mnt && restore -rf -)
# mount /dev/mirror/gm0s1d /mnt/var
# mount /dev/mirror/gm0s1e /mnt/tmp
# mount /dev/mirror/gm0s1f /mnt/usr
# dump -C16 -b64 -0aL -f - /var | (cd /mnt/var && restore -rf -)
# dump -C16 -b64 -0aL -f - /tmp | (cd /mnt/tmp && restore -rf -)
# dump -C16 -b64 -0aL -f - /usr | (cd /mnt/usr && restore -rf -)
```

Отредактируйте файл /mnt/etc/fstab, чтобы он указывал на файловые системы нового зеркала:

```
# Device        Mountpoint  FStype  Options Dump    Pass#
/dev/mirror/gm0s1a  /       ufs rw  1   1
/dev/mirror/gm0s1b  none        swap    sw  0   0
/dev/mirror/gm0s1d  /var        ufs rw  2   2
/dev/mirror/gm0s1e  /tmp        ufs rw  2   2
```

```
/dev/mirror/gm0s1f  /usr        ufs rw  2   2
```

Если модуль ядра geom_mirror.ko не встроен в ядро, файл /mnt/boot/loader.conf редактируется для загрузки модуля при старте системы:

```
geom_mirror_load="YES"
```

Перезагрузите систему, чтобы проверить новое зеркало и убедиться, что все данные скопированы. BIOS увидит зеркало как два отдельных диска, а не как зеркало. Поскольку диски идентичны, не имеет значения, какой из них выбран для загрузки.

См. Устранение неполадок, если возникли проблемы с загрузкой. Отключение питания и извлечение исходного диска ada0 позволит сохранить его в качестве автономной резервной копии.

При использовании зеркало будет вести себя так же, как исходный одиночный диск.

22.3.3. Создание зеркала с существующим диском

В этом примере FreeBSD уже установлена на одном диске ada0. К системе подключён новый диск ada1. На новом диске будет создано однодисковое зеркало, существующая система скопирована на него, после чего старый диск будет добавлен в зеркало. Эта несколько сложная процедура необходима, потому что `gmirror` требует размещения 512-байтового блока метаданных в конце каждого диска, а на существующем ada0 обычно всё пространство уже занято.

Загрузите модуль ядра geom_mirror.ko:

```
# gmirror load
```

Проверьте размер носителя исходного диска с помощью `diskinfo`:

```
# diskinfo -v ada0 | head -n3
/dev/ada0
        512             # sectorsize
        1000204821504   # mediasize in bytes (931G)
```

Создайте зеркало на новом диске. Чтобы убедиться, что объём зеркала не превышает объём исходного диска ada0, используется gnop(8) для создания виртуального диска того же размера. Этот диск не хранит данных, а служит только для ограничения размера зеркала. Когда gmirror(8) создаёт зеркало, он ограничит объём до размера gzero.nop, даже если новый диск ada1 имеет больше места. Обратите внимание, что число *1000204821504* во второй строке соответствует размеру носителя ada0, показанному командой `diskinfo` выше.

```
# geom zero load
```

```
# gnop create -s 1000204821504 gzero
# gmirror label -v gm0 gzero.nop ada1
# gmirror forget gm0
```

Поскольку gzero.nop не хранит никаких данных, зеркало не считает его подключённым. Зеркалу указывается «забыть» неподключённые компоненты, удаляя ссылки на gzero.nop. В результате получается зеркальное устройство, содержащее только один диск — ada1.

После создания gm0 просмотрите таблицу разделов на ada0. Этот вывод сделан для диска объёмом 1 ТБ. Если в конце диска осталось нераспределённое пространство, содержимое можно скопировать напрямую с ada0 на новое зеркало.

Однако, если вывод показывает, что всё пространство на диске занято, как в следующем примере, то для метаданных зеркала размером 512 байт в конце диска не остаётся места.

```
# gpart show ada0
=>        63  1953525105        ada0  MBR  (931G)
          63  1953525105           1  freebsd  [active]  (931G)
```

В этом случае необходимо отредактировать таблицу разделов, уменьшив ёмкость на один сектор для mirror/gm0. Процедура будет описана далее.

В любом случае таблицы разделов на основном диске следует сначала скопировать с помощью `gpart backup` и `gpart restore`.

```
# gpart backup ada0 > table.ada0
# gpart backup ada0s1 > table.ada0s1
```

Эти команды создают два файла, table.ada0 и table.ada0s1. Этот пример приведён для диска объёмом 1 ТБ:

```
# cat table.ada0
MBR 4
1 freebsd         63 1953525105   [active]
```

```
# cat table.ada0s1
BSD 8
1  freebsd-ufs          0    4194304
2 freebsd-swap    4194304   33554432
4  freebsd-ufs   37748736   50331648
5  freebsd-ufs   88080384   41943040
6  freebsd-ufs  130023424  838860800
7  freebsd-ufs  968884224  984640881
```

Если в конце диска не отображается свободное пространство, размер и раздела, и

последнего раздела должен быть уменьшен на один сектор. Отредактируйте два файла, уменьшив размер и раздела, и последнего раздела на единицу. Это последние числа в каждом списке.

```
# cat table.ada0
MBR 4
1 freebsd         63 1953525104   [active]
```

```
# cat table.ada0s1
BSD 8
1  freebsd-ufs          0    4194304
2 freebsd-swap    4194304   33554432
4  freebsd-ufs   37748736   50331648
5  freebsd-ufs   88080384   41943040
6  freebsd-ufs  130023424  838860800
7  freebsd-ufs  968884224  984640880
```

Если хотя бы один сектор в конце диска был нераспределен, эти два файла можно использовать без изменений.

Теперь восстановите таблицу разделов в mirror/gm0:

```
# gpart restore mirror/gm0 < table.ada0
# gpart restore mirror/gm0s1 < table.ada0s1
```

Проверьте таблицу разделов с помощью `gpart show`. В этом примере gm0s1a для /, gm0s1d для /var, gm0s1e для /usr, gm0s1f для /data1 и gm0s1g для /data2.

```
# gpart show mirror/gm0
=>        63  1953525104  mirror/gm0  MBR  (931G)
          63  1953525042           1  freebsd  [active]  (931G)
  1953525105          62              - free -  (31k)

# gpart show mirror/gm0s1
=>         0  1953525042  mirror/gm0s1  BSD  (931G)
           0     2097152             1  freebsd-ufs  (1.0G)
     2097152    16777216             2  freebsd-swap  (8.0G)
    18874368    41943040             4  freebsd-ufs  (20G)
    60817408    20971520             5  freebsd-ufs  (10G)
    81788928   629145600             6  freebsd-ufs  (300G)
   710934528  1242590514             7  freebsd-ufs  (592G)
  1953525042          63                - free -  (31k)
```

Как срез, так и последний раздел должны иметь как минимум один свободный блок в конце диска.

Создайте файловые системы на этих новых разделах. Количество разделов может варьироваться в соответствии с исходным диском, ada0.

```
# newfs -U /dev/mirror/gm0s1a
# newfs -U /dev/mirror/gm0s1d
# newfs -U /dev/mirror/gm0s1e
# newfs -U /dev/mirror/gm0s1f
# newfs -U /dev/mirror/gm0s1g
```

Сделайте зеркало загружаемым, установив загрузочный код в MBR и bsdlabel, а также настроив активный раздел:

```
# gpart bootcode -b /boot/mbr mirror/gm0
# gpart set -a active -i 1 mirror/gm0
# gpart bootcode -b /boot/boot mirror/gm0s1
```

Настройте /etc/fstab для использования новых разделов на зеркале. Сначала создайте резервную копию этого файла, скопировав его в /etc/fstab.orig.

```
# cp /etc/fstab /etc/fstab.orig
```

Отредактируйте файл /etc/fstab, заменив /dev/ada0 на mirror/gm0.

```
# Device        Mountpoint  FStype  Options Dump    Pass#
/dev/mirror/gm0s1a  /       ufs rw  1   1
/dev/mirror/gm0s1b  none        swap    sw  0   0
/dev/mirror/gm0s1d  /var        ufs rw  2   2
/dev/mirror/gm0s1e  /usr        ufs rw  2   2
/dev/mirror/gm0s1f  /data1      ufs rw  2   2
/dev/mirror/gm0s1g  /data2      ufs rw  2   2
```

Если модуль ядра geom_mirror.ko не встроен в ядро, отредактируйте /boot/loader.conf для его загрузки при старте системы:

```
geom_mirror_load="YES"
```

Файловые системы с исходного диска теперь можно скопировать на зеркало с помощью dump(8) и restore(8). Каждая файловая система, сохранённая с помощью `dump -L`, сначала создаст снимок, что может занять некоторое время.

```
# mount /dev/mirror/gm0s1a /mnt
# dump -C16 -b64 -0aL -f - /    | (cd /mnt && restore -rf -)
# mount /dev/mirror/gm0s1d /mnt/var
# mount /dev/mirror/gm0s1e /mnt/usr
```

```
# mount /dev/mirror/gm0s1f /mnt/data1
# mount /dev/mirror/gm0s1g /mnt/data2
# dump -C16 -b64 -0aL -f - /usr | (cd /mnt/usr && restore -rf -)
# dump -C16 -b64 -0aL -f - /var | (cd /mnt/var && restore -rf -)
# dump -C16 -b64 -0aL -f - /data1 | (cd /mnt/data1 && restore -rf -)
# dump -C16 -b64 -0aL -f - /data2 | (cd /mnt/data2 && restore -rf -)
```

Перезагрузите систему, загрузившись с ada1. Если всё работает правильно, система загрузится с mirror/gm0, который теперь содержит те же данные, что ранее были на ada0. Обратитесь к разделу Устранение неполадок, если возникли проблемы с загрузкой.

На этом этапе зеркало всё ещё состоит только из одного диска ada1.

После успешной загрузки с mirror/gm0 последним шагом будет добавление ada0 в зеркало.

Когда диск ada0 добавляется в зеркало, его прежнее содержимое будет перезаписано данными из зеркала. Убедитесь, что mirror/gm0 имеет такое же содержимое, как ada0, перед добавлением ada0 в зеркало. Если данные, ранее скопированные с помощью dump(8) и restore(8), не идентичны тому, что было на ada0, верните /etc/fstab к монтированию файловых систем на ada0, перезагрузите систему и начните всю процедуру заново.

```
# gmirror insert gm0 ada0
GEOM_MIRROR: Device gm0: rebuilding provider ada0
```

Синхронизация между двумя дисками начнется немедленно. Для просмотра прогресса используйте команду `gmirror status`.

```
# gmirror status
      Name    Status  Components
mirror/gm0  DEGRADED  ada1 (ACTIVE)
                      ada0 (SYNCHRONIZING, 64%)
```

Через некоторое время синхронизация завершится.

```
GEOM_MIRROR: Device gm0: rebuilding provider ada0 finished.
# gmirror status
      Name    Status  Components
mirror/gm0  COMPLETE  ada1 (ACTIVE)
                      ada0 (ACTIVE)
```

mirror/gm0 теперь состоит из двух дисков ada0 и ada1, и их содержимое автоматически синхронизируется между собой. При использовании mirror/gm0 будет вести себя так же, как исходный одиночный диск.

22.3.4. Устранение неполадок

Если система больше не загружается, возможно, потребуется изменить настройки BIOS для загрузки с одного из новых зеркалированных дисков. Для загрузки можно использовать любой из зеркальных дисков, так как они содержат идентичные данные.

Если загрузка останавливается с таким сообщением, значит, что-то не так с зеркальным устройством:

```
Mounting from ufs:/dev/mirror/gm0s1a failed with error 19.

Loader variables:
  vfs.root.mountfrom=ufs:/dev/mirror/gm0s1a
  vfs.root.mountfrom.options=rw

Manual root filesystem specification:
  <fstype>:<device> [options]
      Mount <device> using filesystem <fstype>
      and with the specified (optional) option list.

    e.g. ufs:/dev/da0s1a
         zfs:tank
         cd9660:/dev/acd0 ro
           (which is equivalent to: mount -t cd9660 -o ro /dev/acd0 /)

  ?               List valid disk boot devices
  .               Yield 1 second (for background tasks)
  <empty line>    Abort manual input

mountroot>
```

Проблема может быть вызвана тем, что забыли загрузить модуль geom_mirror.ko , не добавив его в /boot/loader.conf. Чтобы исправить её, загрузитесь с установочного носителя FreeBSD и выберите `Shell` при первом запросе. Затем загрузите модуль mirror и смонтируйте зеркальное устройство:

```
# gmirror load
# mount /dev/mirror/gm0s1a /mnt
```

Отредактируйте файл /mnt/boot/loader.conf, добавив строку для загрузки модуля mirror:

```
geom_mirror_load="YES"
```

Сохраните файл и перезагрузите систему.

Другие проблемы, вызывающие `ошибку 19`, требуют больше усилий для исправления. Хотя система должна загружаться с ada0, появится ещё один запрос на выбор оболочки, если

/etc/fstab указан неверно. Введите `ufs:/dev/ada0s1a` в строке загрузчика и нажмите `Enter`. Отмените изменения в /etc/fstab, затем смонтируйте файловые системы с исходного диска (ada0) вместо зеркала. Перезагрузите систему и повторите процедуру снова.

```
Enter full pathname of shell or RETURN for /bin/sh:
# cp /etc/fstab.orig /etc/fstab
# shutdown -r now
```

22.3.5. Восстановление после отказа диска

Преимущество зеркалирования дисков заключается в том, что при отказе одного диска зеркало не теряет данных. В приведённом примере, если ada0 выйдет из строя, зеркало продолжит работать, предоставляя данные с оставшегося рабочего диска ada1.

Для замены вышедшего из строя диска завершите работу системы и физически замените неисправный диск новым диском равной или большей ёмкости. Производители используют довольно произвольные значения при указании ёмкости дисков в гигабайтах, и единственный способ точно убедиться в этом — сравнить общее количество секторов, отображаемое командой `diskinfo -v`. Диск с большей ёмкостью, чем у зеркала, будет работать, хотя дополнительное пространство на новом диске использоваться не будет.

После перезагрузки компьютера зеркало будет работать в "деградированном" режиме с одним диском. Необходимо указать зеркалу забыть диски, которые в данный момент не подключены:

```
# gmirror forget gm0
```

Любые старые метаданные должны быть удалены с заменяемого диска, следуя инструкциям в Проблемы с метаданными. Затем заменяющий диск, ada4 в данном примере, добавляется в зеркало:

```
# gmirror insert gm0 /dev/ada4
```

Синхронизация начинается при установке нового диска в зеркало. Этот процесс копирования данных зеркала на новый диск может занять некоторое время. Производительность зеркала значительно снизится во время копирования, поэтому добавление новых дисков лучше выполнять при низкой нагрузке на компьютер.

Ход выполнения можно отслеживать с помощью `gmirror status`, который показывает диски, находящиеся в процессе синхронизации, и процент завершения. Во время повторной синхронизации статус будет `DEGRADED`, а по завершении процесса изменится на `COMPLETE`.

22.4. RAID3 - чередование на уровне байтов с выделенной чётностью

RAID3 — это метод объединения нескольких дисков в один том с выделенным диском чётности. В системе RAID3 данные разбиваются на байты, которые записываются на все диски массива, за исключением одного диска, выполняющего роль выделенного диска чётности. Это означает, что операции чтения в RAID3 обращаются ко всем дискам массива. Производительность может быть повышена за счет использования нескольких контроллеров дисков. Массив RAID3 обеспечивает отказоустойчивость на уровне одного диска, а его емкость составляет 1 - 1/n от общей емкости всех дисков массива, где n — количество жёстких дисков в массиве. Такая конфигурация в основном подходит для хранения данных большого объёма, таких как мультимедийные файлы.

Для создания массива RAID3 требуется как минимум 3 физических жёстких диска. Каждый диск должен быть одинакового размера, так как запросы ввода-вывода чередуются для чтения или записи на несколько дисков параллельно. Кроме того, из-за особенностей RAID3 количество дисков должно быть равно 3, 5, 9, 17 и так далее, то есть соответствовать формуле 2^n + 1.

Этот раздел демонстрирует, как создать программный RAID3 в системе FreeBSD.

Хотя теоретически возможно загружаться с массива RAID3 в FreeBSD, такая конфигурация встречается редко и не рекомендуется.

22.4.1. Создание выделенного массива RAID3

В FreeBSD поддержка RAID3 реализована классом GEOM graid3(8). Создание выделенного массива RAID3 в FreeBSD требует выполнения следующих шагов.

1. Сначала загрузите модуль ядра geom_raid3.ko, выполнив одну из следующих команд:

   ```
   # graid3 load
   ```

   или:

   ```
   # kldload geom_raid3
   ```

2. Убедитесь, что существует подходящая точка монтирования. Эта команда создаёт новый каталог для использования в качестве точки монтирования:

   ```
   # mkdir /multimedia
   ```

3. Определите имена устройств для дисков, которые будут добавлены в массив, и создайте новое устройство RAID3. Последнее указанное устройство будет использоваться в качестве выделённого диска чётности. В этом примере используются три

неразделённых ATA-диска: ada1 и ada2 для данных, а ada3 для чётности.

```
# graid3 label -v gr0 /dev/ada1 /dev/ada2 /dev/ada3
Metadata value stored on /dev/ada1.
Metadata value stored on /dev/ada2.
Metadata value stored on /dev/ada3.
Done.
```

4. Разделите только что созданное устройство gr0 и разместите на нём файловую систему UFS:

```
# gpart create -s GPT /dev/raid3/gr0
# gpart add -t freebsd-ufs /dev/raid3/gr0
# newfs -j /dev/raid3/gr0p1
```

По экрану пройдёт множество чисел, и через некоторое время процесс завершится. Том создан и готов к монтированию:

```
# mount /dev/raid3/gr0p1 /multimedia/
```

Массив RAID3 теперь готов к использованию.

Для сохранения данной конфигурации после перезагрузки системы требуется дополнительная настройка.

1. Модуль geom_raid3.ko должен быть загружен перед монтированием массива. Для автоматической загрузки модуля ядра во время инициализации системы добавьте следующую строку в /boot/loader.conf:

```
geom_raid3_load="YES"
```

2. Следующую информацию о разделе необходимо добавить в /etc/fstab, чтобы автоматически монтировать файловую систему массива во время загрузки системы:

```
/dev/raid3/gr0p1    /multimedia ufs rw  2   2
```

22.5. Устройства с программным RAID

Некоторые материнские платы и карты расширения добавляют простые аппаратные компоненты, обычно только ПЗУ, что позволяет компьютеру загружаться с RAID-массива. После загрузки доступ к RAID-массиву обеспечивается программным обеспечением, работающим на основном процессоре компьютера. Такой "аппаратно-ускоренный программный RAID" создаёт массивы, не зависящие от конкретной операционной системы и функционирующие ещё до её загрузки.

Несколько уровней RAID поддерживаются в зависимости от используемого оборудования. Полный список см. в graid(8).

graid(8) требует наличия модуля ядра geom_raid.ko, который включён в ядро GENERIC начиная с FreeBSD 9.1. При необходимости его можно загрузить вручную командой `graid load`.

22.5.1. Создание массива

В устройствах с программным RAID часто есть меню, которое можно вызвать, нажав специальные клавиши при загрузке компьютера. Это меню позволяет создавать и удалять RAID-массивы. graid(8) также может создавать массивы напрямую из командной строки.

`graid label` используется для создания нового массива. В данном примере используется материнская плата с чипсетом Intel software RAID, поэтому указан формат метаданных Intel. Новый массив получает метку gm0, это зеркало (RAID1), и использует диски ada0 и ada1.

Некоторое пространство на дисках будет перезаписано при создании нового массива. Предварительно создайте резервную копию существующих данных!

```
# graid label Intel gm0 RAID1 ada0 ada1
GEOM_RAID: Intel-a29ea104: Array Intel-a29ea104 created.
GEOM_RAID: Intel-a29ea104: Disk ada0 state changed from NONE to ACTIVE.
GEOM_RAID: Intel-a29ea104: Subdisk gm0:0-ada0 state changed from NONE to ACTIVE.
GEOM_RAID: Intel-a29ea104: Disk ada1 state changed from NONE to ACTIVE.
GEOM_RAID: Intel-a29ea104: Subdisk gm0:1-ada1 state changed from NONE to ACTIVE.
GEOM_RAID: Intel-a29ea104: Array started.
GEOM_RAID: Intel-a29ea104: Volume gm0 state changed from STARTING to OPTIMAL.
Intel-a29ea104 created
GEOM_RAID: Intel-a29ea104: Provider raid/r0 for volume gm0 created.
```

Проверка состояния показывает, что новое зеркало готово к использованию:

```
# graid status
   Name   Status  Components
raid/r0  OPTIMAL  ada0 (ACTIVE (ACTIVE))
                  ada1 (ACTIVE (ACTIVE))
```

Устройство массива отображается в /dev/raid/. Первый массив называется r0. Дополнительные массивы, если они есть, будут называться r1, r2 и так далее.

В меню BIOS на некоторых из этих устройств можно создавать массивы со специальными символами в их именах. Чтобы избежать проблем с этими специальными символами, массивам присваиваются простые числовые имена, например, r0. Для отображения фактических меток, таких как gm0 в примере выше, используйте sysctl(8):

```
# sysctl kern.geom.raid.name_format=1
```

22.5.2. Несколько томов

Некоторые устройства программного RAID поддерживают более одного *тома* в массиве. Тома работают подобно разделам, позволяя разделять пространство на физических дисках и использовать его различными способами. Например, устройства программного RAID от Intel поддерживают два тома. В этом примере создаётся зеркало размером 40 ГБ для безопасного хранения операционной системы, а затем том RAID0 (чередующийся) размером 20 ГБ для быстрого временного хранения:

```
# graid label -S 40G Intel gm0 RAID1 ada0 ada1
# graid add -S 20G gm0 RAID0
```

Тома появляются как дополнительные записи rX в каталоге /dev/raid/. Массив с двумя томами будет отображать r0 и r1.

См. graid(8) для информации о количестве томов, поддерживаемых различными устройствами программного RAID.

22.5.3. Преобразование одиночного диска в зеркало

При определённых условиях возможно преобразовать существующий одиночный диск в массив graid(8) без переформатирования. Чтобы избежать потери данных во время преобразования, существующий диск должен соответствовать следующим минимальным требованиям:

- Диск должен быть размечен с использованием схемы разделов MBR. Схемы разделов GPT или другие схемы с метаданными в конце диска будут перезаписаны и повреждены метаданными graid(8).
- Для размещения метаданных graid(8) в конце диска должно быть достаточно неразмеченного и неиспользуемого пространства. Размер этих метаданных может варьироваться, но самые большие занимают 64 МБ, поэтому рекомендуется иметь как минимум столько свободного места.

Если диск соответствует этим требованиям, начните с создания полной резервной копии. Затем создайте зеркало с одним диском на этом диске:

```
# graid label Intel gm0 RAID1 ada0 NONE
```

Метаданные graid(8) были записаны в конец диска в неиспользуемое пространство. Теперь можно вставить второй диск в зеркало:

```
# graid insert raid/r0 ada1
```

Данные с исходного диска начнут немедленно копироваться на второй диск. Зеркало будет работать в деградировавшем состоянии до завершения копирования.

22.5.4. Добавление новых дисков к массиву

Диски могут быть добавлены в массив в качестве замены отказавших или отсутствующих дисков. Если нет отказавших или отсутствующих дисков, новый диск становится запасным. Например, добавление нового диска в рабочее зеркало из двух дисков приведет к зеркалу из двух дисков с одним запасным, а не к зеркалу из трёх дисков.

В примере зеркального массива данные сразу же начинают копироваться на только что вставленный диск. Любая существующая информация на новом диске будет перезаписана.

```
# graid insert raid/r0 ada1
GEOM_RAID: Intel-a29ea104: Disk ada1 state changed from NONE to ACTIVE.
GEOM_RAID: Intel-a29ea104: Subdisk gm0:1-ada1 state changed from NONE to NEW.
GEOM_RAID: Intel-a29ea104: Subdisk gm0:1-ada1 state changed from NEW to REBUILD.
GEOM_RAID: Intel-a29ea104: Subdisk gm0:1-ada1 rebuild start at 0.
```

22.5.5. Удаление дисков из массива

Отдельные диски можно навсегда удалить из массива и стереть их метаданные:

```
# graid remove raid/r0 ada1
GEOM_RAID: Intel-a29ea104: Disk ada1 state changed from ACTIVE to OFFLINE.
GEOM_RAID: Intel-a29ea104: Subdisk gm0:1-[unknown] state changed from ACTIVE to NONE.
GEOM_RAID: Intel-a29ea104: Volume gm0 state changed from OPTIMAL to DEGRADED.
```

22.5.6. Остановка массива

Массив можно остановить без удаления метаданных с дисков. Массив будет перезапущен при загрузке системы.

```
# graid stop raid/r0
```

22.5.7. Проверка состояния массива

Статус массива можно проверить в любое время. После добавления диска в зеркало в приведённом выше примере данные копируются с исходного диска на новый:

```
# graid status
   Name    Status  Components
raid/r0  DEGRADED  ada0 (ACTIVE (ACTIVE))
                   ada1 (ACTIVE (REBUILD 28%))
```

Некоторые типы массивов, такие как RAID0 или CONCAT, могут не отображаться в отчёте о состоянии при выходе дисков из строя. Чтобы увидеть эти частично неработоспособные массивы, добавьте -ga:

```
# graid status -ga
          Name  Status  Components
Intel-e2d07d9a  BROKEN  ada6 (ACTIVE (ACTIVE))
```

22.5.8. Удаление массивов

Массивы уничтожаются путём удаления всех томов из них. Когда удаляется последний оставшийся том, массив останавливается, а метаданные удаляются с дисков:

```
# graid delete raid/r0
```

22.5.9. Удаление неожиданных массивов

Диски могут неожиданно содержать метаданные graid(8), оставшиеся от предыдущего использования или тестирования производителем. graid(8) обнаружит эти диски и создаст массив, что помешает доступу к отдельному диску. Для удаления нежелательных метаданных:

1. Загрузите систему. В меню загрузки выберите 2 для перехода в приглашение загрузчика. Введите:

   ```
   OK set kern.geom.raid.enable=0
   OK boot
   ```

   Система загрузится с отключённым graid(8).

2. Создайте резервную копию всех данных на затронутом диске.
3. В качестве обходного решения обнаружение массива graid(8) можно отключить, добавив

   ```
   kern.geom.raid.enable=0
   ```

   в файл /boot/loader.conf.

   Для постоянного удаления метаданных graid(8) с затронутого диска загрузитесь с установочного CD-ROM или USB-накопителя FreeBSD и выберите Shell. Используйте команду status, чтобы найти имя массива, обычно это raid/r0:

   ```
   # graid status
      Name   Status  Components
   raid/r0  OPTIMAL  ada0 (ACTIVE (ACTIVE))
   ```

```
                ada1 (ACTIVE (ACTIVE))
```

Удалите том по имени:

```
# graid delete raid/r0
```

Если отображается более одного тома, повторите процесс для каждого тома. После удаления последнего массива том будет уничтожен.

Перезагрузите систему и проверьте данные, восстановив их из резервной копии при необходимости. После удаления метаданных запись `kern.geom.raid.enable=0` в файле /boot/loader.conf также можно удалить.

22.6. Сетевые устройства GEOM Gate

GEOM предоставляет простой механизм для удалённого доступа к устройствам, таким как диски, компакт-диски и файловые системы, с использованием сетевого демона GEOM Gate - ggated. Система с устройством запускает серверный демон, который обрабатывает запросы от клиентов, использующих ggatec. Устройства не должны содержать конфиденциальных данных, так как соединение между клиентом и сервером не шифруется.

Аналогично NFS, который рассматривается в Network File System (NFS), ggated настраивается с использованием файла экспорта. Этот файл определяет, каким системам разрешён доступ к экспортируемым ресурсам и какой уровень доступа им предоставляется. Например, чтобы предоставить клиенту `192.168.1.5` права на чтение и запись для четвёртого раздела первого SCSI-диска, создайте файл /etc/gg.exports со следующей строкой:

```
192.168.1.5 RW /dev/da0s4d
```

Перед экспортом устройства убедитесь, что оно не смонтировано. Затем запустите `ggated`:

```
# ggated
```

Для указания альтернативного порта прослушивания или изменения расположения файла экспорта по умолчанию доступно несколько вариантов. Подробности см. в ggated(8).

Для доступа к экспортированному устройству на клиентской машине сначала используйте `ggatec`, указав IP-адрес сервера и имя экспортированного устройства. В случае успеха эта команда выведет имя устройства `ggate`, которое нужно смонтировать. Смонтируйте указанное имя устройства на свободную точку монтирования. В этом примере выполняется подключение к разделу /dev/da0s4d на `192.168.1.1`, затем монтируется /dev/ggate0 в /mnt:

```
# ggatec create -o rw 192.168.1.1 /dev/da0s4d
ggate0
```

```
# mount /dev/ggate0 /mnt
```

Устройство на сервере теперь может быть доступно через /mnt на клиенте. Для получения дополнительной информации о `ggatec` и нескольких примеров использования обратитесь к ggatec(8).

Монтирование завершится ошибкой, если устройство в данный момент смонтировано на сервере или любом другом клиенте в сети. Если требуется одновременный доступ к сетевым ресурсам, используйте NFS.

Когда устройство больше не требуется, размонтируйте его с помощью `umount`, чтобы ресурс стал доступен другим клиентам.

22.7. Маркировка дисковых устройств

Во время инициализации системы ядро FreeBSD создаёт узлы устройств по мере их обнаружения. Такой метод поиска устройств вызывает некоторые проблемы. Например, что если новое дисковое устройство будет добавлено через USB? Вполне вероятно, что флеш-накопитель может получить имя устройства da0, а исходное устройство da0 сместится на da1. Это вызовет проблемы с монтированием файловых систем, если они указаны в /etc/fstab, что также может помешать загрузке системы.

Одно из решений — последовательное подключение SCSI-устройств, чтобы новое устройство, добавленное к SCSI-карте, получало неиспользуемые номера. Но что делать с USB-устройствами, которые могут заменить основной SCSI-диск? Это происходит потому, что USB-устройства обычно определяются раньше SCSI-карты. Один из вариантов — подключать такие устройства только после загрузки системы. Другой способ — использовать только один ATA-диск и никогда не указывать SCSI-устройства в /etc/fstab.

Лучшим решением будет использование `glabel` для маркировки дисковых устройств и применение меток в /etc/fstab. Поскольку `glabel` сохраняет метку в последнем секторе заданного провайдера, метка останется неизменной после перезагрузки. Используя эту метку в качестве устройства, файловую систему можно всегда монтировать, независимо от того, через какой узел устройства к ней обращаются.

`glabel` может создавать как временные, так и постоянные метки. Только постоянные метки сохраняются после перезагрузки. Дополнительную информацию о различиях между метками можно найти в glabel(8).

22.7.1. Типы меток и примеры

Постоянные метки могут быть общими или метками файловой системы. Постоянные метки файловой системы можно создать с помощью tunefs(8) или newfs(8). Такие метки создаются в подкаталоге /dev и именуются в соответствии с типом файловой системы. Например, метки файловой системы UFS2 будут созданы в /dev/ufs. Общие постоянные метки можно создать с помощью `glabel label`. Они не привязаны к конкретной файловой системе и будут размещены в /dev/label.

Временные метки удаляются при следующей перезагрузке. Эти метки создаются в /dev/label и подходят для экспериментов. Временную метку можно создать с помощью `glabel create`.

Чтобы создать постоянную метку для файловой системы UFS2 без уничтожения данных, выполните следующую команду:

```
# tunefs -L home /dev/da3
```

Теперь в /dev/ufs должна существовать метка, которую можно добавить в /etc/fstab:

```
/dev/ufs/home       /home           ufs     rw              2       2
```

Файловая система не должна быть смонтирована при попытке запуска `tunefs`.

Теперь файловую систему можно смонтировать:

```
# mount /home
```

Начиная с этого момента, при условии, что модуль ядра geom_label.ko загружается при загрузке через /boot/loader.conf или присутствует опция ядра `GEOM_LABEL`, изменение узла устройства не окажет негативного влияния на систему.

Файловые системы также могут быть созданы с меткой по умолчанию с использованием флага `-L` в `newfs`. Подробнее см. в newfs(8).

Следующая команда может быть использована для удаления метки:

```
# glabel destroy home
```

Следующий пример показывает, как назначить метки разделам загрузочного диска.

Пример 32. Разметка разделов на загрузочном диске

Постоянная маркировка разделов на загрузочном диске позволяет системе продолжать нормальную загрузку, даже если диск будет перемещен на другой контроллер или перенесён в другую систему. В этом примере предполагается, что используется один ATA-диск, который система в настоящее время распознает как ad0. Также предполагается, что используется стандартная схема разделов FreeBSD с /, /var, /usr, /tmp, а также разделом подкачки.

Перезагрузите систему, и на запросе loader(8) нажмите `4`, чтобы загрузиться в однопользовательском режиме. Затем введите следующие команды:

```
# glabel label rootfs /dev/ad0s1a
```

```
GEOM_LABEL: Label for provider /dev/ad0s1a is label/rootfs
# glabel label var /dev/ad0s1d
GEOM_LABEL: Label for provider /dev/ad0s1d is label/var
# glabel label usr /dev/ad0s1f
GEOM_LABEL: Label for provider /dev/ad0s1f is label/usr
# glabel label tmp /dev/ad0s1e
GEOM_LABEL: Label for provider /dev/ad0s1e is label/tmp
# glabel label swap /dev/ad0s1b
GEOM_LABEL: Label for provider /dev/ad0s1b is label/swap
# exit
```

Система продолжит загрузку в многопользовательском режиме. После завершения загрузки отредактируйте файл /etc/fstab и замените стандартные имена устройств на соответствующие метки. Итоговый файл /etc/fstab будет выглядеть следующим образом:

```
# Device                Mountpoint      FStype  Options         Dump    Pass#
/dev/label/swap         none            swap    sw              0       0
/dev/label/rootfs       /               ufs     rw              1       1
/dev/label/tmp          /tmp            ufs     rw              2       2
/dev/label/usr          /usr            ufs     rw              2       2
/dev/label/var          /var            ufs     rw              2       2
```

Систему теперь можно перезагрузить. Если всё прошло успешно, она загрузится в обычном режиме, и команда `mount` покажет:

```
# mount
/dev/label/rootfs on / (ufs, local)
devfs on /dev (devfs, local)
/dev/label/tmp on /tmp (ufs, local, soft-updates)
/dev/label/usr on /usr (ufs, local, soft-updates)
/dev/label/var on /var (ufs, local, soft-updates)
```

Класс glabel(8) поддерживает тип метки для файловых систем UFS, основанный на уникальном идентификаторе файловой системы, `ufsid`. Эти метки могут быть найдены в /dev/ufsid и создаются автоматически во время запуска системы. Можно использовать метки `ufsid` для монтирования разделов с помощью /etc/fstab. Для получения списка файловых систем и соответствующих им меток `ufsid` используйте команду `glabel status`:

```
% glabel status
                  Name  Status  Components
ufsid/486b6fc38d330916     N/A  ad4s1d
ufsid/486b6fc16926168e     N/A  ad4s1f
```

В приведённом выше примере ad4s1d соответствует /var, а ad4s1f — /usr. Используя указанные значения `ufsid`, эти разделы можно смонтировать, добавив следующие строки в

/etc/fstab:

```
/dev/ufsid/486b6fc38d330916        /var        ufs        rw        2      2
/dev/ufsid/486b6fc16926168e        /usr        ufs        rw        2      2
```

Любые разделы с метками `ufsid` могут быть смонтированы таким образом, что устраняет необходимость вручную создавать постоянные метки, сохраняя при этом преимущества монтирования, независимого от имён устройств.

22.8. Журналирование UFS через GEOM

Поддержка журналирования для файловых систем UFS доступна в FreeBSD. Реализация предоставляется через подсистему GEOM и настраивается с помощью `gjournal`. В отличие от других реализаций журналирования файловых систем, метод `gjournal` работает на уровне блоков и не является частью файловой системы. Это расширение GEOM.

В журналировании хранится журнал транзакций файловой системы, таких как изменения, составляющие полную операцию записи на диск, до того как метаданные и записи файлов будут записаны на диск. Этот журнал транзакций может быть позднее воспроизведен для повторного выполнения операций файловой системы, предотвращая её рассогласование.

Этот метод предоставляет ещё один механизм защиты от потери данных и нарушения целостности файловой системы. В отличие от мягких обновлений, которые отслеживают и контролируют обновления метаданных, и снимков, создающих образ файловой системы, журнал хранится в отдельном месте на диске, специально выделенном для этой задачи. Для повышения производительности журнал может храниться на другом диске. В такой конфигурации провайдер журнала или устройство хранения должны быть указаны после устройства, для которого включается ведение журнала.

Ядро GENERIC предоставляет поддержку `gjournal`. Чтобы автоматически загружать модуль ядра geom_journal.ko при загрузке, добавьте следующую строку в /boot/loader.conf:

```
geom_journal_load="YES"
```

Если используется собственный вариант ядра, убедитесь, что следующая строка присутствует в конфигурационном файле ядра:

```
options GEOM_JOURNAL
```

После загрузки модуля журнал можно создать на новой файловой системе, выполнив следующие шаги. В этом примере da4 — это новый SCSI-диск:

```
# gjournal load
# gjournal label /dev/da4
```

Это загрузит модуль и создаст узел устройства /dev/da4.journal на /dev/da4.

Файловая система UFS теперь может быть создана на журналируемом устройстве, а затем смонтирована в существующей точке монтирования:

```
# newfs -O 2 -J /dev/da4.journal
# mount /dev/da4.journal /mnt
```

В случае нескольких разделов журнал будет создан для каждого отдельного раздела. Например, если ad4s1 и ad4s2 являются разделами, то `gjournal` создаст ad4s1.journal и ad4s2.journal.

Журналирование также можно включить на существующих файловых системах с помощью `tunefs`. Однако *всегда* делайте резервную копию перед изменением существующей файловой системы. В большинстве случаев `gjournal` завершится с ошибкой, если не сможет создать журнал, но это не защищает от потери данных из-за неправильного использования `tunefs`. Дополнительную информацию об этих командах можно найти в gjournal(8) и tunefs(8).

Возможно журналирование загрузочного диска в системе FreeBSD. Подробные инструкции приведены в статье Реализация журналирования UFS на настольном ПК.

Глава 23. Файловая система Z (ZFS)

ZFS — это продвинутая файловая система, разработанная для решения основных проблем, присущих предыдущему программному обеспечению подсистем хранения данных.

Первоначально разработанная в Sun Microsystems™, дальнейшая разработка открытой версии ZFS переместилась в проект OpenZFS.

ZFS имеет три основные цели проектирования:

- Целостность данных: Все данные включают контрольную сумму. ZFS вычисляет контрольные суммы и записывает их вместе с данными. При последующем чтении этих данных ZFS пересчитывает контрольные суммы. Если контрольные суммы не совпадают, что означает обнаружение одной или нескольких ошибок данных, ZFS попытается автоматически исправить ошибки, если доступны двойные-, зеркальные- или блоки чётности.
- Объединенное хранилище: добавление физических устройств хранения в пул и выделение пространства из этого общего пула. Пространство доступно для всех файловых систем и томов и увеличивается за счет добавления новых устройств хранения в пул.
- Производительность: механизмы кэширования обеспечивают повышенную производительность. ARC — это продвинутый кэш для чтения, основанный на оперативной памяти. ZFS предоставляет второй уровень кэша для чтения на основе диска — L2ARC, а также кэш для синхронной записи на основе диска под названием ZIL.

Полный список возможностей и терминологии приведён в Особенности и терминология ZFS.

23.1. Что отличает ZFS от других

ZFS — это не просто файловая система, она принципиально отличается от традиционных файловых систем. Объединение традиционно разделённых ролей менеджера томов и файловой системы даёт ZFS уникальные преимущества. Теперь файловая система осведомлена о структуре нижележащих дисков. Традиционные файловые системы могли существовать только на одном диске. Если было два диска, приходилось создавать две отдельные файловые системы. Традиционная конфигурация аппаратного RAID решала эту проблему, предоставляя операционной системе один логический диск, состоящий из пространства физических дисков, поверх которого операционная система размещала файловую систему. Даже в программных решениях RAID, таких как предоставляемые GEOM, файловая система UFS, находящаяся поверх RAID, считает, что работает с одним устройством. Комбинация менеджера томов и файловой системы в ZFS решает эту проблему и позволяет создавать файловые системы, которые совместно используют общий пул доступного хранилища. Одно из больших преимуществ осведомленности ZFS о физической структуре дисков заключается в том, что существующие файловые системы автоматически расширяются при добавлении дополнительных дисков в пул. Это новое пространство становится доступным для файловых систем. ZFS также может применять разные свойства к каждой файловой системе. Это делает полезным создание отдельных

файловых систем и наборов данных вместо единой монолитной файловой системы.

23.2. Краткое руководство по началу работы

Установщик FreeBSD может установить систему непосредственно в пул ZFS, конфигурация, известная как Root-on-ZFS; см. Разметка диска с использованием Root-on-ZFS с помощью мастера. В этом разделе показано, как создавать и управлять дополнительными пулами ZFS и наборами данных на работающей системе.

FreeBSD может монтировать пулы и наборы данных ZFS во время инициализации системы. Чтобы включить эту функцию, запустите:

```
# sysrc zfs_enable="YES"
```

Это добавляет `zfs_enable="YES"` в /etc/rc.conf. Выполнение `service zfs enable` вносит то же изменение.

Затем запустите службу:

```
# service zfs start
```

Примеры в этом разделе предполагают использование трёх SATA-дисков с именами устройств ada0, ada1 и ada2. Пользователям оборудования SCSI/SAS следует использовать имена устройств da, пользователям оборудования NVMe — имена устройств nda.

23.2.1. Пул на одном диске

Сначала создайте раздел GPT на пустом диске. Затем добавьте раздел типа `freebsd-zfs`, из которого создается один нерезервируемый пул:

```
# gpart create -s gpt ada1
# gpart add -t freebsd-zfs ada1
# zpool create example /dev/ada1p1
```

Для просмотра нового пула ознакомьтесь с выводом команды `df`:

```
# df /example
Filesystem     1K-blocks    Used    Avail Capacity  Mounted on
example         17547136       0 17547136     0%    /example
```

Этот вывод показывает создание и монтирование пула `example`, который теперь доступен как файловая система. Создайте файлы для пользователей, чтобы посмотреть, что все работает:

```
# cd /example
# ls
# touch testfile
# ls -al
total 4
drwxr-xr-x   2 root  wheel    3 Aug 29 23:15 .
drwxr-xr-x  21 root  wheel  512 Aug 29 23:12 ..
-rw-r--r--   1 root  wheel    0 Aug 29 23:15 testfile
```

Этот пул пока не использует расширенные функции и свойства ZFS. Чтобы создать набор данных в этом пуле с включённым сжатием:

```
# zfs create example/compressed
# zfs set compression=gzip example/compressed
```

Набор данных `example/compressed` теперь представляет собой сжатую файловую систему ZFS. Попробуйте скопировать несколько больших файлов в /example/compressed.

Отключите сжатие с помощью:

```
# zfs set compression=off example/compressed
```

Чтобы отмонтировать файловую систему, используйте `zfs umount`, а затем проверьте с помощью `df`:

```
# zfs umount example/compressed
# df
Filesystem   1K-blocks    Used    Avail Capacity  Mounted on
example       17547008       0 17547008     0%    /example
```

Для повторного монтирования файловой системы, чтобы сделать её снова доступной, используйте `zfs mount` и проверьте с помощью `df`:

```
# zfs mount example/compressed
# df
Filesystem         1K-blocks    Used    Avail Capacity  Mounted on
example             17547008       0 17547008     0%    /example
example/compressed  17547008       0 17547008     0%    /example/compressed
```

Выполнение команды `mount` отображает пул и файловые системы:

```
# mount
/dev/ada0p1 on / (ufs, local)
devfs on /dev (devfs, local)
```

```
/dev/ada0p2 on /usr (ufs, local, soft-updates)
example on /example (zfs, local)
example/compressed on /example/compressed (zfs, local)
```

Используйте наборы данных ZFS как любую файловую систему после создания. Настраивайте другие доступные функции для каждого набора данных по мере необходимости. В примере ниже создаётся новая файловая система с именем `data`. Предполагается, что файловая система содержит важные файлы, и для неё настроено хранение двух копий каждого блока данных.

```
# zfs create example/data
# zfs set copies=2 example/data
```

Используйте `df` для просмотра данных и использования пространства:

```
# df
Filesystem         1K-blocks    Used    Avail Capacity  Mounted on
/dev/ada0p1          2026030  235234  1628714    13%    /
devfs                      1       1        0   100%    /dev
/dev/ada0p2         54098308 1032864 48737580     2%    /usr
example             17547008       0 17547008     0%    /example
example/compressed  17547008       0 17547008     0%    /example/compressed
example/data        17547008       0 17547008     0%    /example/data
```

Обратите внимание, что все файловые системы в пуле имеют одинаковое доступное пространство. Использование `df` в этих примерах показывает, что файловые системы занимают столько места, сколько им нужно, и все используют один и тот же пул. ZFS устраняет такие понятия, как тома и разделы, и позволяет нескольким файловым системам совместно использовать один пул.

Чтобы уничтожить файловые системы, а затем пул, который больше не нужен:

```
# zfs destroy example/compressed
# zfs destroy example/data
# zpool destroy example
```

23.2.2. RAID-Z

Диски выходят из строя. Один из способов избежать потери данных при отказе диска — использование RAID. ZFS поддерживает эту возможность в своей конструкции пула. Пуллы RAID-Z требуют трёх или более дисков, но предоставляют больше полезного пространства, чем зеркальные пуллы.

Этот пример создает пул RAID-Z. Сначала на каждом диске создаются разделы GPT, затем укажите эти разделы GPT для добавления в пул с помощью ключевого слова `raidz`:

```
# gpart create -s gpt ada0
# gpart add -t freebsd-zfs ada0
# gpart create -s gpt ada1
# gpart add -t freebsd-zfs ada1
# gpart create -s gpt ada2
# gpart add -t freebsd-zfs ada2
# zpool create storage raidz ada0p1 ada1p1 ada2p1
```

Рекомендовано использовать от трёх до девяти устройств в конфигурации RAID-Z. Для сред, требующих единого пула из 10 или более дисков, рекомендуется разбить его на меньшие группы RAID-Z. Если доступно два диска, ZFS-зеркалирование обеспечит избыточность при необходимости. Подробнее см. в zpool(8).

Предыдущий пример создал пул `storage`. В этом примере в этом пуле создаётся новая файловая система с именем `home`:

```
# zfs create storage/home
```

Включить сжатие и сохранить дополнительную копию каталогов и файлов:

```
# zfs set copies=2 storage/home
# zfs set compression=gzip storage/home
```

Чтобы сделать это новым домашним каталогом для пользователей, скопируйте пользовательские данные в этот каталог и создайте соответствующие символические ссылки:

```
# cp -rp /home/* /storage/home
# rm -rf /home /usr/home
# ln -s /storage/home /home
# ln -s /storage/home /usr/home
```

Данные пользователя теперь хранятся в только что созданном /storage/home. Проверьте это, добавив нового пользователя и войдя в систему под его учётной записью.

Создайте снимок файловой системы для последующего отката:

```
# zfs snapshot storage/home@2026-07-11
```

ZFS создаёт снимки набора данных, а не отдельного каталога или файла.

Символ `@` является разделителем между именем файловой системы или тома и именем снимка. Перед удалением важного каталога создайте резервную копию файловой системы,

а затем откатитесь к более раннему снимку, в котором каталог ещё существует:

```
# zfs rollback storage/home@2026-07-11
```

Чтобы перечислить все доступные снимки, выполните команду `ls` в каталоге .zfs/snapshot файловой системы. Например, чтобы увидеть сделанный снимок:

```
# ls /storage/home/.zfs/snapshot
```

Напишите скрипт для создания регулярных снимков пользовательских данных. Со временем снимки могут занимать много места на диске. Удалите предыдущий снимок с помощью команды:

```
# zfs destroy storage/home@2026-07-11
```

После тестирования сделайте /storage/home настоящим /home с помощью следующей команды:

```
# zfs set mountpoint=/home storage/home
```

Выполните команды `df` и `mount`, чтобы убедиться, что система теперь распознает файловую систему как настоящий /home:

```
# mount
/dev/ada0p1 on / (ufs, local)
devfs on /dev (devfs, local)
/dev/ada0p2 on /usr (ufs, local, soft-updates)
storage on /storage (zfs, local)
storage/home on /home (zfs, local)
# df
Filesystem   1K-blocks    Used    Avail Capacity  Mounted on
/dev/ada0p1    2026030  235240  1628708    13%    /
devfs                1       1        0   100%    /dev
/dev/ada0p2   54098308 1032826 48737618     2%    /usr
storage       26320512       0 26320512     0%    /storage
storage/home  26320512       0 26320512     0%    /home
```

Настройка RAID-Z завершена. Для добавления ежедневных отчётов о состоянии созданных файловых систем в ночные запуски periodic(8) добавьте следующую строку в /etc/periodic.conf:

```
daily_status_zfs_enable="YES"
```

periodic(8) также может выполнять запланированные проверки пула; см. Проверка пула.

23.2.3. Восстановление RAID-Z

Каждый программный RAID имеет метод контроля своего состояния. Просмотр состояния устройств RAID-Z осуществляется с помощью:

```
# zpool status -x
```

Если все пулы находятся в состоянии онлайн и все работает нормально, сообщение будет следующим:

```
all pools are healthy
```

Если возникла проблема, например, диск находится в состоянии оффлайн, состояние пула будет выглядеть так:

```
  pool: storage
 state: DEGRADED
status: One or more devices has been taken offline by the administrator.
        Sufficient replicas exist for the pool to continue functioning in a
        degraded state.
action: Online the device using 'zpool online' or replace the device with
        'zpool replace'.
  scan: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        storage     DEGRADED     0     0     0
          raidz1-0  DEGRADED     0     0     0
            ada0p1  ONLINE       0     0     0
            ada1p1  OFFLINE      0     0     0
            ada2p1  ONLINE       0     0     0

errors: No known data errors
```

"OFFLINE" показывает, что администратор перевел ada1p1 в автономный режим с помощью:

```
# zpool offline storage ada1p1
```

Выключите компьютер и замените диск ada1p1. Заново создайте GPT-раздел типа freebsd-zfs на новом диске. Включите компьютер и верните ada1p1 в пул:

```
# zpool replace storage ada1p1
```

Далее снова проверьте статус, на этот раз без `-x`, чтобы отобразить все пулы:

```
# zpool status storage
  pool: storage
 state: ONLINE
  scan: resilvered 3.21G in 00:04:36 with 0 errors on Sat Jul 11 10:32:19 2026
config:

        NAME        STATE     READ WRITE CKSUM
        storage     ONLINE       0     0     0
          raidz1-0  ONLINE       0     0     0
            ada0p1  ONLINE       0     0     0
            ada1p1  ONLINE       0     0     0
            ada2p1  ONLINE       0     0     0

errors: No known data errors
```

В этом примере все в порядке.

23.2.4. Проверка данных

ZFS использует контрольные суммы для проверки целостности хранимых данных. Создание файловых систем автоматически включает их.

Отключение контрольных сумм возможно, но *не* рекомендуется! Контрольные суммы занимают мало места и обеспечивают целостность данных. Большинство функций ZFS не будут работать корректно при отключённых контрольных суммах. Их отключение не приведет к заметному повышению производительности.

Проверка контрольных сумм данных (называемая *scrubbing*) обеспечивает целостность пула `storage` с помощью:

```
# zpool scrub storage
```

Продолжительность очистки зависит от объёма хранимых данных. Большие объёмы данных требуют пропорционально больше времени для проверки. Поскольку очистка интенсивно использует операции ввода-вывода, ZFS позволяет выполнять только одну очистку на каждом пуле одновременно. После завершения очистки просмотрите статус с помощью `zpool status`:

```
# zpool status storage
  pool: storage
 state: ONLINE
```

```
  scan: scrub repaired 0B in 00:19:16 with 0 errors on Sat Jul 11 10:32:19 2026
config:

        NAME        STATE     READ WRITE CKSUM
        storage     ONLINE       0     0     0
          raidz1-0  ONLINE       0     0     0
            ada0p1  ONLINE       0     0     0
            ada1p1  ONLINE       0     0     0
            ada2p1  ONLINE       0     0     0

errors: No known data errors
```

Отображение даты завершения последней очистки помогает определить, когда начать следующую. Регулярная очистка защищает данные от тихих повреждений и гарантирует целостность пула.

См. zfs(8) и zpool(8) для других опций ZFS.

23.3. Администрирование `zpool`

Управление ZFS осуществляется с помощью двух основных утилит. Утилита `zpool` контролирует работу пула и позволяет добавлять, удалять, заменять и управлять дисками. Утилита `zfs` позволяет создавать, уничтожать и управлять наборами данных, включая как файловые системы, так и тома.

23.3.1. Создание и удаление пулов хранения данных

Самое важное решение при создании пула хранения ZFS — это выбор типов vdev для группировки физических дисков. См. список типов vdev для получения подробной информации о возможных вариантах. Типы vdev определяют характеристики избыточности, ёмкости и производительности пула. Раскладка пула не фиксируется навсегда при его создании. Зеркала позволяют добавлять новые диски в vdev, а полосы (stripes) обновляются до зеркал путём присоединения нового диска к vdev. Vdev RAID-Z увеличиваются на один диск за раз с помощью расширения RAID-Z, которое требует OpenZFS 2.3 или более поздней версии, впервые доступной во FreeBSD 15.0. Добавление новых vdev расширяет пул в любое время, а удаление верхних уровней vdev возможно в определённых пределах. Изменение формы активного пула занимает время и пропускную способность ввода-вывода, поэтому выбор подходящих типов vdev заранее остаётся важным.

Создайте простой зеркальный пул на дисках, содержащих GPT-разделы типа `freebsd-zfs`:

```
# zpool create mypool mirror /dev/ada1p1 /dev/ada2p1
# zpool status
  pool: mypool
 state: ONLINE
config:
```

```
        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            ada1p1  ONLINE       0     0     0
            ada2p1  ONLINE       0     0     0

errors: No known data errors
```

Чтобы создать более одного vdev одной командой, укажите группы дисков, разделённые ключевым словом типа vdev, в данном примере `mirror`:

```
# zpool create mypool mirror /dev/ada1p1 /dev/ada2p1 mirror /dev/ada3p1 /dev/ada4p1
# zpool status
  pool: mypool
 state: ONLINE
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            ada1p1  ONLINE       0     0     0
            ada2p1  ONLINE       0     0     0
          mirror-1  ONLINE       0     0     0
            ada3p1  ONLINE       0     0     0
            ada4p1  ONLINE       0     0     0

errors: No known data errors
```

Пулы также могут использовать разделы вместо целых дисков. Размещение ZFS в отдельном разделе позволяет одному диску иметь другие разделы для иных целей. В частности, это позволяет добавлять разделы с загрузочным кодом и файловыми системами, необходимыми для загрузки. Это даёт возможность загружаться с дисков, которые также являются членами пула. ZFS не приводит к снижению производительности на FreeBSD при использовании раздела вместо целого диска. Использование разделов также позволяет администратору *недоиспользовать* диски, задействуя не всю их емкость. Если будущий заменяемый диск того же номинального размера, что и оригинальный, на самом деле имеет немного меньшую емкость, меньший раздел все равно поместится на заменяемом диске.

Создание пула RAID-Z2 с использованием разделов:

```
# zpool create mypool raidz2 /dev/ada0p1 /dev/ada1p1 /dev/ada2p1 /dev/ada3p1
/dev/ada4p1 /dev/ada5p1
# zpool status
  pool: mypool
 state: ONLINE
config:
```

```
        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          raidz2-0  ONLINE       0     0     0
            ada0p1  ONLINE       0     0     0
            ada1p1  ONLINE       0     0     0
            ada2p1  ONLINE       0     0     0
            ada3p1  ONLINE       0     0     0
            ada4p1  ONLINE       0     0     0
            ada5p1  ONLINE       0     0     0

errors: No known data errors
```

ZFS выравнивает и подбирает размер своих минимальных записей для каждого vdev на основе значения `ashift` этого vdev, которое является логарифмом размера сектора по основанию 2. OpenZFS определяет размер сектора, сообщаемый дисками при создании vdev, и выбирает `ashift` соответствующим образом. Некоторые диски сообщают секторы в 512 байт для совместимости, но внутренне используют секторы в 4096 байт. Создавайте пулы на таких дисках с явным параметром `-o ashift=12`, чтобы принудительно установить выравнивание по 4096 байт:

```
# zpool create -o ashift=12 mypool mirror /dev/ada1p1 /dev/ada2p1
```

Установка переменной sysctl(8) `vfs.zfs.vdev.min_auto_ashift` в значение `12` дает тот же результат для создания пула и для последующих операций `zpool add` и `zpool attach`. `ashift` — это свойство каждого vdev, а не пула в целом. Оно фиксируется при создании vdev и не может быть изменено впоследствии, поэтому проверьте его перед записью данных в пул.

Уничтожьте пул, который больше не нужен, чтобы повторно использовать диски. Перед уничтожением пула необходимо размонтировать файловые системы в этом пуле. Если какой-либо набор данных используется, операция размонтирования завершится неудачей без уничтожения пула. Принудительное уничтожение пула выполняется с помощью `-f`. Это может привести к неопределённому поведению приложений, у которых были открыты файлы в этих наборах данных.

23.3.2. Свойства пулов

Подобно наборам данных, пулы имеют свойства, которые сообщают о состоянии и управляют поведением. Отобразите каждое свойство пула с помощью `zpool get all mypool` или укажите конкретные свойства:

```
# zpool get health,capacity mypool
NAME    PROPERTY  VALUE   SOURCE
mypool  health    ONLINE  -
mypool  capacity  27%     -
```

Измените изменяемое свойство с помощью `zpool set`:

```
# zpool set comment="Backup pool" mypool
# zpool get comment mypool
NAME    PROPERTY  VALUE        SOURCE
mypool  comment   Backup pool  local
```

Такие свойства, как `size`, `capacity`, `fragmentation` и `health`, являются статусными значениями только для чтения. Другие изменяют поведение пула; в этой главе используется `autoexpand` для автоматического расширения пула (Увеличение размера пула), `autoreplace` с горячими резервными дисками (Горячие резервные диски и автоматическая замена с помощью zfsd), `autotrim` на пулах с SSD (TRIM and Initialization) и `compatibility` для переносимых пулов (Обновление пула дисков). Устанавливайте свойства при создании пула, передавая `-o` команде `zpool create`, как показано выше с `ashift`. zpoolprops(7) описывает все свойства пула.

23.3.3. Добавление и удаление устройств

Существует два способа добавления дисков в пул: подключение диска к существующему vdev с помощью `zpool attach` или добавление vdev в пул с помощью `zpool add`. Некоторые типы vdev позволяют добавлять диски в vdev после его создания. Виртуальные устройства (vdev) RAID-Z принимают новые диски исключительно посредством расширения RAID-Z.

Пул, созданный с одним диском, не обладает избыточностью. Он может обнаружить повреждение данных, но не может его исправить, так как нет другой копии данных. Свойство Копии (copies) может восстановить данные после небольшого сбоя, например, повреждённого сектора, но не обеспечивает такой же уровень защиты, как зеркалирование или RAID-Z. Начиная с пула, состоящего из однодискового vdev, используйте `zpool attach` для добавления нового диска в vdev, создавая зеркало. Также используйте `zpool attach` для добавления новых дисков в зеркальную группу, увеличивая избыточность и производительность чтения. При разметке дисков, используемых для пула, повторите разметку первого диска на втором. Используйте `gpart backup` и `gpart restore` для упрощения этого процесса.

Обновите однодисковый vdev (stripe) ada0p3 до зеркала, присоединив ada1p3:

```
# zpool status
  pool: mypool
 state: ONLINE
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          ada0p3    ONLINE       0     0     0

errors: No known data errors
# zpool attach mypool ada0p3 ada1p3
Make sure to wait until resilver is done before rebooting.
# gpart bootcode -b /boot/pmbr -p /boot/gptzfsboot -i 1 ada1
partcode written to ada1p1
bootcode written to ada1
```

```
# zpool status
  pool: mypool
 state: ONLINE
status: One or more devices is currently being resilvered.  The pool will
        continue to function, possibly in a degraded state.
action: Wait for the resilver to complete.
  scan: resilver in progress since Sat Jul 11 11:15:25 2026
        738M scanned at 105M/s, 522M issued at 74.6M/s, 781M total
        522M resilvered, 66.84% done, 00:00:03 to go
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            ada0p3  ONLINE       0     0     0
            ada1p3  ONLINE       0     0     0  (resilvering)

errors: No known data errors
# zpool status
  pool: mypool
 state: ONLINE
  scan: resilvered 781M in 00:00:16 with 0 errors on Sat Jul 11 11:18:36 2026
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            ada0p3  ONLINE       0     0     0
            ada1p3  ONLINE       0     0     0

errors: No known data errors
```

Если добавление дисков к существующему vdev невозможно, альтернативным методом является добавление другого vdev в пул. Добавление vdev повышает производительность за счет распределения записей между vdev. Каждый vdev обеспечивает свою собственную избыточность. Возможно смешивание типов vdev, таких как `mirror` и `RAID-Z`, но это не рекомендуется. Добавление не избыточного vdev к пулу, содержащему mirror или RAID-Z vdev, подвергает риску данные во всем пуле. Распределение записей означает, что отказ не избыточного диска приведет к потере части каждого блока, записанного в пул.

ZFS распределяет данные по всем vdev. Например, при использовании двух зеркальных vdev это фактически эквивалентно RAID 10, где записи распределяются по двум наборам зеркал. ZFS выделяет пространство таким образом, что каждый vdev достигает 100% заполненности одновременно. Наличие vdev с разным количеством свободного места снижает производительность, так как больше данных записывается на менее заполненный vdev.

При подключении новых устройств к загрузочному пулу не забудьте обновить загрузочный код.

Присоедините вторую группу зеркал (ada2p3 и ada3p3) к существующему зеркалу:

```
# zpool status
  pool: mypool
 state: ONLINE
  scan: resilvered 781M in 00:00:16 with 0 errors on Sat Jul 11 12:11:56 2026
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            ada0p3  ONLINE       0     0     0
            ada1p3  ONLINE       0     0     0

errors: No known data errors
# zpool add mypool mirror ada2p3 ada3p3
# gpart bootcode -b /boot/pmbr -p /boot/gptzfsboot -i 1 ada2
partcode written to ada2p1
bootcode written to ada2
# gpart bootcode -b /boot/pmbr -p /boot/gptzfsboot -i 1 ada3
partcode written to ada3p1
bootcode written to ada3
# zpool status
  pool: mypool
 state: ONLINE
  scan: resilvered 781M in 00:00:16 with 0 errors on Sat Jul 11 09:58:28 2026
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            ada0p3  ONLINE       0     0     0
            ada1p3  ONLINE       0     0     0
          mirror-1  ONLINE       0     0     0
            ada2p3  ONLINE       0     0     0
            ada3p3  ONLINE       0     0     0

errors: No known data errors
```

`zpool add` также присоединяет выделенные устройства журнала и кэша к существующему пулу. Журнал vdev хранит журнал намерений ZFS на отдельном хранилище с низкой задержкой, ускоряя синхронные записи, такие как те, что выполняются базами данных и NFS. Зеркалируйте устройства журнала, потому что потеря незеркалированного устройства журнала вместе с сбоем системы стоит пулу последних нескольких секунд синхронных записей; см. раздел Синхронные записи, ZIL и SLOG для рекомендаций по размеру и настройке. Кэш vdev расширяет ARC вторым уровнем кэша чтения на быстром хранилище. Устройства кэша не требуют избыточности, так как ZFS вместо этого читает любой блок, который не удалось прочитать из кэша, с исходных дисков пула. Содержимое кэша, L2ARC, по умолчанию сохраняется после перезагрузки.

Добавьте зеркальный журнальный vdev и NVMe-устройство кэша к существующему пулу:

```
# zpool add mypool log mirror ada4p2 ada5p2
# zpool add mypool cache nda0p2
# zpool status mypool
  pool: mypool
 state: ONLINE
  scan: resilvered 781M in 00:00:16 with 0 errors on Sat Jul 11 09:58:28 2026
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            ada0p3  ONLINE       0     0     0
            ada1p3  ONLINE       0     0     0
          mirror-1  ONLINE       0     0     0
            ada2p3  ONLINE       0     0     0
            ada3p3  ONLINE       0     0     0
        logs
          mirror-2  ONLINE       0     0     0
            ada4p2  ONLINE       0     0     0
            ada5p2  ONLINE       0     0     0
        cache
          nda0p2    ONLINE       0     0     0

errors: No known data errors
```

Команда `zpool detach` удаляет отдельные диски из зеркального виртуального устройства (vdev) при условии сохранения достаточной избыточности. Если в группе зеркала остаётся единственный диск, эта группа перестает быть зеркалом и становится страйпом, что подвергает весь пул риску в случае выхода из строя оставшегося диска.

Удалить диск из трёхдисковой зеркальной группы:

```
# zpool status
  pool: mypool
 state: ONLINE
  scan: scrub repaired 0B in 00:01:11 with 0 errors on Sat Jul 11 03:14:02 2026
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            ada0p3  ONLINE       0     0     0
            ada1p3  ONLINE       0     0     0
            ada2p3  ONLINE       0     0     0

errors: No known data errors
# zpool detach mypool ada2p3
```

```
# zpool status
  pool: mypool
 state: ONLINE
  scan: scrub repaired 0B in 00:01:11 with 0 errors on Sat Jul 11 03:14:02 2026
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            ada0p3  ONLINE       0     0     0
            ada1p3  ONLINE       0     0     0

errors: No known data errors
```

`zpool remove` удаляет целые vdev верхнего уровня из пула. ZFS эвакуирует vdev, копируя все выделенные ему данные на другие vdev в пуле, а затем отсоединяет диски. Удаление работает для горячих резервов, кэша, журналов, специальных и дедупликационных устройств, а также для data vdev, которые являются одиночными дисками или зеркалами. Data vdev не могут быть удалены из пулов, содержащих vdev RAID-Z верхнего уровня, и все vdev верхнего уровня в пуле должны использовать одинаковый `ashift`. Эвакуация выполняется в фоновом режиме, и пул остается онлайн на протяжении всего процесса; следите за прогрессом с помощью `zpool status`. После завершения удаления ZFS сохраняет таблицу в памяти, сопоставляющую блоки удаленного vdev с их новыми местоположениями. Таблица небольшая, но постоянная; `zpool remove -n` оценивает использование памяти перед началом удаления.

Удалите один из двух зеркальных vdev из пула:

```
# zpool remove mypool mirror-1
# zpool status mypool
  pool: mypool
 state: ONLINE
  scan: scrub repaired 0B in 00:01:11 with 0 errors on Sat Jul 11 03:14:02 2026
remove: Evacuation of mirror in progress since Sat Jul 11 10:15:31 2026
        1.71G copied out of 2.32G at 111M/s, 73.71% done, 0h0m to go
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            ada0p3  ONLINE       0     0     0
            ada1p3  ONLINE       0     0     0
          mirror-1  ONLINE       0     0     0
            ada2p3  ONLINE       0     0     0
            ada3p3  ONLINE       0     0     0

errors: No known data errors
# zpool status mypool
  pool: mypool
```

```
 state: ONLINE
  scan: scrub repaired 0B in 00:01:11 with 0 errors on Sat Jul 11 03:14:02 2026
remove: Removal of vdev 1 copied 2.32G in 0h0m, completed on Sat Jul 11 10:15:53 2026
        10.9K memory used for removed device mappings
config:

        NAME          STATE     READ WRITE CKSUM
        mypool        ONLINE       0     0     0
          mirror-0    ONLINE       0     0     0
            ada0p3    ONLINE       0     0     0
            ada1p3    ONLINE       0     0     0
          indirect-1  ONLINE       0     0     0

errors: No known data errors
```

Запись `indirect-1` является заполнителем для переотображённых блоков удалённого vdev и не содержит дисков. Отмените выполняющееся удаление с помощью `zpool remove -s`.

23.3.4. Расширение RAID-Z

Расширение RAID-Z увеличивает существующий vdev RAID-Z на один диск за раз, не изменяя его уровень чётности.

Расширение RAID-Z требует OpenZFS 2.3 или более поздней версии, впервые доступной в FreeBSD 15.0. Пулы, созданные в более ранних версиях, требуют включения функции `raidz_expansion` с помощью `zpool upgrade` перед расширением.

Чтобы расширить vdev RAID-Z, выполните `zpool attach`, указав имя vdev RAID-Z, как показано в `zpool status`, и новый диск:

```
# zpool attach mypool raidz2-0 ada6p3
```

Расширение перераспределяет существующие данные по увеличенному набору дисков в фоновом режиме, при этом пул остается online и используется. Добавьте `-w`, чтобы заставить `zpool attach` ожидать завершения расширения. `zpool status` сообщает о ходе выполнения в строке `expand:`:

```
# zpool status mypool
  pool: mypool
 state: ONLINE
  scan: scrub repaired 0B in 06:14:36 with 0 errors on Sat Jul 11 04:31:19 2026
expand: expansion of raidz2-0 in progress since Sat Jul 11 09:42:12 2026
        1.83T / 5.36T copied at 186M/s, 34.14% done, 05:31:43 to go
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
```

```
          raidz2-0  ONLINE       0     0     0
            ada0p3  ONLINE       0     0     0
            ada1p3  ONLINE       0     0     0
            ada2p3  ONLINE       0     0     0
            ada3p3  ONLINE       0     0     0
            ada4p3  ONLINE       0     0     0
            ada5p3  ONLINE       0     0     0
            ada6p3  ONLINE       0     0     0

errors: No known data errors
```

Когда расширение завершается, ёмкость нового диска становится доступной:

```
# zpool status mypool | grep expand:
expand: expanded raidz2-0 copied 5.36T in 08:23:35, on Sat Jul 11 18:05:47 2026
```

Блоки, записанные до расширения, сохраняют прежнее соотношение данных и чётности и продолжают занимать место соответствующим образом. Таким образом, полный прирост ёмкости от нового диска применяется к данным, записанным после расширения, тогда как старые данные выигрывают только при перезаписи, например, с помощью `zfs rewrite`. Повторите процедуру для увеличения vdev на несколько дисков, подключая их по одному.

23.3.5. Пулы dRAID

dRAID — это вариант RAID-Z, который распределяет резервную емкость горячего резерва по всем дискам в vdev. dRAID vdev состоит из внутренних групп RAID-Z плюс опциональных *распределенных резервов*, равномерно распределенных по каждому диску. Когда диск выходит из строя, ZFS выполняет перестроение в распределенный резерв, читая и записывая на все оставшиеся диски параллельно, завершая процесс за долю времени, которое требуется для ресилвернинга на один резервный диск. Замена вышедшего из строя диска затем перестраивает его из распределенного резерва, восстанавливая полную защиту. dRAID ориентирован на пулы с множеством дисков, где время перестроения RAID-Z становится опасно большим.

Тип vdev описывает полную компоновку, согласно zpoolconcepts(7):

```
draid[parity][:datad][:childrenc][:spares]
```

parity — это уровень чётности внутренних групп, от 1 до 3, по умолчанию 1. *data* — это количество устройств данных на группу резервирования, по умолчанию 8. *children* — это общее количество дисков, служащее перекрестной проверкой при перечислении многих устройств. *spares* — это количество распределенных горячих резервов, по умолчанию 0.

Создайте пул из 24-дискового vdev dRAID с двойной четностью, 8 дисками данных на группу избыточности и 2 распределенными резервными дисками:

```
# zpool create mypool draid2:8d:24c:2s ada0 ada1 ada2 ada3 ada4 ada5 ada6 ada7 ada8
ada9 ada10 ada11 ada12 ada13 ada14 ada15 ada16 ada17 ada18 ada19 ada20 ada21 ada22
ada23
```

Распределенные запасные диски отображаются в разделе `spares` в `zpool status`, с именами вроде `draid2-0-0` для первого запасного диска первого vdev.

dRAID обменивает эффективность использования пространства на скорость перестроения. В отличие от RAID-Z, dRAID использует фиксированную ширину полосы, дополняя мелкие записи нулями. При стандартных 8 дисках данных и секторах 4 КБ минимальное выделение составляет 32 КБ, что снижает полезную ёмкость и эффективность сжатия для наборов данных, где преобладают мелкие блоки. Ёмкость распределённых запасных дисков также резервируется заранее, независимо от того, произойдёт ли сбой диска когда-либо. Объединение пула dRAID со специальным vdev удерживает метаданные и мелкие блоки вне широких полос и возвращает большую часть потерянной эффективности. Для пулов с несколькими дисками RAID-Z остаётся лучшим выбором.

23.3.6. Специальные классы выделения

Специальные классы выделения привязывают vdev-устройства к конкретным типам данных пула. Vdev special хранит метаданные пула, такие как непрямые блоки и dnode, а также, опционально, блоки данных небольших файлов. Vdev `dedup` хранит таблицы дедупликации. Размещение этих метаданных на быстрых устройствах, таких как зеркала NVMe, ускоряет операции, требовательные к метаданным, например обход каталогов, `zfs list` и поиск при дедупликации в пулах, состоящих в остальном из более медленных дисков.

Добавьте зеркальный специальный vdev к существующему пулу:

```
# zpool add mypool special mirror nda0 nda1
```

Новые выделения метаданных затем направляются в специальный vdev; существующие метаданные остаются на месте, пока не будут перезаписаны.

Установка свойства `special_small_blocks` набора данных позволяет специальному vdev хранить блоки данных размером до указанного значения:

```
# zfs set special_small_blocks=16K mypool/projects
```

Допустимые значения — ноль, который отключает хранение блоков данных в специальном классе, или степень двойки от 512 байт до 1 МБ в FreeBSD 14.x; FreeBSD 15.0 принимает значения вплоть до максимального размера блока в 16 МБ. Установка этого свойства на `recordsize` набора данных отправляет все блоки данных этого набора данных в специальный класс. Когда специальный vdev заполняется, новые выделения перетекают обратно в обычный класс вместо сбоя.

Специальный vdev — это не кэш. Он хранит единственную копию метаданных пула, выделенных ему, и его потеря уничтожает пул. Согласуйте избыточность special и dedup vdev с избыточностью data vdev, например, используя зеркало.

Удаление специального или дедупликационного vdev с помощью `zpool remove` возможно с учётом ограничений, описанных в Добавление и удаление устройств. На практике требование соответствия `ashift` часто препятствует удалению, когда специальный vdev использует устройства с иным размером сектора, чем data vdev.

23.3.7. Проверка состояния пула

Статус пула важен. Если диск отключается или ZFS обнаруживает ошибку чтения, записи или контрольной суммы, соответствующий счетчик ошибок увеличивается. Вывод команды `status` показывает конфигурацию и состояние каждого устройства в пуле, а также состояние всего пула. Также отображаются действия, которые следует предпринять, и детали последнего `scrub`.

```
# zpool status
  pool: mypool
 state: ONLINE
  scan: scrub repaired 0B in 02:25:36 with 0 errors on Sat Jul 11 06:51:26 2026
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          raidz2-0  ONLINE       0     0     0
            ada0p3  ONLINE       0     0     0
            ada1p3  ONLINE       0     0     0
            ada2p3  ONLINE       0     0     0
            ada3p3  ONLINE       0     0     0
            ada4p3  ONLINE       0     0     0
            ada5p3  ONLINE       0     0     0

errors: No known data errors
```

Опции подстраивают вывод под конкретную задачу:

- `-v` выводит подробную информацию об ошибках данных, печатая полный список ошибок, обнаруженных с момента последней полной очистки пула, включая имена затронутых файлов.
- `-x` отображает состояние пулов с ошибками или пулов, которые в противном случае недоступны, скрывая исправные пулы.
- `-s` отображает количество медленных операций ввода-вывода на каждом конечном vdev, то есть операций, которые не завершились в течение 30 секунд.
- `-e` отображает только неисправные vdev, сокращая вывод для пулов с большим количеством устройств; этот флаг требует FreeBSD 14.1 или более поздней версии.

- `-p` отображает числа как точные, разбираемые значения вместо округлённых, удобочитаемых.
- `-t` отображает статус TRIM для каждого vdev.

См. zpool-status(8) для полного списка опций.

23.3.8. Сброс состояния ошибки

При обнаружении ошибки ZFS увеличивает счетчики ошибок чтения, записи или контрольных сумм. Чтобы очистить сообщение об ошибке и сбросить счетчики, используйте команду `zpool clear mypool`. Сброс состояния ошибки может быть важен для автоматизированных скриптов, которые уведомляют администратора при возникновении ошибки в пуле. Без очистки старых ошибок скрипты могут не сообщать о новых ошибках.

23.3.9. Замена рабочего устройства

Может потребоваться заменить один диск на другой. При замене рабочего диска процесс сохраняет старый диск в режиме онлайн во время замены. Пул никогда не переходит в состояние деградировавшего, что снижает риск потери данных. Выполнение команды `zpool replace` копирует данные со старого диска на новый. После завершения операции ZFS отключает старый диск от vdev. Если новый диск больше старого, можно расширить zpool, используя новое пространство. См. Расширение пула.

Заменить работающее устройство в пуле:

```
# zpool status
  pool: mypool
 state: ONLINE
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            ada0p3  ONLINE       0     0     0
            ada1p3  ONLINE       0     0     0

errors: No known data errors
# zpool replace mypool ada1p3 ada2p3
Make sure to wait until resilvering finishes before rebooting.

When booting from the pool 'mypool', update the boot code on the newly attached disk
'ada2p3'.

Assuming GPT partitioning is used and ada0 is the new boot disk, use the following
command:

        gpart bootcode -b /boot/pmbr -p /boot/gptzfsboot -i 1 ada0
# gpart bootcode -b /boot/pmbr -p /boot/gptzfsboot -i 1 ada2
# zpool status
```

```
  pool: mypool
 state: ONLINE
status: One or more devices is currently being resilvered.  The pool will
        continue to function, possibly in a degraded state.
action: Wait for the resilver to complete.
  scan: resilver in progress since Sat Jul 11 14:21:35 2026
        781M / 781M scanned, 604M / 781M issued at 101M/s
        604M resilvered, 77.39% done, 00:00:01 to go
config:

        NAME             STATE     READ WRITE CKSUM
        mypool           ONLINE       0     0     0
          mirror-0       ONLINE       0     0     0
            ada0p3       ONLINE       0     0     0
            replacing-1  ONLINE       0     0     0
              ada1p3     ONLINE       0     0     0
              ada2p3     ONLINE       0     0     0  (resilvering)

errors: No known data errors
# zpool status
  pool: mypool
 state: ONLINE
  scan: resilvered 781M in 00:00:17 with 0 errors on Sat Jul 11 14:21:52 2026
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            ada0p3  ONLINE       0     0     0
            ada2p3  ONLINE       0     0     0

errors: No known data errors
```

На зеркальных vdev-устройствах добавление `-s` к `zpool replace` или `zpool attach` выполняет *последовательное восстановление* вместо восстанавливающего ресильверинга. Последовательное восстановление копирует данные в порядке расположения на диске без проверки контрольных сумм каждого блока, восстанавливая избыточность гораздо быстрее. Поскольку копирование пропускает проверку контрольных сумм, ZFS автоматически запускает scrub пула после завершения последовательного восстановления. Последовательное восстановление работает на зеркальных и dRAID vdev-устройствах, но не на RAID-Z. Перезапустите текущий resilver с начала с помощью `zpool resilver mypool`.

23.3.10. Обработка неисправных устройств

Когда диск в пуле выходит из строя, устройство vdev, к которому принадлежит этот диск, переходит в деградировавшее состояние. Данные остаются доступными, но с пониженной производительностью, поскольку ZFS вычисляет недостающие данные из доступной избыточности. Чтобы восстановить устройство vdev в полностью работоспособное состояние, замените вышедший из строя физический диск. Затем ZFS получает указание

начать операцию восстановления. ZFS пересчитывает данные с вышедшего из строя устройства из доступной избыточности и записывает их на заменённый диск. После завершения процесса устройство vdev возвращается в состояние онлайн.

Если vdev не имеет избыточности или если устройства вышли из строя и недостаточно избыточности для компенсации, пул переходит в состояние faulted. Если недостаточно устройств для восстановления связи, пул становится неработоспособным, что требует восстановления данных из резервных копий.

При замене вышедшего из строя диска имя отказавшего диска изменяется на GUID нового диска. Новый параметр имени устройства для `zpool replace` не требуется, если заменяющее устройство имеет то же имя устройства.

Заменить вышедший из строя диск с помощью `zpool replace`:

```
# zpool status
  pool: mypool
 state: DEGRADED
status: One or more devices could not be opened.  Sufficient replicas exist for
        the pool to continue functioning in a degraded state.
action: Attach the missing device and online it using 'zpool online'.
   see: https://openzfs.github.io/openzfs-docs/msg/ZFS-8000-2Q
config:

        NAME                     STATE     READ WRITE CKSUM
        mypool                   DEGRADED     0     0     0
          mirror-0               DEGRADED     0     0     0
            ada0p3               ONLINE       0     0     0
            316502962686821739   UNAVAIL      0     0     0  was /dev/ada1p3

errors: No known data errors
# zpool replace mypool 316502962686821739 ada2p3
# zpool status
  pool: mypool
 state: DEGRADED
status: One or more devices is currently being resilvered.  The pool will
        continue to function, possibly in a degraded state.
action: Wait for the resilver to complete.
  scan: resilver in progress since Sat Jul 11 14:52:21 2026
        781M / 781M scanned, 641M / 781M issued at 128M/s
        640M resilvered, 82.04% done, 00:00:01 to go
config:

        NAME                         STATE     READ WRITE CKSUM
        mypool                       DEGRADED     0     0     0
          mirror-0                   DEGRADED     0     0     0
            ada0p3                   ONLINE       0     0     0
            replacing-1              UNAVAIL      0     0     0
              15732067398082357289   UNAVAIL      0     0     0  was /dev/ada1p3/old
              ada2p3                 ONLINE       0     0     0  (resilvering)
```

```
errors: No known data errors
# zpool status
  pool: mypool
 state: ONLINE
  scan: resilvered 781M in 00:00:17 with 0 errors on Sat Jul 11 14:52:38 2026
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            ada0p3  ONLINE       0     0     0
            ada2p3  ONLINE       0     0     0

errors: No known data errors
```

23.3.11. Горячие резервные диски и автоматическая замена с помощью zfsd

Пулы с избыточными vdev могут регистрировать горячие резервные диски, которые находятся в режиме ожидания для замены вышедшего из строя устройства. Резервный диск должен быть не меньше устройства, которое он заменяет. Укажите резервные диски при создании пула, перечислив их после ключевого слова `spare`, как в `zpool create mypool mirror ada0p3 ada1p3 spare ada3p3`, или добавьте их в существующий пул с помощью `zpool add`:

```
# zpool add mypool spare ada3p3
# zpool status
  pool: mypool
 state: ONLINE
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            ada0p3  ONLINE       0     0     0
            ada1p3  ONLINE       0     0     0
        spares
          ada3p3    AVAIL

errors: No known data errors
```

Запасной диск остается неиспользуемым, отображаясь как `AVAIL`, пока устройство не выйдет из строя. Активируйте запасной диск вручную с помощью `zpool replace`, указав отказавшее устройство и запасной диск:

```
# zpool replace mypool ada1p3 ada3p3
# zpool status
  pool: mypool
```

```
 state: DEGRADED
status: One or more devices could not be opened.  Sufficient replicas exist for
        the pool to continue functioning in a degraded state.
action: Attach the missing device and online it using 'zpool online'.
   see: https://openzfs.github.io/openzfs-docs/msg/ZFS-8000-2Q
  scan: resilvered 781M in 00:00:21 with 0 errors on Sat Jul 11 09:14:33 2026
config:

        NAME          STATE     READ WRITE CKSUM
        mypool        DEGRADED     0     0     0
          mirror-0    DEGRADED     0     0     0
            ada0p3    ONLINE       0     0     0
            spare-1   DEGRADED     0     0     0
              ada1p3  UNAVAIL      0     0     0  cannot open
              ada3p3  ONLINE       0     0     0
        spares
          ada3p3      INUSE     currently in use

errors: No known data errors
```

Запасной диск является временной мерой, и пул остается в этом состоянии, пока администратор не разрешит ситуацию. После физической замены вышедшего из строя диска и повторного зеркалирования нового устройства с помощью `zpool replace`, верните запасной диск в режим ожидания с помощью `zpool detach mypool ada3p3`. Чтобы вместо этого повысить запасной диск до постоянного члена vdev, отсоедините вышедшее из строя устройство с помощью `zpool detach mypool ada1p3`.

Свойство пула `autoreplace`, по умолчанию выключенное, управляет тем, что происходит, когда новый диск появляется в том же физическом месте, что и устройство, ранее принадлежавшее пулу. При установке `autoreplace` в значение `on`, ZFS автоматически заменяет старое устройство новым диском, не требуя команды `zpool replace`:

```
# zpool set autoreplace=on mypool
```

На FreeBSD автоматические реакции на сбои, включая выполнение `autoreplace`, выполняются zfsd(8), демоном управления отказами ZFS в базовой системе. Демон не включён по умолчанию. Включите и запустите его с помощью команды:

```
# sysrc zfsd_enable="YES"
# service zfsd start
```

zfsd(8) прослушивает события пула и:

- Активирует горячий резерв, когда устройство исчезает из избыточного vdev или когда vdev становится деградировавшим или неисправным.
- Отмечает vdev как отказавший, когда он создает более 50 ошибок ввода-вывода или более 8 отложенных операций ввода-вывода за период в 60 секунд, затем активирует

горячий резерв.

- Помечает vdev как деградировавший, когда он генерирует более 50 ошибок контрольной суммы за 60-секундный период, затем активирует горячий резерв.
- Возвращает устройство в онлайн-режим, когда оно снова появляется, запуская ресильверинг данных, записанных, пока оно отсутствовало.
- Заменяет отсутствующее устройство новым диском, который появляется в том же физическом местоположении, когда в пуле установлено значение `autoreplace` в `on`.
- Возвращает горячие резервы в режим ожидания после завершения ресильвера постоянной замены.

FreeBSD использует zfsd(8) для управления отказами. Демон событий ZFS (ZED), присутствующий на платформах Linux OpenZFS, не является частью FreeBSD.

Изучите поток событий, который управляет этими решениями, с помощью `zpool events`. Добавление `-v` показывает полную полезную нагрузку каждого события, а `-f` оставляет команду работающей, выводя новые события по мере их поступления. ZFS также доставляет эти события в devd(8). Правила в /etc/devd/zfs.conf записывают их в системный журнал и служат шаблоном для пользовательских действий, таких как отправка почты при деградации пула.

23.3.12. Чистка пула

Регулярно выполняйте scrub для пулов, желательно не реже одного раза в месяц. Операция `scrub` интенсивно использует диски и может снизить производительность во время выполнения. Избегайте периодов высокой нагрузки при планировании `scrub` или снизьте его влияние с помощью настраиваемых параметров планировщика ввода-вывода, описанных в разделе «Настройка».

```
# zpool scrub mypool
# zpool status
  pool: mypool
 state: ONLINE
  scan: scrub in progress since Sat Jul 11 20:52:54 2026
        130G / 8.60T scanned at 649M/s, 116G / 8.60T issued at 580M/s
        0B repaired, 1.32% done, 04:15:41 to go
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          raidz2-0  ONLINE       0     0     0
            ada0p3  ONLINE       0     0     0
            ada1p3  ONLINE       0     0     0
            ada2p3  ONLINE       0     0     0
            ada3p3  ONLINE       0     0     0
            ada4p3  ONLINE       0     0     0
            ada5p3  ONLINE       0     0     0
```

```
errors: No known data errors
```

Приостановите выполняющуюся проверку с помощью `zpool scrub -p mypool`, если она мешает другой работе; повторный запуск `zpool scrub mypool` возобновит её с того места, где она остановилась. Чтобы вместо этого отменить операцию проверки, выполните `zpool scrub -s mypool`. Команда `zpool scrub -w mypool` ожидает завершения проверки перед возвратом, что полезно в скриптах. Если после проверки сообщается об ошибках, `zpool scrub -e mypool` выполняет *проверку ошибок*, которая повторно проверяет только блоки с известными ошибками, завершаясь гораздо быстрее, чем полная проверка. Проверка ошибок требует OpenZFS 2.2, впервые включённого в FreeBSD 14.0.

periodic(8) может автоматически запускать проверки. Чтобы проверить каждый пул, последняя проверка которого была более 35 дней назад, добавьте эту строку в /etc/periodic.conf:

```
daily_scrub_zfs_enable="YES"
```

Установите `daily_scrub_zfs_default_threshold="дней"`, чтобы изменить количество дней между проверками. periodic.conf(5) также описывает пороговые значения для отдельных пулов. Установка `daily_status_zfs_enable="YES"` добавляет проверку состояния пула в ежедневный отчёт periodic(8).

23.3.13. Самолечение

Контрольные суммы, хранимые с блоками данных, позволяют файловой системе *самовосстанавливаться*. Эта функция автоматически исправляет данные, контрольная сумма которых не совпадает с записанной на другом устройстве, входящем в состав пула хранения. Например, в конфигурации зеркала с двумя дисками, где один из дисков начинает работать со сбоями и больше не может корректно хранить данные. Это становится ещё хуже, если данные долгое время не были доступны, как в случае долгосрочного архивного хранения. Традиционные файловые системы требуют выполнения команд для проверки и исправления данных, таких как fsck(8). Эти команды занимают время, а в сложных случаях администратору приходится выбирать, какую операцию восстановления выполнить. Когда ZFS обнаруживает блок данных с несовпадающей контрольной суммой, он пытается прочитать данные с зеркального диска. Если этот диск может предоставить корректные данные, ZFS передаст их приложению и исправит данные на диске с ошибочной контрольной суммой. Это происходит без какого-либо вмешательства администратора в ходе обычной работы пула.

Следующий пример демонстрирует самовосстановление при создании зеркального пула из разделов /dev/ada0p1 и /dev/ada1p1.

```
# zpool create healer mirror /dev/ada0p1 /dev/ada1p1
# zpool status healer
  pool: healer
 state: ONLINE
```

```
config:

    NAME        STATE     READ WRITE CKSUM
    healer      ONLINE       0     0     0
      mirror-0  ONLINE       0     0     0
        ada0p1  ONLINE       0     0     0
        ada1p1  ONLINE       0     0     0

errors: No known data errors
# zpool list
NAME     SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP    HEALTH  ALTROOT
healer   960M   111K   960M        -         -     0%     0%  1.00x    ONLINE  -
```

Скопируйте какие-нибудь важные данные в пул для защиты от ошибок данных с использованием функции самовосстановления и создайте контрольную сумму пула для последующего сравнения.

```
# cp /some/important/data /healer
# zpool list
NAME     SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP    HEALTH  ALTROOT
healer   960M  67.7M   892M        -         -     0%     7%  1.00x    ONLINE  -
# sha1 /healer > checksum.txt
# cat checksum.txt
SHA1 (/healer) = 2753eff56d77d9a536ece6694bf0a82740344d1f
```

Симулируйте повреждение данных, записав случайные данные в начало одного из дисков в зеркале. Чтобы предотвратить восстановление данных ZFS при обнаружении, экспортируйте пул перед повреждением и снова импортируйте его после.

Это опасная операция, которая может уничтожить важные данные, и приведена здесь только для демонстрации. **Не пытайтесь** выполнить её во время нормальной работы хранилища данных. Также этот пример преднамеренного повреждения не должен выполняться на диске, содержащем файловую систему, не использующую ZFS на другом разделе. Не используйте имена дисковых устройств, кроме тех, что входят в пул. Убедитесь, что существуют резервные копии пула, и проверьте их перед выполнением команды!

```
# zpool export healer
# dd if=/dev/random of=/dev/ada1p1 bs=1m count=200
200+0 records in
200+0 records out
209715200 bytes transferred in 62.992162 secs (3329227 bytes/sec)
# zpool import healer
```

Статус пула показывает, что на одном устройстве произошла ошибка. Обратите внимание, что приложения, читающие данные из пула, не получили некорректных данных. ZFS

предоставил данные с устройства ada0p1 с правильными контрольными суммами. Чтобы найти устройство с неверной контрольной суммой, ищите то, у которого в столбце `CKSUM` указано ненулевое значение.

```
# zpool status healer
  pool: healer
 state: ONLINE
status: One or more devices has experienced an unrecoverable error.  An
        attempt was made to correct the error.  Applications are unaffected.
action: Determine if the device needs to be replaced, and clear the errors
        using 'zpool clear' or replace the device with 'zpool replace'.
   see: https://openzfs.github.io/openzfs-docs/msg/ZFS-8000-9P
config:

    NAME        STATE     READ WRITE CKSUM
    healer      ONLINE       0     0     0
      mirror-0  ONLINE       0     0     0
        ada0p1  ONLINE       0     0     0
        ada1p1  ONLINE       0     0     1

errors: No known data errors
```

ZFS обнаружил ошибку и обработал её, используя избыточность на неповреждённом зеркальном диске ada0p1. Сравнение контрольных сумм с исходными покажет, восстановлена ли согласованность пула.

```
# sha1 /healer >> checksum.txt
# cat checksum.txt
SHA1 (/healer) = 2753eff56d77d9a536ece6694bf0a82740344d1f
SHA1 (/healer) = 2753eff56d77d9a536ece6694bf0a82740344d1f
```

Генерируйте контрольные суммы до и после намеренного изменения данных, пока данные в пуле ещё совпадают. Это демонстрирует, как ZFS способен автоматически обнаруживать и исправлять любые ошибки, когда контрольные суммы различаются. Обратите внимание, что это возможно только при наличии достаточной избыточности в пуле. Пул, состоящий из одного устройства, не обладает возможностями самовосстановления. Именно поэтому контрольные суммы так важны в ZFS — не отключайте их ни по какой причине. ZFS не требует использования fsck(8) или аналогичной программы проверки целостности файловой системы для обнаружения и исправления ошибок, а также поддерживает доступность пула даже при наличии проблемы. Теперь требуется операция scrub для перезаписи поврежденных данных на ada1p1.

```
# zpool scrub healer
# zpool status healer
  pool: healer
 state: ONLINE
status: One or more devices has experienced an unrecoverable error.  An
```

```
        attempt was made to correct the error.  Applications are unaffected.
action: Determine if the device needs to be replaced, and clear the errors
        using 'zpool clear' or replace the device with 'zpool replace'.
   see: https://openzfs.github.io/openzfs-docs/msg/ZFS-8000-9P
  scan: scrub in progress since Sat Jul 11 10:34:02 2026
        40.8M scanned at 20.4M/s, 24.9M issued at 12.4M/s, 67.0M total
        9.63M repaired, 37.16% done, 00:00:03 to go
config:

    NAME        STATE     READ WRITE CKSUM
    healer      ONLINE       0     0     0
      mirror-0  ONLINE       0     0     0
        ada0p1  ONLINE       0     0     0
        ada1p1  ONLINE       0     0   627  (repairing)

errors: No known data errors
```

Операция scrub читает данные с ada0p1 и перезаписывает любые данные с некорректной контрольной суммой на ada1p1, что отображается как `(repairing)` в выводе `zpool status`. После завершения операции состояние пула изменяется на:

```
# zpool status healer
  pool: healer
 state: ONLINE
status: One or more devices has experienced an unrecoverable error.  An
        attempt was made to correct the error.  Applications are unaffected.
action: Determine if the device needs to be replaced, and clear the errors
        using 'zpool clear' or replace the device with 'zpool replace'.
   see: https://openzfs.github.io/openzfs-docs/msg/ZFS-8000-9P
  scan: scrub repaired 66.5M in 00:00:06 with 0 errors on Sat Jul 11 10:34:08 2026
config:

    NAME        STATE     READ WRITE CKSUM
    healer      ONLINE       0     0     0
      mirror-0  ONLINE       0     0     0
        ada0p1  ONLINE       0     0     0
        ada1p1  ONLINE       0     0 2.72K

errors: No known data errors
```

После завершения операции очистки, когда все данные синхронизированы с ada0p1 на ada1p1, сбросьте сообщения об ошибках из состояния пула, выполнив команду `zpool clear`, как описано в разделе Сброс состояния ошибки.

```
# zpool clear healer
# zpool status healer
  pool: healer
 state: ONLINE
```

```
  scan: scrub repaired 66.5M in 00:00:06 with 0 errors on Sat Jul 11 10:34:08 2026
config:

    NAME        STATE     READ WRITE CKSUM
    healer      ONLINE       0     0     0
      mirror-0  ONLINE       0     0     0
        ada0p1  ONLINE       0     0     0
        ada1p1  ONLINE       0     0     0

errors: No known data errors
```

Пулу возвращено полностью рабочее состояние, все счётчики ошибок теперь обнулены.

23.3.14. TRIM и инициализация

Твердотельные накопители и тонко выделяемое хранилище работают лучше всего, когда ZFS сообщает им, какие блоки больше не используются. `zpool trim` передаёт эту информацию каждому устройству в пуле, которое поддерживает команды TRIM или UNMAP:

```
# zpool trim mypool
```

Выполните усечение одного устройства, указав его имя после пула, как в `zpool trim mypool nda0p3`. `zpool trim -w` ожидает завершения усечения, `-c` отменяет выполняющееся усечение, а `-s` приостанавливает его; повторный запуск `zpool trim` возобновляет приостановленное усечение. На устройствах, поддерживающих эту возможность, `zpool trim --secure` запрашивает безопасное усечение, при котором устройство гарантирует стирание данных, хранящихся на усечённых блоках.

`zpool status -t` показывает прогресс trim для каждого устройства:

```
# zpool status -t mypool
  pool: mypool
 state: ONLINE
  scan: scrub repaired 0B in 00:12:33 with 0 errors on Sat Jul 11 03:31:11 2026
config:

    NAME        STATE     READ WRITE CKSUM
    mypool      ONLINE       0     0     0
      mirror-0  ONLINE       0     0     0
        nda0p3  ONLINE       0     0     0  (23% trimmed, started at Sat Jul 11
09:15:04 2026)
        nda1p3  ONLINE       0     0     0  (22% trimmed, started at Sat Jul 11
09:15:04 2026)

errors: No known data errors
```

Установка свойства пула `autotrim` в значение `on` заставляет ZFS непрерывно отправлять

небольшие команды TRIM по мере освобождения пространства пулом:

```
# zpool set autotrim=on mypool
```

Автоматическая утилизация пропускает небольшие освобожденные области и создает постоянную дополнительную нагрузку на устройства. Для большинства систем отключение `autotrim` и выполнение полной команды `zpool trim` по расписанию дает лучшие результаты. Начиная с FreeBSD 14.1 и более поздних версий, periodic(8) выполняет такую утилизацию ежедневно, если в /etc/periodic.conf содержится `daily_trim_zfs_enable="YES"`.

`zpool initialize` является аналогом процедуры усечения: она записывает шаблон во всё нераспределённое пространство на указанных устройствах или на каждое подходящее устройство в пуле, если имена устройств не заданы. Это заставляет нижележащее хранилище выделить пространство заранее, избегая задержек при первой записи на тонко подготовленные виртуальные диски, а также проверяет новые устройства перед тем, как доверить им данные.

```
# zpool initialize mypool
```

Как и в случае с усечением, `-c` отменяет инициализацию, `-s` приостанавливает её, а `-w` ожидает её завершения; `-u` удаляет состояние инициализации с устройств. Подробнее см. zpool-trim(8) и zpool-initialize(8).

23.3.15. Контрольные точки пула

`zpool checkpoint` сохраняет состояние всего пула, включая каждый набор данных и конфигурацию пула, так что последующий откат возвращает пул к точному состоянию, которое он имел при создании контрольной точки. Создавайте контрольную точку перед рискованными административными операциями, такими как обновление операционной системы, включение новых функций пула или реорганизация наборов данных:

```
# zpool checkpoint mypool
```

Пул может хранить не более одной контрольной точки одновременно. Столбец `CKPOINT` команды `zpool list` показывает, сколько места занимает контрольная точка по мере расхождения пула с сохранённым состоянием:

```
# zpool list mypool
NAME     SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP    HEALTH  ALTROOT
mypool  4.50T  1.42T  3.08T     221M         -     4%    31%  1.00x    ONLINE  -
```

Чтобы откатить изменения, экспортируйте пул и импортируйте его снова с помощью `--rewind-to-checkpoint`:

```
# zpool export mypool
# zpool import --rewind-to-checkpoint mypool
```

Перемотка (rewinding) отбрасывает все изменения, внесенные в пул после создания контрольной точки, включая более новые снимки и изменения свойств. Удалите контрольную точку, которая больше не нужна, чтобы освободить занимаемое ею пространство; `-w` ожидает завершения удаления:

```
# zpool checkpoint -d -w mypool
```

Пока существует контрольная точка, ZFS отказывает в операциях, изменяющих структуру пула: `zpool remove`, `zpool attach`, `zpool detach`, `zpool split` и `zpool reguid`. Пространство, освобожденное после создания контрольной точки, не возвращается, пока контрольная точка не будет удалена, поэтому долгоживущая контрольная точка на активном пуле может заполнить его.

Контрольная точка не является заменой приведенных в снимков или резервных копий. Она защищает от административных ошибок в течение короткого периода, располагается на тех же дисках, что и пул, а откат отбрасывает все, записанное после ее создания.

См. zpool-checkpoint(8) для подробностей.

23.3.16. Увеличение размера пула

Наименьшее устройство в каждом vdev ограничивает полезный размер избыточного пула. Замените наименьшее устройство на устройство большего размера. После завершения операции замены или пересинхронизации пул может расшириться для использования ёмкости нового устройства. Например, рассмотрим зеркало из диска на 1 ТБ и диска на 2 ТБ. Полезное пространство составляет 1 ТБ. При замене диска на 1 ТБ другим диском на 2 ТБ процесс пересинхронизации копирует существующие данные на новый диск. Поскольку оба устройства теперь имеют ёмкость 2 ТБ, доступное пространство зеркала увеличивается до 2 ТБ.

Пул не использует новую ёмкость автоматически по умолчанию. Установите свойство пула `autoexpand` в значение `on` перед заменой устройств, и пул увеличится, как только последнее устройство меньшего размера покинет vdev:

```
# zpool set autoexpand=on mypool
```

Иначе, начните расширение, используя `zpool online -e` для каждого устройства. После расширения всех устройств дополнительное пространство становится доступным для пула.

Чтобы увеличить размер vdev RAID-Z, добавив диск вместо замены существующих, см. Расширение RAID-Z.

23.3.17. Импорт и экспорт пулов

Экспортируйте пулы перед перемещением на другую систему. ZFS отмонтирует все наборы данных, помечая каждое устройство как экспортированное, но все ещё заблокированное, чтобы предотвратить использование другими дисками. Это позволяет импортировать пулы на других машинах, операционных системах с поддержкой ZFS и даже на аппаратных архитектурах другого типа (с некоторыми оговорками, см. zpool(8)). Если в наборе данных есть открытые файлы, используйте `zpool export -f` для принудительного экспорта пула. Используйте эту команду с осторожностью. Наборы данных будут принудительно отмонтированы, что может привести к неожиданному поведению приложений, работавших с открытыми файлами в этих наборах данных.

Экспортируйте пул, который не используется:

```
# zpool export mypool
```

Импорт пула автоматически монтирует наборы данных. Если такое поведение нежелательно, используйте `zpool import -N`, чтобы предотвратить это. `zpool import -o` устанавливает временные свойства для данного конкретного импорта. `zpool import -o altroot=` позволяет импортировать пул с базовой точкой монтирования вместо корня файловой системы. Если пул последний раз использовался в другой системе и не был корректно экспортирован, принудительно импортируйте его с помощью `zpool import -f`. `zpool import -a` импортирует все пулы, которые, по-видимому, не используются другой системой.

zpool-import(8) также предоставляет опции для восстановления поврежденных пулов. `zpool import -F` пытается откатить пул к более ранней группе транзакций, когда последние повреждены, отбрасывая последние несколько секунд записей. `zpool import -m` позволяет импортировать пул, у которого отсутствует выделенное устройство журнала. `zpool import -o readonly=on` импортирует пул в режиме только для чтения, предотвращая все записи, что полезно для анализа последствий или при спасении данных из отказывающего пула. `zpool import -R /mnt` объединяет `altroot=/mnt` с `cachefile=none`, сохраняя спасательный импорт от нарушения путей монтирования и кэша пулов работающей системы.

Выведите список всех доступных пулов для импорта:

```
# zpool import
   pool: mypool
     id: 9930174748043525076
  state: ONLINE
 action: The pool can be imported using its name or numeric identifier.
 config:

        mypool    ONLINE
          ada0p1  ONLINE
```

Импортируйте пул с альтернативным корневым каталогом:

```
# zpool import -o altroot=/mnt mypool
# zfs list
NAME                 USED  AVAIL  REFER  MOUNTPOINT
mypool               110K  47.0G    31K  /mnt/mypool
```

При загрузке сценарии запуска ZFS, включённые через `zfs_enable="YES"` в /etc/rc.conf, импортируют каждый пул, записанный в /etc/zfs/zpool.cache. Импорт пула автоматически добавляет его в этот файл кэша; свойство пула `cachefile` управляет этим поведением. Пулы, импортированные с `cachefile=none`, не восстанавливаются автоматически после перезагрузки.

Импорт пула, зашифрованного GELI

Опция `Encrypt Disks` инсталлятора FreeBSD шифрует разделы пула с помощью geli(8), описанного в Шифрование разделов диска, а не с помощью собственного шифрования ZFS. На установленной системе загрузчик запрашивает парольную фразу и подключает провайдеров перед монтированием корневого пула, но сама ZFS ничего не знает о шифровании: при загрузке с аварийного носителя или переносе дисков на другой хост `zpool import` не находит пул, пока провайдеры GELI не будут подключены.

Присоедините зашифрованный freebsd-zfs раздел каждого диска в пуле, указав парольную фразу, затем импортируйте пул:

```
# geli attach ada0p4
Enter passphrase:
# zpool import -f -R /mnt zroot
```

Используйте `gpart show`, чтобы определить разделы freebsd-zfs; при стандартной раскладке установщика зашифрованный раздел — это p4. Системы, установленные с устаревшей раскладкой, которая хранит отдельный незашифрованный загрузочный пул, дополнительно содержат файл ключа в /boot/encryption.key в этом пуле; подключите эти провайдеры с помощью `geli attach -k /path/to/encryption.key`.

23.3.18. Обновление пула дисков

После обновления FreeBSD или при импорте пула с системы, использующей более старую версию, вручную обновите пул, чтобы активировать поддержку новых функций. Перед обновлением учтите, может ли пул потребоваться для импорта на более старой системе. Обновление является необратимым процессом. Обновление старых пулов возможно, но понижение версии пулов с новыми функциями — нет.

Пулы больше не имеют единого номера версии. Вместо этого каждое изменение формата на диске представляет собой отдельный *признак* (feature flag), описанный в zpool-features(7). Признак *включается* (enabled), когда пул разрешает его использование, и становится *активным* (active), как только пул сохраняет данные, зависящие от него. Включённый, но неактивный признак не влияет на совместимость. Как только признак становится активным, программное обеспечение без его поддержки больше не может импортировать

пул, хотя пулы, все активные признаки которых являются «совместимыми только для чтения», всё ещё допускают импорт в режиме только для чтения.

`zpool status` сообщает, когда поддерживаемые возможности не включены в пуле:

```
# zpool status mypool
  pool: mypool
 state: ONLINE
status: Some supported and requested features are not enabled on the pool.
        The pool can still be used, but some features are unavailable.
action: Enable all features using 'zpool upgrade'. Once this is done,
        the pool may no longer be accessible by software that does not support
        the features. See zpool-features(7) for details.
  scan: scrub repaired 0B in 00:11:03 with 0 errors on Sat Jul 11 03:15:41 2026
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            ada0p1  ONLINE       0     0     0
            ada1p1  ONLINE       0     0     0

errors: No known data errors
```

Выполнение `zpool upgrade` без аргументов выводит список всех пулов с отключёнными возможностями:

```
# zpool upgrade
This system supports ZFS pool feature flags.

Some supported features are not enabled on the following pools. Once a
feature is enabled the pool may become incompatible with software
that does not support the feature. See zpool-features(7) for details.

Обратите внимание, что функция «совместимость» (compatibility) пула может
использоваться для предотвращения
обновлений функциональных возможностей.

POOL  FEATURE
---------------
mypool
      zilsaxattr
      head_errlog
      blake3
      block_cloning
      vdev_zaps_v2
```

Используйте `zpool upgrade -v`, чтобы вывести список всех функций, поддерживаемых

работающей системой, вместе с устаревшими номерами версий. Включите все поддерживаемые функции в пуле:

```
# zpool upgrade mypool
This system supports ZFS pool feature flags.

Enabled the following features on 'mypool':
  zilsaxattr
  head_errlog
  blake3
  block_cloning
  vdev_zaps_v2
```

Пулы, которые должны оставаться импортируемыми на системах, работающих с более старым программным обеспечением ZFS, могут ограничивать, какие возможности `zpool create` и `zpool upgrade` включают, через свойство пула `compatibility`:

```
# zpool create -o compatibility=openzfs-2.1-freebsd mypool mirror /dev/ada0p1
/dev/ada1p1
```

Установка свойства на существующем пуле с помощью `zpool set` также работает. Каждый именованный набор функций соответствует файлу в /usr/share/zfs/compatibility.d, и значение по умолчанию `off` позволяет использовать все функции. Установка свойства не отключает уже включённые функции.

Обновите загрузочный код на системах, которые загружаются с пула, чтобы поддержать новые свойства пула. Используйте `gpart bootcode` для раздела, содержащего загрузочный код. Доступны два типа загрузочного кода в зависимости от способа загрузки системы: GPT (наиболее распространённый вариант) и EFI (для более современных систем).

Для загрузки в устаревшем режиме с использованием GPT используйте следующую команду:

```
# gpart bootcode -b /boot/pmbr -p /boot/gptzfsboot -i 1 ada1
```

Для систем, использующих EFI для загрузки, выполните следующую команду:

```
# mount -t msdosfs /dev/ada1p1 /boot/efi
# cp /boot/loader.efi /boot/efi/efi/boot/BOOTX64.efi
# cp /boot/loader.efi /boot/efi/efi/freebsd/loader.efi
```

Данные инструкции применимы к системам amd64. В зависимости от архитектуры файл имеет другое имя. Обратитесь к uefi(8), чтобы узнать

имена файлов, специфичных для конкретной архитектуры.

Примените загрузочный код ко всем загрузочным дискам в пуле. Подробнее см. в gpart(8).

23.3.19. Отображение записанной истории пулов

ZFS записывает команды, которые изменяют пул, включая создание наборов данных, изменение свойств или замену диска. Просмотр истории создания пула полезен, так же как и проверка того, какой пользователь выполнил конкретное действие и когда. История не хранится в файле журнала, а является частью самого пула. Команда для просмотра этой истории называется `zpool history`:

```
# zpool history
History for 'tank':
2026-07-11.10:24:05 zpool create tank mirror /dev/ada0p1 /dev/ada1p1
2026-07-11.18:50:58 zfs set atime=off tank
2026-07-11.18:51:09 zfs set checksum=fletcher4 tank
2026-07-11.18:51:18 zfs create tank/backup
```

Вывод показывает команды `zpool` и `zfs`, изменяющие пул каким-либо образом, вместе с временной меткой. Команды вроде `zfs list` не включаются. Если имя пула не указано, ZFS отображает историю всех пулов.

`zpool history` может отображать ещё больше информации при использовании опций `-i` или `-l`. Опция `-i` показывает события, инициированные пользователем, а также внутренние события ZFS, зарегистрированные в журнале.

```
# zpool history -i
History for 'tank':
2026-07-11.10:24:05 [txg:5] create pool version 5000; software version zfs-2.2.7-
FreeBSD_gd75f9ee8b; uts myzfsbox 14.3-RELEASE 1403000 amd64
2026-07-11.18:50:53 [txg:50] set tank (21) atime=0
2026-07-11.18:50:58 zfs set atime=off tank
2026-07-11.18:51:04 [txg:53] set tank (21) checksum=7
2026-07-11.18:51:09 zfs set checksum=fletcher4 tank
2026-07-11.18:51:13 [txg:55] create tank/backup (39)
2026-07-11.18:51:18 zfs create tank/backup
```

Показать более подробную информацию, добавив `-l`. Отображение записей истории в длинном формате, включая такие сведения, как имя пользователя, выполнившего команду, и имя хоста, на котором произошло изменение.

```
# zpool history -l
History for 'tank':
2026-07-11.10:24:05 zpool create tank mirror /dev/ada0p1 /dev/ada1p1 [user 0 (root) on
myzfsbox:global]
```

```
2026-07-11.18:50:58 zfs set atime=off tank [user 0 (root) on myzfsbox:global]
2026-07-11.18:51:09 zfs set checksum=fletcher4 tank [user 0 (root) on myzfsbox:global]
2026-07-11.18:51:18 zfs create tank/backup [user 0 (root) on myzfsbox:global]
```

Вывод показывает, что пользователь `root` создал зеркальный пул с дисками /dev/ada0p1 и /dev/ada1p1. Также в каждой команде отображается имя хоста `myzfsbox`. Отображение имени хоста становится важным при экспорте пула с одной системы и импорте на другую. Можно различить команды, выполненные на другой системе, по имени хоста, записанному для каждой команды.

Объедините оба варианта с `zpool history`, чтобы получить максимально детальную информацию для любого заданного пула. История пула предоставляет ценную информацию при отслеживании выполненных действий или необходимости более детального вывода для отладки.

23.3.20. Мониторинг производительности

Встроенная система мониторинга может отображать статистику операций ввода-вывода пула в реальном времени. Она показывает объём свободного и занятого пространства в пуле, количество операций чтения и записи в секунду, а также используемую пропускную способность ввода-вывода. По умолчанию ZFS отслеживает и отображает все пулы в системе. Укажите имя пула, чтобы ограничить мониторинг только этим пулом. Простой пример:

```
# zpool iostat
               capacity     operations    bandwidth
pool        alloc   free   read  write   read  write
----------  -----  -----  -----  -----  -----  -----
data         288G  1.53T      2     11  11.3K  57.1K
```

Чтобы непрерывно отслеживать активность ввода-вывода, укажите число в качестве последнего параметра, задающее интервал в секундах между обновлениями. Следующая строка статистики выводится после каждого интервала. Нажмите `Ctrl` + `C`, чтобы остановить непрерывный мониторинг. Укажите второе число в командной строке после интервала, чтобы задать общее количество отображаемой статистики.

Отображать более детальную статистику ввода-вывода с помощью `-v`. Каждое устройство в пуле отображается с отдельной строкой статистики. Это полезно для просмотра операций чтения и записи, выполняемых на каждом устройстве, и может помочь определить, замедляет ли какое-либо отдельное устройство работу пула. В этом примере показан зеркальный пул с двумя устройствами:

```
# zpool iostat -v
                           capacity     operations    bandwidth
pool                     alloc   free   read  write   read  write
-----------------------  -----  -----  -----  -----  -----  -----
data                      288G  1.53T      2     12  9.23K  61.5K
```

```
  mirror                  288G  1.53T      2     12  9.23K  61.5K
    ada1p1                   -      -      0      4  5.61K  61.7K
    ada2p1                   -      -      1      4  5.04K  61.7K
----------------------  -----  -----  -----  -----  -----  -----
```

`zpool iostat` предоставляет несколько более детальных представлений о конвейере ввода-вывода. `-l` (latency) добавляет столбцы со средней задержкой, разделяя общее время ожидания на время на диске и время, проведенное в различных очередях ввода-вывода. `-q` (queue) показывает количество ожидающих и активных операций в каждой очереди на vdev. `-w` выводит полные гистограммы задержек, а `-r` (request size) выводит гистограммы размеров запросов для каждого листового vdev.

Столбцы задержки в `-l` позволяют легко заметить один медленный диск, который замедляет работу всего vdev:

```
# zpool iostat -lv data
              capacity     operations     bandwidth    total_wait     disk_wait
syncq_wait    asyncq_wait  scrub   trim  rebuild
pool        alloc   free   read  write   read  write   read  write   read  write
read  write   read  write   wait   wait     wait
----------  -----  -----  -----  -----  -----  -----  -----  -----  -----  -----
-----  -----  -----  -----  -----  -----  -------
data         288G  1.53T    156    212  19.4M  26.6M   14ms  186ms    5ms  184ms
1us    2us    8ms    2ms      -      -        -
  mirror-0   288G  1.53T    156    212  19.4M  26.6M   14ms  186ms    5ms  184ms
1us    2us    8ms    2ms      -      -        -
    ada1p1      -      -     79    106   9.7M  13.3M    3ms    5ms    3ms    4ms
1us    2us    1ms    1ms      -      -        -
    ada2p1      -      -     77    106   9.7M  13.3M  212ms  187ms  208ms  183ms
1us    2us   15ms    3ms      -      -        -
----------  -----  -----  -----  -----  -----  -----  -----  -----  -----  -----
-----  -----  -----  -----  -----  -----  -------
```

В этом примере ada2p1 требует сотни миллисекунд дискового времени для операций ввода-вывода, которые его зеркальный партнёр ada1p1 выполняет за однозначные миллисекунды, при этом оба обрабатывают схожее количество операций. Запись в зеркало завершается только после того, как завершится самый медленный участник, поэтому задержка записи всего vdev отслеживает ada2p1. Высокая задержка при нормальной работе, без ошибок чтения, записи или контрольных сумм в `zpool status`, часто указывает на диск, который выполняет внутренние повторы, и является кандидатом на замену.

Скрипты, которым необходимо дождаться завершения фоновой активности пула, могут блокироваться на `zpool wait`:

```
# zpool wait -t scrub,resilver mypool
```

Команда возвращает управление, как только ни одно из указанных действий, таких как

`scrub`, `resilver`, `trim` или `remove`, не выполняется. Полный список действий приведён в zpool-wait(8).

`zpool status`, `zpool list`, `zpool get` и `zfs list` принимают `-j` для вывода JSON, который более устойчив для разбора скриптами, чем таблицы, читаемые человеком. Вывод JSON требует OpenZFS 2.3 или новее, впервые доступный во FreeBSD 15.0.

23.3.21. Разделение пула хранения данных

ZFS может разделить пул, состоящий из одного или нескольких зеркальных vdev, на два пула. Если не указано иное, ZFS отсоединяет последний элемент каждого зеркала и создаёт новый пул с теми же данными. Обязательно выполните пробный запуск операции с параметром `-n` сначала. Это отобразит детали запрошенной операции без её фактического выполнения. Это помогает убедиться, что операция выполнит то, что задумал пользователь.

23.4. Управление с помощью утилиты `zfs`

Утилита `zfs` позволяет создавать, удалять и управлять всеми существующими наборами данных ZFS в пределах пула. Для управления самим пулом используйте `zpool`.

23.4.1. Создание и удаление наборов данных

В отличие от традиционных дисков и менеджеров томов, пространство в ZFS *не* выделяется заранее. В традиционных файловых системах после разметки и выделения пространства невозможно добавить новую файловую систему без добавления нового диска. В ZFS создание новых файловых систем возможно в любое время. Каждый *набор данных* обладает свойствами, включая такие функции, как сжатие, дедупликация, кэширование и квоты, а также другие полезные свойства, такие как режим только для чтения, чувствительность к регистру, сетевое общее использование файлов и точка монтирования. Возможно вложение наборов данных друг в друга, при этом дочерние наборы данных наследуют свойства от своих родительских. Делегирование, репликация, снимки, клетки позволяют администрировать и уничтожать каждый набор данных как единое целое. Создание отдельного набора данных для каждого типа или группы файлов имеет свои преимущества. Недостатком наличия большого количества наборов данных является то, что некоторые команды, такие как `zfs list`, будут выполняться медленнее, а монтирование сотен или даже тысяч наборов данных замедлит процесс загрузки FreeBSD.

Создайте новый набор данных и включите для него сжатие LZ4:

```
# zfs list
NAME                  USED  AVAIL  REFER  MOUNTPOINT
mypool                781M  93.2G   144K  none
mypool/ROOT           777M  93.2G   144K  none
mypool/ROOT/default   777M  93.2G   777M  /
mypool/tmp            176K  93.2G   176K  /tmp
mypool/usr            616K  93.2G   144K  /usr
```

```
mypool/usr/home       184K  93.2G   184K  /usr/home
mypool/usr/ports      144K  93.2G   144K  /usr/ports
mypool/usr/src        144K  93.2G   144K  /usr/src
mypool/var           1.20M  93.2G   608K  /var
mypool/var/crash      148K  93.2G   148K  /var/crash
mypool/var/log        178K  93.2G   178K  /var/log
mypool/var/mail       144K  93.2G   144K  /var/mail
mypool/var/tmp        152K  93.2G   152K  /var/tmp
# zfs create -o compress=lz4 mypool/usr/mydataset
# zfs list
NAME                   USED  AVAIL  REFER  MOUNTPOINT
mypool                 781M  93.2G   144K  none
mypool/ROOT            777M  93.2G   144K  none
mypool/ROOT/default    777M  93.2G   777M  /
mypool/tmp             176K  93.2G   176K  /tmp
mypool/usr             704K  93.2G   144K  /usr
mypool/usr/home        184K  93.2G   184K  /usr/home
mypool/usr/mydataset  87.5K  93.2G  87.5K  /usr/mydataset
mypool/usr/ports       144K  93.2G   144K  /usr/ports
mypool/usr/src         144K  93.2G   144K  /usr/src
mypool/var            1.20M  93.2G   610K  /var
mypool/var/crash       148K  93.2G   148K  /var/crash
mypool/var/log         178K  93.2G   178K  /var/log
mypool/var/mail        144K  93.2G   144K  /var/mail
mypool/var/tmp         152K  93.2G   152K  /var/tmp
```

Уничтожение набора данных выполняется гораздо быстрее, чем удаление файлов в наборе данных, так как не требует сканирования файлов и обновления соответствующих метаданных.

Уничтожьте созданный набор данных:

```
# zfs list
NAME                   USED  AVAIL  REFER  MOUNTPOINT
mypool                 880M  93.1G   144K  none
mypool/ROOT            777M  93.1G   144K  none
mypool/ROOT/default    777M  93.1G   777M  /
mypool/tmp             176K  93.1G   176K  /tmp
mypool/usr             101M  93.1G   144K  /usr
mypool/usr/home        184K  93.1G   184K  /usr/home
mypool/usr/mydataset   100M  93.1G   100M  /usr/mydataset
mypool/usr/ports       144K  93.1G   144K  /usr/ports
mypool/usr/src         144K  93.1G   144K  /usr/src
mypool/var            1.20M  93.1G   610K  /var
mypool/var/crash       148K  93.1G   148K  /var/crash
mypool/var/log         178K  93.1G   178K  /var/log
mypool/var/mail        144K  93.1G   144K  /var/mail
mypool/var/tmp         152K  93.1G   152K  /var/tmp
# zfs destroy mypool/usr/mydataset
# zfs list
```

```
NAME                  USED  AVAIL  REFER  MOUNTPOINT
mypool                781M  93.2G   144K  none
mypool/ROOT           777M  93.2G   144K  none
mypool/ROOT/default   777M  93.2G   777M  /
mypool/tmp            176K  93.2G   176K  /tmp
mypool/usr            616K  93.2G   144K  /usr
mypool/usr/home       184K  93.2G   184K  /usr/home
mypool/usr/ports      144K  93.2G   144K  /usr/ports
mypool/usr/src        144K  93.2G   144K  /usr/src
mypool/var           1.21M  93.2G   612K  /var
mypool/var/crash      148K  93.2G   148K  /var/crash
mypool/var/log        178K  93.2G   178K  /var/log
mypool/var/mail       144K  93.2G   144K  /var/mail
mypool/var/tmp        152K  93.2G   152K  /var/tmp
```

В современных версиях ZFS команда `zfs destroy` выполняется асинхронно, и освобождённое пространство может появиться в пуле только через несколько минут. Используйте команду `zpool get freeing poolname` для просмотра свойства `freeing`, которое показывает, сколько места ещё остаётся освободить в фоновом режиме. Используйте команду `zpool wait -t free poolname`, чтобы дождаться завершения фонового освобождения. Если существуют дочерние наборы данных, например снимки или другие наборы данных, уничтожение родительского набора невозможно. Для удаления набора данных и его дочерних элементов используйте `-r`, чтобы рекурсивно удалить набор данных и его потомков. Опция `-n -v` позволяет вывести список наборов данных и снимков, которые будут удалены данной операцией, без фактического выполнения удаления. Также отображается пространство, которое будет освобождено после удаления снимков.

23.4.2. Создание и удаление томов

Том — это особый тип набора данных. Вместо монтирования в качестве файловой системы он представляется как блочное устройство в /dev/zvol/имя_пула/набор_данных. Это позволяет использовать том для других файловых систем, в качестве дисков для виртуальной машины или сделать его доступным для других сетевых узлов с использованием таких протоколов, как iSCSI или HAST.

Отформатируйте том с любой файловой системой или без файловой системы для хранения сырых данных. Для пользователя том выглядит как обычный диск. Размещение обычных файловых систем на этих *zvols* предоставляет возможности, которых нет у обычных дисков или файловых систем. Например, использование свойства сжатия на томе объёмом 250 МБ позволяет создать сжатую файловую систему FAT.

```
# zfs create -V 250m -o compression=on tank/fat32
# zfs list tank
NAME USED AVAIL REFER MOUNTPOINT
tank 258M  670M   31K /tank
# newfs_msdos -F32 /dev/zvol/tank/fat32
# mount -t msdosfs /dev/zvol/tank/fat32 /mnt
# df -h /mnt | grep fat32
Filesystem           Size Used Avail Capacity Mounted on
```

```
/dev/zvol/tank/fat32 249M  24k  249M     0%   /mnt
# mount | grep fat32
/dev/zvol/tank/fat32 on /mnt (msdosfs, local)
```

Создание тома резервирует в пуле достаточно места для хранения полного размера тома. Чтобы пропустить это резервирование, создайте *разреженный том* (sparse volume), также известный как тонко выделенный том, с помощью `-s`:

```
# zfs create -s -V 250g tank/bigdisk
```

Разреженные тома позволяют выделять больше пространства, чем фактически имеется в пуле. Записи в разреженный том могут завершиться ошибкой `ENOSPC`, когда в пуле заканчивается место, поэтому используйте разреженные тома только там, где потребитель тома допускает ошибки записи.

Свойство `volmode` управляет тем, как ZFS предоставляет том операционной системе. Установка значения `full` предоставляет том как провайдер GEOM с максимальной функциональностью, а `geom` является псевдонимом для `full`. Установка значения `dev` предоставляет том как обычный узел устройства, скрывая любые разделы, которые он содержит. Тома с `volmode`, установленным в `none`, не предоставляются за пределами ZFS вообще, но по-прежнему поддерживают снимки, клоны и репликацию, что делает их полезными в качестве целей репликации. Свойство по умолчанию имеет значение `default`, которое откладывается на системную настройку `zvol_volmode`, описанную в zfs(4).

Тома хорошо работают в качестве виртуальных дисков для виртуальных машин bhyve(8) и как хранилище для экстентов iSCSI: объявите устройство тома как LUN в ctl.conf(5) и обслуживайте его с помощью ctld(8).

Не размещайте пространство подкачки на томах ZFS. Когда в системе недостаточно памяти, сам ZFS нуждается в памяти для завершения записи, которая освободила бы память, и система может зависнуть. Подробности см. в upstream-проблеме OpenZFS openzfs/zfs#7734. Вместо этого используйте выделенный раздел подкачки.

Уничтожение тома во многом аналогично уничтожению обычного набора данных файловой системы. Операция выполняется почти мгновенно, но освобождение места может занять несколько минут и происходит в фоновом режиме.

23.4.3. Переименование набора данных

Для изменения имени набора данных используйте `zfs rename`. Эту же команду можно использовать для изменения родительского набора данных. Переименование набора данных с изменением родительского набора приведёт к изменению значений свойств, унаследованных от родительского набора данных. Переименование набора данных размонтирует, а затем снова смонтирует его в новом месте (с учётом наследования от нового родительского набора данных). Чтобы предотвратить это поведение, используйте опцию `-u`.

Переименовать набор данных и переместить его под другой родительский набор данных:

```
# zfs list
NAME                   USED  AVAIL  REFER  MOUNTPOINT
mypool                 780M  93.2G   144K  none
mypool/ROOT            777M  93.2G   144K  none
mypool/ROOT/default    777M  93.2G   777M  /
mypool/tmp             176K  93.2G   176K  /tmp
mypool/usr             704K  93.2G   144K  /usr
mypool/usr/home        184K  93.2G   184K  /usr/home
mypool/usr/mydataset  87.5K  93.2G  87.5K  /usr/mydataset
mypool/usr/ports       144K  93.2G   144K  /usr/ports
mypool/usr/src         144K  93.2G   144K  /usr/src
mypool/var            1.21M  93.2G   614K  /var
mypool/var/crash       148K  93.2G   148K  /var/crash
mypool/var/log         178K  93.2G   178K  /var/log
mypool/var/mail        144K  93.2G   144K  /var/mail
mypool/var/tmp         152K  93.2G   152K  /var/tmp
# zfs rename mypool/usr/mydataset mypool/var/newname
# zfs list
NAME                  USED  AVAIL  REFER  MOUNTPOINT
mypool                780M  93.2G   144K  none
mypool/ROOT           777M  93.2G   144K  none
mypool/ROOT/default   777M  93.2G   777M  /
mypool/tmp            176K  93.2G   176K  /tmp
mypool/usr            616K  93.2G   144K  /usr
mypool/usr/home       184K  93.2G   184K  /usr/home
mypool/usr/ports      144K  93.2G   144K  /usr/ports
mypool/usr/src        144K  93.2G   144K  /usr/src
mypool/var           1.29M  93.2G   614K  /var
mypool/var/crash      148K  93.2G   148K  /var/crash
mypool/var/log        178K  93.2G   178K  /var/log
mypool/var/mail       144K  93.2G   144K  /var/mail
mypool/var/newname   87.5K  93.2G  87.5K  /var/newname
mypool/var/tmp        152K  93.2G   152K  /var/tmp
```

Переименование снимков выполняется той же командой. Из-за особенностей снимков, их переименование не может изменить родительский набор данных. Для рекурсивного переименования снимка укажите `-r`; это также переименует все снимки с таким же именем в дочерних наборах данных.

```
# zfs list -t snapshot
NAME                                 USED  AVAIL  REFER  MOUNTPOINT
mypool/var/newname@first_snapshot       0      -  87.5K  -
# zfs rename mypool/var/newname@first_snapshot new_snapshot_name
# zfs list -t snapshot
NAME                                    USED  AVAIL  REFER  MOUNTPOINT
mypool/var/newname@new_snapshot_name       0      -  87.5K  -
```

23.4.4. Установка свойств наборов данных

Каждый набор данных ZFS имеет свойства, которые определяют его поведение. Большинство свойств автоматически наследуются от родительского набора данных, но могут быть переопределены локально. Установите свойство для набора данных с помощью `zfs set свойство=значение набор_данных`. Большинство свойств имеют ограниченный набор допустимых значений, `zfs get` отобразит каждое возможное свойство и допустимые значения. Использование `zfs inherit` возвращает большинство свойств к их унаследованным значениям. Также возможны пользовательские свойства. Они становятся частью конфигурации набора данных и предоставляют дополнительную информацию о наборе данных или его содержимом. Чтобы отличить эти пользовательские свойства от встроенных в ZFS, используйте двоеточие (`:`), чтобы создать пользовательское пространство имён для свойства.

```
# zfs set custom:costcenter=1234 tank
# zfs get custom:costcenter tank
NAME PROPERTY           VALUE SOURCE
tank custom:costcenter  1234  local
```

Чтобы удалить пользовательское свойство, используйте `zfs inherit` с параметром `-r`. Если пользовательское свойство не определено ни в одном из родительских наборов данных, эта опция удаляет его (но история пула всё равно сохраняет запись об изменении).

```
# zfs inherit -r custom:costcenter tank
# zfs get custom:costcenter tank
NAME    PROPERTY           VALUE              SOURCE
tank    custom:costcenter  -                  -
# zfs get all tank | grep custom:costcenter
#
```

Получение и установка свойств общего доступа

Два часто используемых и полезных свойства наборов данных — это параметры общего доступа NFS и SMB. Их установка определяет, экспортируются ли наборы данных ZFS в сеть и каким образом. Во FreeBSD ZFS интегрируется с NFS-сервером в базовой системе: установка свойства `sharenfs` экспортирует набор данных через mountd(8). ZFS записывает запись экспорта в /etc/zfs/exports, которую mountd(8) читает в дополнение к /etc/exports. Для общего доступа к наборам данных требуется включенный NFS-сервер в /etc/rc.conf:

```
zfs_enable="YES"
nfs_server_enable="YES"
```

Значение `sharenfs` может быть `on`, `off` или списком параметров exports(5), применяемых к экспорту.

Чтобы получить текущее состояние общего каталога, введите:

```
# zfs get sharenfs mypool/usr/home
NAME             PROPERTY  VALUE    SOURCE
mypool/usr/home  sharenfs  on       local
```

Чтобы включить общий доступ к набору данных, введите:

```
#  zfs set sharenfs=on mypool/usr/home
```

Установите другие параметры для общего доступа к наборам данных через NFS, такие как `-alldirs`, `-maproot` и `-network`. Чтобы задать параметры для набора данных, доступного через NFS, введите:

```
#  zfs set sharenfs="-alldirs,-maproot=root,-network=192.168.1.0/24" mypool/usr/home
```

Свойство `sharesmb` не работает во FreeBSD, так как библиотека совместного доступа реализует только NFS. Чтобы предоставить набор данных по SMB, установите пакет, например net/samba422, и укажите путь смонтированного набора данных в его конфигурации.

23.4.5. NFSv4 ACL

ZFS во FreeBSD хранит ACL в стиле NFSv4 нативно; свойство набора данных `acltype` по умолчанию имеет значение `nfsv4`. Каждый файл и каталог может нести список управления доступом наряду с традиционными битами разрешений, обеспечивая более тонкий контроль, чем модель владелец/группа/остальные. Отображайте и редактируйте эти списки с помощью getfacl(1) и setfacl(1).

Отобразить ACL файла:

```
% getfacl /usr/home/alice/notes.txt
# file: /usr/home/alice/notes.txt
# owner: alice
# group: alice
           owner@:rw-p--aARWcCos:-------:allow
           group@:r-----a-R-c--s:-------:allow
        everyone@:r-----a-R-c--s:-------:allow
```

Предоставьте дополнительному пользователю доступ на запись, добавив запись `allow`:

```
% setfacl -m u:bob:rwp::allow /usr/home/alice/notes.txt
```

Два свойства наборов данных управляют тем, как ACL взаимодействуют с традиционной обработкой разрешений. Свойство `aclmode` (`discard`, `groupmask`, `passthrough` или `restricted`) определяет, что происходит с ACL, когда chmod(1) изменяет биты разрешений файла. При

значении по умолчанию, `discard`, chmod удаляет все записи ACL, которые не представляют новый режим файла. Свойство `aclinherit` (`discard`, `noallow`, `restricted`, `passthrough` или `passthrough-x`) определяет, какие записи ACL новые файлы и каталоги наследуют от родительского каталога. При значении по умолчанию, `restricted`, унаследованные записи теряют разрешения `write_acl` и `write_owner`.

Смотрите setfacl(1) и zfsprops(7) для полного синтаксиса записей и описаний свойств.

23.4.6. Управление снимками

Снимки — одна из самых мощных функций ZFS. Снимок предоставляет доступную только для чтения копию набора данных на определённый момент времени. Благодаря механизму Copy-On-Write (COW), ZFS быстро создаёт снимки, сохраняя старые версии данных на диске. Если снимков не существует, ZFS освобождает место для последующего использования при перезаписи или удалении данных. Снимки экономят дисковое пространство, записывая только различия между текущим набором данных и предыдущей версией. Создание снимков возможно для целых наборов данных, но не для отдельных файлов или каталогов. Снимок набора данных дублирует всё его содержимое. Это включает свойства файловой системы, файлы, каталоги, права доступа и так далее. Снимки не занимают дополнительного места при создании, но начинают потреблять пространство по мере изменения блоков, на которые они ссылаются. Рекурсивные снимки, созданные с помощью `-r`, формируют снимки с одинаковыми именами для набора данных и его дочерних элементов, обеспечивая согласованный снимок файловых систем на определённый момент времени. Это может быть важно, когда приложение использует файлы в связанных наборах данных или зависящие друг от друга. Без снимков резервная копия содержала бы файлы из разных моментов времени.

Снимки в ZFS предоставляют множество функций, которых нет даже в других файловых системах с поддержкой снимков. Типичный пример использования снимков — быстрое резервное копирование текущего состояния файловой системы перед выполнением рискованных действий, таких как установка программного обеспечения или обновление системы. Если действие завершится неудачей, откат к снимку вернёт систему в состояние на момент его создания. Если обновление прошло успешно, снимок можно удалить для освобождения места. Без снимков неудачное обновление часто требует восстановления из резервных копий, что утомительно, отнимает много времени и может привести к простою системы, в течение которого она будет недоступна. Откат к снимкам выполняется быстро, даже во время обычной работы системы, с минимальным или нулевым временем простоя. Экономия времени огромна, особенно для многотерабайтных систем хранения, учитывая время, необходимое для копирования данных из резервной копии. Снимки не заменяют полное резервное копирование пула, но предлагают быстрый и простой способ сохранить копию набора данных на определённый момент времени.

Создание снимков

Для создания снимков используйте `zfs snapshot набордданных@имяснимка`. Добавление опции `-r` создаёт снимок рекурсивно с тем же именем для всех дочерних наборов данных.

Создать рекурсивный снимок всего пула:

```
# zfs list -t all
NAME                                  USED  AVAIL  REFER  MOUNTPOINT
mypool                                780M  93.2G   144K  none
mypool/ROOT                           777M  93.2G   144K  none
mypool/ROOT/default                   777M  93.2G   777M  /
mypool/tmp                            176K  93.2G   176K  /tmp
mypool/usr                            616K  93.2G   144K  /usr
mypool/usr/home                       184K  93.2G   184K  /usr/home
mypool/usr/ports                      144K  93.2G   144K  /usr/ports
mypool/usr/src                        144K  93.2G   144K  /usr/src
mypool/var                           1.29M  93.2G   616K  /var
mypool/var/crash                      148K  93.2G   148K  /var/crash
mypool/var/log                        178K  93.2G   178K  /var/log
mypool/var/mail                       144K  93.2G   144K  /var/mail
mypool/var/newname                   87.5K  93.2G  87.5K  /var/newname
mypool/var/newname@new_snapshot_name     0      -  87.5K  -
mypool/var/tmp                        152K  93.2G   152K  /var/tmp
# zfs snapshot -r mypool@my_recursive_snapshot
# zfs list -t snapshot
NAME                                      USED  AVAIL  REFER  MOUNTPOINT
mypool@my_recursive_snapshot                 0      -   144K  -
mypool/ROOT@my_recursive_snapshot            0      -   144K  -
mypool/ROOT/default@my_recursive_snapshot     0      -   777M  -
mypool/tmp@my_recursive_snapshot             0      -   176K  -
mypool/usr@my_recursive_snapshot             0      -   144K  -
mypool/usr/home@my_recursive_snapshot        0      -   184K  -
mypool/usr/ports@my_recursive_snapshot       0      -   144K  -
mypool/usr/src@my_recursive_snapshot         0      -   144K  -
mypool/var@my_recursive_snapshot             0      -   616K  -
mypool/var/crash@my_recursive_snapshot       0      -   148K  -
mypool/var/log@my_recursive_snapshot         0      -   178K  -
mypool/var/mail@my_recursive_snapshot        0      -   144K  -
mypool/var/newname@new_snapshot_name         0      -  87.5K  -
mypool/var/newname@my_recursive_snapshot     0      -  87.5K  -
mypool/var/tmp@my_recursive_snapshot         0      -   152K  -
```

Снимки не отображаются при обычной операции `zfs list`. Для вывода списка снимков добавьте `-t snapshot` к команде `zfs list`. Опция `-t all` показывает как файловые системы, так и снимки.

Снимки не монтируются напрямую, поэтому в столбце `MOUNTPOINT` не отображается путь. ZFS не указывает доступное дисковое пространство в столбце `AVAIL`, так как снимки доступны только для чтения после их создания. Сравните снимок с исходным набором данных:

```
# zfs list -rt all mypool/usr/home
NAME                                 USED  AVAIL  REFER  MOUNTPOINT
mypool/usr/home                      184K  93.2G   184K  /usr/home
mypool/usr/home@my_recursive_snapshot     0      -   184K  -
```

Отображение набора данных и снимка вместе показывает, как снимки работают в стиле COW. Они сохраняют внесённые изменения (*дельту*), а не полное содержимое файловой системы заново. Это означает, что снимки занимают мало места при внесении изменений. Наблюдайте за использованием пространства ещё внимательнее, скопировав файл в набор данных, а затем создав второй снимок:

```
# cp /etc/passwd /var/tmp
# zfs snapshot mypool/var/tmp@after_cp
# zfs list -rt all mypool/var/tmp
NAME                                  USED  AVAIL  REFER  MOUNTPOINT
mypool/var/tmp                        206K  93.2G   118K  /var/tmp
mypool/var/tmp@my_recursive_snapshot   88K      -   152K  -
mypool/var/tmp@after_cp                 0      -   118K  -
```

Второй снимок содержит изменения в наборе данных после операции копирования. Это обеспечивает значительную экономию пространства. Обратите внимание, что размер снимка `mypool/var/tmp@my_recursive_snapshot` также изменился в столбце `USED`, показывая разницу между ним и последующим снимком.

Сравнение снимков

ZFS предоставляет встроенную команду для сравнения различий в содержимом между двумя снимками. Это полезно при наличии множества снимков, сделанных за определённый период, когда пользователь хочет увидеть, как файловая система изменялась со временем. Например, `zfs diff` позволяет пользователю найти последний снимок, который ещё содержит случайно удалённый файл. Применение этой команды к двум снимкам, созданным в предыдущем разделе, даёт следующий вывод:

```
# zfs list -rt all mypool/var/tmp
NAME                                  USED  AVAIL  REFER  MOUNTPOINT
mypool/var/tmp                        206K  93.2G   118K  /var/tmp
mypool/var/tmp@my_recursive_snapshot   88K      -   152K  -
mypool/var/tmp@after_cp                 0      -   118K  -
# zfs diff mypool/var/tmp@my_recursive_snapshot
M       /var/tmp/
+       /var/tmp/passwd
```

Команда выводит список изменений между указанным снимком (в данном случае `mypool/var/tmp@my_recursive_snapshot`) и активной файловой системой. Первый столбец показывает тип изменения:

+	Добавление пути или файла.
-	Удаление пути или файла.
M	Изменение пути или файла.
R	Переименование пути или файла.

Сравнивая вывод с таблицей, становится ясно, что ZFS добавил passwd после создания снимка `mypool/var/tmp@my_recursive_snapshot`. Это также привело к изменению родительского каталога, смонтированного в `/var/tmp`.

Сравнение двух снимков полезно при использовании функции репликации ZFS для передачи набора данных на другой хост в целях резервного копирования.

Сравните два снимка, указав полное имя набора данных и имя снимка для обоих наборов:

```
# cp /var/tmp/passwd /var/tmp/passwd.copy
# zfs snapshot mypool/var/tmp@diff_snapshot
# zfs diff mypool/var/tmp@my_recursive_snapshot mypool/var/tmp@diff_snapshot
M       /var/tmp/
+       /var/tmp/passwd
+       /var/tmp/passwd.copy
# zfs diff mypool/var/tmp@my_recursive_snapshot mypool/var/tmp@after_cp
M       /var/tmp/
+       /var/tmp/passwd
```

Администратор резервного копирования может сравнить два снимка, полученных от отправляющего хоста, и определить фактические изменения в наборе данных. Дополнительную информацию см. в разделе Репликация.

Откат снимка состояния (Snapshot Rollback)

Когда доступен хотя бы один снимок, можно в любой момент к нему вернуться. Чаще всего это необходимо, когда текущее состояние набора данных больше не является корректным или требуется более старая версия. Такие ситуации, как неудачные локальные тесты разработки, неудачные обновления системы, нарушающие её функциональность, или необходимость восстановления удалённых файлов или каталогов, встречаются довольно часто. Для возврата к снимку используйте команду `zfs rollback имя_снимка`. Если изменений много, операция займёт значительное время. В течение этого времени набор данных всегда остаётся в согласованном состоянии, подобно тому, как база данных, соответствующая принципам ACID, выполняет откат. Это происходит, пока набор данных находится в рабочем состоянии и доступен без необходимости простоя. После отката к снимку набор данных возвращается в то же состояние, в котором он находился на момент создания снимка. Откат к снимку удаляет все другие данные в наборе, не входящие в снимок. Создание снимка текущего состояния набора данных перед откатом к предыдущему — хорошая практика, если некоторые данные могут потребоваться позже. Таким образом, пользователь может переключаться между снимками, не теряя ценные данные.

В первом примере выполняется откат к снимку, потому что неосторожная операция `rm` удалила больше данных, чем планировалось.

```
# zfs list -rt all mypool/var/tmp
NAME                                 USED  AVAIL  REFER  MOUNTPOINT
mypool/var/tmp                       262K  93.2G   120K  /var/tmp
mypool/var/tmp@my_recursive_snapshot    88K      -   152K  -
```

```
mypool/var/tmp@after_cp               53.5K      -   118K  -
mypool/var/tmp@diff_snapshot             0       -   120K  -
# ls /var/tmp
passwd          passwd.copy     vi.recover
# rm /var/tmp/passwd*
# ls /var/tmp
vi.recover
```

На этом этапе пользователь замечает удаление лишних файлов и хочет вернуть их обратно. ZFS предоставляет простой способ восстановления с использованием отката, если регулярно создаются снимки важных данных. Чтобы вернуть файлы и начать заново с последнего снимка, выполните команду:

```
# zfs rollback mypool/var/tmp@diff_snapshot
# ls /var/tmp
passwd          passwd.copy     vi.recover
```

Операция отката восстановила набор данных до состояния последнего снимка. Также возможен откат к более раннему снимку, если после него были созданы другие снимки. При попытке выполнить это действие ZFS выдаст следующее предупреждение:

```
# zfs list -rt snapshot mypool/var/tmp
NAME                                   USED  AVAIL  REFER  MOUNTPOINT
mypool/var/tmp@my_recursive_snapshot    88K      -   152K  -
mypool/var/tmp@after_cp               53.5K      -   118K  -
mypool/var/tmp@diff_snapshot             0       -   120K  -
# zfs rollback mypool/var/tmp@my_recursive_snapshot
cannot rollback to 'mypool/var/tmp@my_recursive_snapshot': more recent snapshots exist
use '-r' to force deletion of the following snapshots:
mypool/var/tmp@after_cp
mypool/var/tmp@diff_snapshot
```

Это предупреждение означает, что между текущим состоянием набора данных и снимком, к которому пользователь хочет выполнить откат, существуют другие снимки. Для завершения отката удалите эти снимки. ZFS не может отслеживать все изменения между различными состояниями набора данных, поскольку снимки доступны только для чтения. ZFS не удалит затронутые снимки, если пользователь явно не укажет параметр `-r`, подтверждая, что это желаемое действие. Если это действительно требуется, и вы осознаёте последствия потери всех промежуточных снимков, выполните команду:

```
# zfs rollback -r mypool/var/tmp@my_recursive_snapshot
# zfs list -rt snapshot mypool/var/tmp
NAME                                   USED  AVAIL  REFER  MOUNTPOINT
mypool/var/tmp@my_recursive_snapshot     8K      -   152K  -
# ls /var/tmp
vi.recover
```

Вывод команды `zfs list -t snapshot` подтверждает удаление промежуточных снимков в результате выполнения `zfs rollback -r`.

Восстановление отдельных файлов из снимков

Снимки хранятся в скрытом каталоге родительского набора данных: .zfs/snapshot/имя_снимка. По умолчанию эти каталоги не отображаются даже при выполнении стандартной команды `ls -a`. Несмотря на то, что каталог не виден, доступ к нему осуществляется как к обычному каталогу. Свойство `snapdir` определяет, будут ли эти скрытые каталоги отображаться в списке содержимого каталога. Установка свойства в значение `visible` позволяет им появляться в выводе команды `ls` и других команд, работающих с содержимым каталогов.

```
# zfs get snapdir mypool/var/tmp
NAME            PROPERTY  VALUE    SOURCE
mypool/var/tmp  snapdir   hidden   default
# ls -a /var/tmp
.               ..              passwd          vi.recover
# zfs set snapdir=visible mypool/var/tmp
# ls -a /var/tmp
.               ..              .zfs            passwd          vi.recover
```

Восстановите отдельные файлы в предыдущее состояние, скопировав их из снимка обратно в родительский набор данных. Структура каталогов в .zfs/snapshot содержит каталоги с именами, соответствующими ранее созданным снимкам, что упрощает их идентификацию. В следующем примере показано, как восстановить файл из скрытого каталога .zfs, скопировав его из снимка, содержащего последнюю версию файла:

```
# rm /var/tmp/passwd
# ls -a /var/tmp
.               ..              .zfs            vi.recover
# ls /var/tmp/.zfs/snapshot
after_cp                my_recursive_snapshot
# ls /var/tmp/.zfs/snapshot/after_cp
passwd          vi.recover
# cp /var/tmp/.zfs/snapshot/after_cp/passwd /var/tmp
```

Даже если свойство `snapdir` установлено в hidden, выполнение команды `ls .zfs/snapshot` всё равно покажет содержимое этого каталога. Администратор решает, отображать ли эти каталоги. Это настройка для каждого набора данных. Копирование файлов или каталогов из скрытого .zfs/snapshot достаточно просто. Попытка сделать наоборот приведёт к такой ошибке:

```
# cp /etc/rc.conf /var/tmp/.zfs/snapshot/after_cp/
cp: /var/tmp/.zfs/snapshot/after_cp/rc.conf: Read-only file system
```

Ошибка напоминает пользователю, что снимки доступны только для чтения и не могут изменяться после создания. Копирование файлов в каталоги снимков и их удаление оттуда запрещены, так как это изменило бы состояние набора данных, который они представляют.

Снимки занимают место в зависимости от того, насколько родительская файловая система изменилась с момента создания снимка. Свойство `used` снимка отслеживает используемое им пространство. Его свойство `written` показывает, какой объём данных был записан на диск между предыдущим и текущим снимком.

Для удаления снимков и освобождения пространства используйте `zfs destroy набор_данных@снимок`. Добавление `-r` рекурсивно удаляет все снимки с таким же именем в родительском наборе данных. Добавление `-n -v` к команде выводит список снимков, которые будут удалены, и оценку освобождаемого пространства без фактического выполнения операции удаления.

Удержания снимков

Удержание (hold, hold) помечает снимок пользовательским тегом и предотвращает его уничтожение. Попытки уничтожить удерживаемый снимок с помощью `zfs destroy` завершаются ошибкой `EBUSY`. Снимок может иметь любое количество удержаний, каждое из которых идентифицируется уникальным именем тега. Удержания защищают снимки, которые всё ещё используются другими процессами, например, самый последний снимок, общий для обеих сторон цепочки репликации (см. Репликация). Случайное уничтожение этого общего снимка приводит к тому, что следующая репликация должна будет начинаться с полного потока (full stream).

Установите удержание с помощью `zfs hold tag snapshot`. Добавление `-r` также устанавливает удержания на снимки с тем же именем во всех дочерних наборах данных.

```
# zfs hold keepme mypool/var/tmp@my_recursive_snapshot
# zfs destroy mypool/var/tmp@my_recursive_snapshot
cannot destroy snapshot mypool/var/tmp@my_recursive_snapshot: dataset is busy
```

`zfs holds` выводит список удержаний на снимке, показывая имя тега и время установки каждого удержания:

```
# zfs holds mypool/var/tmp@my_recursive_snapshot
NAME                                   TAG     TIMESTAMP
mypool/var/tmp@my_recursive_snapshot   keepme  Sat Jul 11 09:41 2026
```

Снимите удержание с помощью `zfs release`. Уничтожение снимка снова становится возможным, как только снимается последнее удержание на нём:

```
# zfs release keepme mypool/var/tmp@my_recursive_snapshot
# zfs destroy mypool/var/tmp@my_recursive_snapshot
```

См. zfs-hold(8) для получения дополнительной информации.

23.4.7. Закладки

Перекрестная ссылка: zfs[zfs-term-bookmark,закладка] фиксирует момент времени, в который был создан снимок, без сохранения каких-либо данных снимка. Закладки почти не занимают места на диске, независимо от того, насколько изменится набор данных впоследствии. Как и снимок, закладка может служить источником инкрементальных данных для `zfs send`. В отличие от снимка, она не препятствует ZFS освобождать старые данные. Такое сочетание делает закладки полезными для инкрементальной репликации: после отправки снимка на другой пул создайте закладку на нем и уничтожьте его на отправляющей стороне, освобождая место, в то время как принимающая сторона сохраняет свою копию снимка. Смотрите раздел Инкрементальные резервные копии для описания самого процесса репликации.

Создайте закладку из снимка с помощью `zfs bookmark`. В именах закладок в качестве разделителя между именем набора данных и именем закладки используется `#`, так же как в именах снимков используется `@`.

```
# zfs snapshot mypool/data@snap1
# zfs send mypool/data@snap1 | zfs receive backup/data
# zfs bookmark mypool/data@snap1 mypool/data#snap1bm
# zfs destroy mypool/data@snap1
```

Перечислите закладки с помощью `zfs list -t bookmark`:

```
# zfs list -t bookmark mypool/data
NAME                 USED  AVAIL  REFER  MOUNTPOINT
mypool/data#snap1bm     -      -   184K  -
```

Когда наступает время для следующего инкрементального резервного копирования, создайте новый снимок и используйте закладку в качестве инкрементального источника:

```
# zfs snapshot mypool/data@snap2
# zfs send -i mypool/data#snap1bm mypool/data@snap2 | zfs receive backup/data
```

Это работает, поскольку принимающий пул всё ещё имеет снапшот *backup/data@snap1*, соответствующий закладке. Закладки служат только в качестве источников для инкрементального резервирования. Смонтировать закладку, выполнить откат к ней или восстановить файлы из неё невозможно, так как она не содержит данных. Дополнительную информацию см. в zfs-bookmark(8).

23.4.8. Управление клонами

Клон — это копия снимка, которая рассматривается как обычный набор данных. В отличие от снимка, клон доступен для записи, может быть смонтирован и имеет свои собственные

свойства. После создания клона с помощью команды `zfs clone` уничтожение исходного снимка становится невозможным. Чтобы изменить отношение «родитель-потомок» между клоном и снимком, используйте команду `zfs promote`. Продвижение клона делает снимок потомком клона, а не исходного родительского набора данных. Это изменит способ учёта пространства в ZFS, но не повлияет на фактически используемый объём. Клон можно смонтировать в любом месте иерархии файловой системы ZFS, а не только ниже исходного расположения снимка.

Чтобы продемонстрировать функцию клонирования, используйте следующий набора данных в качестве примера:

```
# zfs list -rt all camino/home/joe
NAME                    USED  AVAIL  REFER  MOUNTPOINT
camino/home/joe         108K   1.3G    87K  /usr/home/joe
camino/home/joe@plans    21K      -  85.5K  -
camino/home/joe@backup    0K      -    87K  -
```

Типичное применение клонов — эксперименты с определённым набором данных, при этом снимок остаётся в качестве резервной копии на случай возникновения проблем. Поскольку снимки нельзя изменить, создаётся доступный для чтения и записи клон снимка. После достижения нужного результата в клоне, клон повышается до набора данных, а старая файловая система удаляется. Удаление родительского набора данных не является обязательным, так как клон и набор данных могут без проблем сосуществовать.

```
# zfs clone camino/home/joe@backup camino/home/joenew
# ls /usr/home/joe*
/usr/home/joe:
backup.txz     plans.txt

/usr/home/joenew:
backup.txz     plans.txt
# df -h /usr/home
Filesystem          Size    Used   Avail Capacity  Mounted on
usr/home/joe        1.3G     31k    1.3G     0%    /usr/home/joe
usr/home/joenew     1.3G     31k    1.3G     0%    /usr/home/joenew
```

Создание клона делает его точной копией состояния набора данных на момент создания снимка. Теперь можно изменять клон независимо от исходного набора данных. Связь между ними осуществляется через снимок. ZFS записывает эту связь в свойстве `origin`. Повышение клона с помощью `zfs promote` делает клон независимым набором данных. Это удаляет значение свойства `origin` и отключает новый независимый набор данных от снимка. Вот пример:

```
# zfs get origin camino/home/joenew
NAME                 PROPERTY  VALUE                    SOURCE
camino/home/joenew   origin    camino/home/joe@backup   -
# zfs promote camino/home/joenew
```

```
# zfs get origin camino/home/joenew
NAME                 PROPERTY  VALUE   SOURCE
camino/home/joenew   origin    -       -
```

После внесения изменений, таких как копирование loader.conf в продвинутую клон-копию, например, старый каталог в этом случае становится устаревшим. Вместо него продвинутая клон-копия может его заменить. Для этого сначала выполните `zfs destroy` для старого набора данных, а затем `zfs rename` для клона, указав имя старого набора данных (или совершенно другое имя).

```
# cp /boot/defaults/loader.conf /usr/home/joenew
# zfs destroy -f camino/home/joe
# zfs rename camino/home/joenew camino/home/joe
# ls /usr/home/joe
backup.txz     loader.conf     plans.txt
# df -h /usr/home
Filesystem          Size    Used   Avail Capacity  Mounted on
usr/home/joe        1.3G    128k    1.3G     0%    /usr/home/joe
```

Клонированный снимок теперь является обычным набором данных. Он содержит все данные из исходного снимка, а также добавленные файлы, такие как loader.conf. Клоны предоставляют полезные возможности пользователям ZFS в различных сценариях. Например, можно предоставлять клетки в виде снимков с различными наборами установленных приложений. Пользователи могут клонировать эти снимки и добавлять свои собственные приложения по своему усмотрению. После внесения необходимых изменений клоны можно повысить до полноценных наборов данных и предоставить их конечным пользователям для работы, как с обычными наборами данных. Это экономит время и снижает административные затраты при предоставлении таких клеток.

23.4.9. Блочное клонирование

Блочное клонирование обеспечивает копирование при записи для отдельных копий файлов. При его включении копирование файла с помощью cp(1) не дублирует данные файла. Вместо этого новый файл ссылается на те же блоки, что и оригинал, и ZFS записывает новые блоки только тогда, когда впоследствии изменяется один из файлов. cp(1) выполняет копирование с помощью copy_file_range(2), которое ZFS превращает в операции блочного клонирования, делая копии внутри пула практически мгновенными и изначально не потребляющими дополнительное пространство.

В отличие от дедупликация, блочное клонирование не поддерживает требующую много памяти таблицу контрольных сумм. Совместное использование происходит только в момент копирования, и запись одинаковых данных отдельными операциями по-прежнему сохраняет их дважды. В отличие от клонов, блочное клонирование не требует снимка, работает с отдельными файлами, а не с целыми наборами данных, и не оставляет отношения `origin` после себя.

Блочное клонирование требует возможности пула `block_cloning` из OpenZFS 2.2, впервые выпущенной в FreeBSD 14.0 (см. zpool-features(7)). Управляющая системная переменная

sysctl(8) `vfs.zfs.bclone_enabled` определяет, создаёт ли ZFS блочные клоны на самом деле. По умолчанию она равна 0 в FreeBSD 14.x и 1 в FreeBSD 15.0. Пока sysctl равен 0, `copy_file_range()` возвращается к обычному копированию. Чтобы включить блочное клонирование в FreeBSD 14.x:

```
# sysctl vfs.zfs.bclone_enabled=1
vfs.zfs.bclone_enabled: 0 -> 1
```

Чтобы сохранить эту настройку после перезагрузок, добавьте следующую строку в /etc/sysctl.conf:

```
vfs.zfs.bclone_enabled=1
```

Свойства пула `bcloneused`, `bclonesaved` и `bcloneratio` показывают пространство, используемое клонированными блоками, пространство, сэкономленное за счёт клонирования, и результирующий коэффициент экономии:

```
# cp /var/tmp/database.dump /var/tmp/database.copy
# zpool list -o name,bcloneused,bclonesaved,bcloneratio mypool
NAME    BCLONE_USED  BCLONE_SAVED  BCLONE_RATIO
mypool         1.7G          1.7G         2.00x
```

23.4.10. Репликация

Хранение данных в единственном пуле в одном месте подвергает их рискам, таким как кража, стихийные бедствия или действия людей. Регулярное резервное копирование всего пула крайне важно. ZFS предоставляет встроенную функцию сериализации, которая может отправлять потоковое представление данных на стандартный вывод. Используя эту функцию, можно сохранять эти данные в другом пуле, подключённом к локальной системе, или отправлять их по сети на другую систему. Снимки являются основой для этой репликации (см. раздел о снимках ZFS). Команды, используемые для репликации данных, — это `zfs send` и `zfs receive`.

Эти примеры демонстрируют репликацию ZFS с использованием следующих двух пулов:

```
# zpool list
NAME    SIZE  ALLOC   FREE   CKPOINT  EXPANDSZ   FRAG   CAP  DEDUP  HEALTH  ALTROOT
backup  960M    77K   896M         -         -     0%    0%  1.00x  ONLINE  -
mypool  984M  43.7M   940M         -         -     0%    4%  1.00x  ONLINE  -
```

Имя пула *mypool* — это основной пул, в который данные регулярно записываются и откуда они читаются. Используйте второй резервный пул *backup* на случай, если основной пул станет недоступен. Обратите внимание, что этот переход на резервный пул не выполняется автоматически в ZFS, а должен быть осуществлён вручную системным администратором при необходимости. Используйте снимок (snapshot), чтобы обеспечить согласованную

версию файловой системы для репликации. После создания снимка *mypool* скопируйте его в пул *backup* путём репликации снимков. Это не включает изменения, сделанные после последнего снимка.

```
# zfs snapshot mypool@backup1
# zfs list -t snapshot
NAME                    USED  AVAIL  REFER  MOUNTPOINT
mypool@backup1             0      -  43.6M  -
```

Теперь, когда снимок существует, используйте `zfs send` для создания потока, представляющего содержимое снимка. Сохраните этот поток в файл или примите его в другом пуле. Поток записывается в стандартный вывод, но перенаправьте его в файл или канал, иначе появится ошибка:

```
# zfs send mypool@backup1
Error: Stream can not be written to a terminal.
You must redirect standard output.
```

Для резервного копирования набора данных с помощью `zfs send` перенаправьте вывод в файл, расположенный в подключённом пуле резервных копий. Убедитесь, что в пуле достаточно свободного места для размещения отправленного снимка, то есть для данных, содержащихся в снимке, а не для изменений по сравнению с предыдущим снимком.

```
# zfs send mypool@backup1 > /backup/backup1
# zpool list
NAME    SIZE  ALLOC   FREE   CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP  HEALTH  ALTROOT
backup  960M  63.7M   896M         -         -     0%     6%  1.00x  ONLINE  -
mypool  984M  43.7M   940M         -         -     0%     4%  1.00x  ONLINE  -
```

Команда `zfs send` передала все данные из снимка *backup1* в пул *backup*. Для автоматического создания и отправки таких снимков используйте задание cron(8).

Вместо хранения резервных копий в виде архивных файлов ZFS может получать их как активную файловую систему, обеспечивая прямой доступ к резервным данным. Для доступа к фактическим данным, содержащимся в этих потоках, используйте `zfs receive`, чтобы преобразовать потоки обратно в файлы и каталоги. В приведённом ниже примере объединяются `zfs send` и `zfs receive` с использованием конвейера для копирования данных из одного пула в другой. После завершения передачи данные можно использовать напрямую в целевом пуле. Реплицировать набор данных можно только в пустой набор данных.

```
# zfs snapshot mypool@replica1
# zfs send -v mypool@replica1 | zfs receive backup/mypool
full send of mypool@replica1 estimated size is 50.1M
total estimated size is 50.1M
TIME        SENT   SNAPSHOT mypool@replica1
```

```
10:22:01   9.75M   mypool@replica1
10:22:02   25.4M   mypool@replica1

# zpool list
NAME    SIZE  ALLOC   FREE   CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP  HEALTH  ALTROOT
backup  960M  63.7M   896M         -         -     0%     6%  1.00x  ONLINE  -
mypool  984M  43.7M   940M         -         -     0%     4%  1.00x  ONLINE  -
```

Инкрементные резервные копии

`zfs send` также может определить разницу между двумя снимками и отправить отдельные различия между ними. Это экономит место на диске и время передачи. Например:

```
# zfs snapshot mypool@replica2
# zfs list -t snapshot
NAME                   USED  AVAIL  REFER  MOUNTPOINT
mypool@replica1         5.72M      -  43.6M  -
mypool@replica2             0      -  44.1M  -
# zpool list
NAME    SIZE  ALLOC   FREE   CKPOINT  EXPANDSZ   FRAG   CAP  DEDUP  HEALTH  ALTROOT
backup  960M  61.7M   898M         -         -     0%    6%  1.00x  ONLINE  -
mypool  960M  50.2M   910M         -         -     0%    5%  1.00x  ONLINE  -
```

Создайте второй снимок с именем *replica2*. Этот второй снимок содержит изменения, внесённые в файловую систему в период между текущим моментом и предыдущим снимком *replica1*. Использование `zfs send -i` с указанием пары снимков создаёт инкрементальный поток репликации, содержащий изменённые данные. Это выполняется успешно, если исходный снимок уже существует на принимающей стороне.

```
# zfs send -v -i mypool@replica1 mypool@replica2 | zfs receive backup/mypool
send from @replica1 to mypool@replica2 estimated size is 5.02M
total estimated size is 5.02M
TIME        SENT   SNAPSHOT mypool@replica2

# zpool list
NAME    SIZE  ALLOC   FREE   CKPOINT  EXPANDSZ   FRAG  CAP  DEDUP  HEALTH  ALTROOT
backup  960M  80.8M   879M         -         -     0%   8%  1.00x  ONLINE  -
mypool  960M  50.2M   910M         -         -     0%   5%  1.00x  ONLINE  -

# zfs list
NAME                         USED  AVAIL  REFER  MOUNTPOINT
backup                      55.4M   240G   152K  /backup
backup/mypool               55.3M   240G  55.2M  /backup/mypool
mypool                      55.6M  11.6G  55.0M  /mypool

# zfs list -t snapshot
NAME                                      USED  AVAIL  REFER  MOUNTPOINT
backup/mypool@replica1                    104K      -  50.2M  -
```

```
backup/mypool@replica2                    0      -  55.2M  -
mypool@replica1                        29.9K     -  50.0M  -
mypool@replica2                           0      -  55.0M  -
```

Инкрементный поток реплицировал изменённые данные вместо полной копии *replica1*. Передача только различий заняла гораздо меньше времени и сэкономила место на диске, избегая копирования всего пула каждый раз. Это особенно полезно при репликации по медленной сети или при тарификации за каждый переданный байт.

Доступна новая файловая система *backup/mypool* с файлами и данными из пула *mypool*. Указание `-p` копирует свойства наборов данных, включая настройки сжатия, квоты и точки монтирования. Указание `-R` копирует все дочерние наборы данных вместе с их свойствами. Автоматизируйте отправку и получение для создания регулярных резервных копий во втором пуле. Раздел «Параметры потока отправки» более подробно описывает эти и другие полезные параметры.

Параметры отправки потока

`zfs send` предоставляет параметры, которые управляют тем, что включает поток и насколько он компактен. Указание `-p` включает свойства набора данных, такие как настройки сжатия, квоты и точки монтирования, в поток. Указание `-R` генерирует пакет репликационного потока, который включает набор данных, все дочерние наборы данных, снимки, клоны и свойства вплоть до именованного снимка, и подразумевает `-p`. Пример в Отправка зашифрованных бекапов через SSH использует `-R` для репликации полного дерева набора данных. Указание `-v` выводит информацию о сгенерированном потоке, включая отчет о ходе выполнения за секунду.

Другие параметры изменяют формат блоков данных в потоке:

- `-L` разрешает блоки размером более 128 КБ в потоке, сохраняя размер блоков на диске для наборов данных с `recordsize` выше 128 КБ вместо разделения таких блоков.
- `-e` создаёт более компактный поток, используя записи `WRITE_EMBEDDED` для крошечных блоков, которые функция пула `embedded_data` хранит непосредственно в указателе блока.
- `-c` отправляет блоки, сжатые на диске, в их сжатом виде, а не распаковывает их сначала, экономя время процессора на обеих системах и сокращая объем передаваемых данных для сжатых наборов данных.
- `-w` отправляет зашифрованный набор данных в необработанном виде, точно так, как он хранится на диске, и без загрузки его ключа шифрования, как описано в разделе Отправка зашифрованных наборов данных. Для незашифрованных наборов данных `-w` эквивалентно `-Lec`.

Объединение этих параметров как `zfs send -Lec` является хорошим выбором по умолчанию для современных пулов. Принимающий пул должен поддерживать все возможности, которые использует поток, и `zfs receive` отклоняет поток, требующий возможности, отсутствующей в пуле. Смотрите zfs-send(8) и zpool-features(7) для полного списка параметров и возможностей пула, от которых они зависят.

Возобновляемые передачи

Разорванное сетевое соединение или перезагрузка во время большой передачи обычно означает, что нужно начинать заново. Использование `zfs receive -s` сохраняет частично принятое состояние на принимающем наборе данных вместо его удаления, когда поток завершается преждевременно.

```
# zfs send -v mypool@replica1 | zfs receive -s backup/mypool
full send of mypool@replica1 estimated size is 50.1M
total estimated size is 50.1M
TIME        SENT   SNAPSHOT mypool@replica1
10:31:01   9.75M   mypool@replica1
^C
```

После прерывания принимающий набор данных сохраняет непрозрачный токен возобновления в своём свойстве `receive_resume_token`. Токен записывает, какой снимок содержал прерванный поток и какая его часть была получена. Передача токена команде `zfs send -t` создаёт новый поток, который продолжает с того места, где передача остановилась:

```
# zfs get -H -o value receive_resume_token backup/mypool
1-1211c4f4a-f8-
789c636064000310a501c49c50360710a715e5e7a69766a63040c1eabb735735ce8f8d5420c0e5c9e8d4d3
d28a5388e4d3d200
# zfs send -t $(zfs get -H -o value receive_resume_token backup/mypool) | zfs receive
-s backup/mypool
```

Возобновлённая передача завершается нормально и удаляет сохранённое частичное состояние. Пока передача не возобновлена или не прервана, частичное состояние занимает место в принимающем пуле и блокирует другие приёмы в тот же набор данных. Чтобы отказаться от прерванной передачи вместо её возобновления, используйте `zfs receive -A` для удаления сохранённого частичного состояния:

```
# zfs receive -A backup/mypool
```

Отправка зашифрованных резервных копий через SSH

Отправка потоков данных по сети — это хороший способ создания удалённой резервной копии, но у этого метода есть недостаток. Данные, передаваемые по сетевому соединению, не шифруются, что позволяет любому перехватить их и преобразовать обратно в данные без ведома отправителя. Это нежелательно при отправке потоков через интернет на удалённый хост. Используйте SSH для безопасного шифрования данных, передаваемых по сетевому соединению. Поскольку ZFS требует перенаправления потока из стандартного вывода, его легко передать через SSH с помощью конвейера. Чтобы содержимое файловой системы оставалось зашифрованным и на удалённой системе, используйте встроенное шифрование и отправляйте набор данных в виде необработанного (raw) потока, как описано в «Отправка зашифрованных наборов данных».

Измените некоторые настройки и сначала примите меры безопасности. Здесь описаны необходимые шаги для операции `zfs send`; дополнительную информацию о SSH см. в OpenSSH.

Измените конфигурацию следующим образом:

- Беспарольный доступ SSH между отправляющим и принимающим хостами с использованием SSH-ключей
- Для отправки и получения потоков ZFS требуются привилегии пользователя `root`. Это подразумевает вход в принимающую систему под учётной записью `root`.
- По соображениям безопасности вход пользователя `root` по умолчанию запрещён.
- Используйте систему Делегирование ZFS, чтобы разрешить пользователю без прав `root` на каждой системе выполнять соответствующие операции отправки и получения. На передающей системе:

```
# zfs allow -u someuser send,snapshot mypool
```

- Чтобы подключить пул, непривилегированный пользователь должен быть владельцем каталога, а обычные пользователи должны иметь разрешение на подключение файловых систем.

На принимающей системе:

```
# sysctl vfs.usermount=1
vfs.usermount: 0 -> 1
# echo vfs.usermount=1 >> /etc/sysctl.conf
# zfs create recvpool/backup
# zfs allow -u someuser create,mount,receive recvpool/backup
# chown someuser /recvpool/backup
```

Непривилегированный пользователь теперь может получать и монтировать наборы данных, а также реплицирует набор данных *home* на удалённую систему:

```
% zfs snapshot -r mypool/home@monday
% zfs send -R mypool/home@monday | ssh someuser@backuphost zfs recv -dvu
recvpool/backup
```

Создайте рекурсивный снимок с именем *monday* для набора данных файловой системы *home* в пуле *mypool*. Затем `zfs send -R` включает в поток набор данных, все дочерние наборы данных, снимки, клоны и настройки. Передайте вывод через SSH на ожидающий `zfs receive` на удалённом хосте *backuphost*. Рекомендуется использовать IP-адрес или полное доменное имя. Принимающая машина записывает данные в набор данных *backup* в пуле *recvpool*. Добавление опции `-d` к команде `zfs recv` отбрасывает первый элемент пути отправленного снимка (обычно это имя пула) и присоединяет оставшуюся часть пути к принимаемому набору данных, создавая при необходимости все промежуточные файловые системы. `-u`

отключает монтирование файловых систем на принимающей стороне. Использование `-v` показывает подробности о передаче, включая затраченное время и объём переданных данных.

Корректирующий приём

Процедура проверки (scrub) обнаруживает поврежденные данные, но в пуле без достаточной избыточности она не может исправить повреждения и вместо этого сообщает о постоянных ошибках. Если на другом пуле существует неповрежденная копия затронутого снимка, корректирующий прием с помощью `zfs receive -c` восстанавливает поврежденные блоки на месте, используя поток отправки как источник здоровых данных. Набор данных остается на своем месте, без каких-либо откатов или переименований.

Корректирующий приём требует OpenZFS 2.2, впервые поставленный во FreeBSD 14.0.

В этом примере проверка пула *mypool* на одном диске обнаружила неустранимые повреждения:

```
# zpool status -v mypool
  pool: mypool
 state: ONLINE
status: One or more devices has experienced an error resulting in data
        corruption.  Applications may be affected.
action: Restore the file in question if possible.  Otherwise restore the
        entire pool from backup.
   see: https://openzfs.github.io/openzfs-docs/msg/ZFS-8000-8A
  scan: scrub repaired 0B in 00:01:22 with 1 errors on Sat Jul 11 09:14:37 2026
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          ada0      ONLINE       0     0     2

errors: Permanent errors have been detected in the following files:

        /mypool/reports/summary-2026.db
```

Снимок *replica2* по-прежнему ссылается на повреждённые блоки, а пул *backup* содержит неповреждённую копию этого снимка. Отправьте снимок из пула backup и получите его корректирующе в соответствующий снимок на повреждённом пуле, затем выполните проверку (scrub), чтобы подтвердить восстановление:

```
# zfs send backup/mypool@replica2 | zfs receive -c mypool@replica2
# zpool scrub -w mypool
# zpool status mypool | grep errors:
errors: No known data errors
```

Поток должен содержать тот же снапшот, который существует на повреждённом наборе данных. Корректирующий приём восстанавливает только те блоки данных, которые присутствуют в потоке; он не может исправить метаданные или данные, записанные после создания снапшота. После этого всегда выполняйте scrub, чтобы убедиться, что пул больше не содержит повреждённых данных.

23.4.11. Квоты наборов данных, пользователей и групп

Используйте Квоты наборов данных, чтобы ограничить объём пространства, используемого определённым набором данных. Референтные квоты работают схожим образом, но учитывают пространство, используемое самим набором данных, исключая снимки и дочерние наборы данных. Аналогично, используйте пользовательские и групповые квоты, чтобы предотвратить исчерпание всего пространства в пуле или наборе данных пользователями или группами. Квоты проектов, описанные ниже, ограничивают занимаемое пространство для произвольного дерева каталогов.

Следующие примеры предполагают, что пользователи уже существуют в системе. Перед добавлением пользователя в систему убедитесь, что вы сначала создали его домашний набор данных и установили `mountpoint` в `/home/bob`. Затем создайте пользователя и укажите домашний каталог на расположение `mountpoint` набора данных. Это правильно установит права владельца и группы без перекрытия уже существующих путей домашних каталогов.

Чтобы установить квоту набора данных в 10 ГБ для storage/home/bob:

```
# zfs set quota=10G storage/home/bob
```

Чтобы установить контрольную квоту в 10 ГБ для storage/home/bob:

```
# zfs set refquota=10G storage/home/bob
```

Удалить квоту в 10 ГБ для storage/home/bob:

```
# zfs set quota=none storage/home/bob
```

Общий формат `userquota@пользователь=размер`, и имя пользователя должно быть в одном из следующих форматов:

- POSIX-совместимое имя, например *joe*.
- Числовой идентификатор POSIX, например, *789*.
- Имя SID, например, *joe.bloggs@example.com*.
- Числовой идентификатор SID, например, *S-1-123-456-789*.

Например, чтобы установить пользовательскую квоту в 50 ГБ для пользователя с именем *joe*:

```
# zfs set userquota@joe=50G storage/home/joe
```

Чтобы удалить любую квоту:

```
# zfs set userquota@joe=none storage/home/joe
```

Свойства квот пользователей не отображаются командой `zfs get all`. Пользователи без прав `root` не могут видеть квоты других, если им не предоставлена привилегия `userquota`. Пользователи с этой привилегией могут просматривать и устанавливать квоты для всех.

Общий формат для установки квоты группы: `groupquota@группа=размер`.

Чтобы установить квоту для группы *firstgroup* в 50 ГБ, используйте:

```
# zfs set groupquota@firstgroup=50G storage/home
```

Чтобы удалить квоту для группы *firstgroup* или убедиться, что она не установлена, используйте вместо этого:

```
# zfs set groupquota@firstgroup=none storage/home
```

Как и в случае с пользовательскими квотами, пользователи без прав `root` могут видеть квоты, связанные с группами, к которым они принадлежат. Пользователь с привилегией `groupquota` или `root` может просматривать и устанавливать квоты для всех групп.

Для отображения объёма пространства, используемого каждым пользователем в файловой системе или снимке, вместе с квотами, используйте `zfs userspace`. Для информации о группах используйте `zfs groupspace`. Подробнее о поддерживаемых опциях или о том, как отобразить только определённые опции, см. в zfs-userspace(8).

Привилегированные пользователи и `root` могут просмотреть квоту для storage/home/bob, используя:

```
# zfs get quota storage/home/bob
```

Проектные квоты

Квоты наборов данных и пользователей подходят не для любой структуры. Когда несколько деревьев каталогов, принадлежащих разным проектам, используют один набор данных, квоты проектов ограничивают пространство, потребляемое каждым деревом, вместо этого. Проект — это произвольный числовой идентификатор, присваиваемый файлам и каталогам.

`zfs project` управляет идентификатором проекта файлов и каталогов. Установка флага наследования проекта на каталог приводит к тому, что новые файлы и подкаталоги, созданные внутри него, наследуют его идентификатор проекта. Чтобы назначить идентификатор проекта *42* существующему дереву, рекурсивно установите флаг и идентификатор:

```
# zfs project -s -p 42 -r /storage/proj
```

Наложить ограничение в 100 ГБ на все файлы, принадлежащие проекту *42*, с помощью свойства `projectquota`:

```
# zfs set projectquota@42=100G storage
```

`zfs projectspace` отображает пространство, потребляемое каждым проектом на наборе данных, а также любые квоты:

```
# zfs projectspace -o name,used,quota storage
NAME   USED  QUOTA
42    1.95G   100G
```

Свойство `projectobjquota` ограничивает количество объектов, которыми может владеть проект, аналогичным образом. Отобразите или очистите идентификатор проекта для каталога с помощью `zfs project -d` и `zfs project -C`. Подробности см. в zfs-project(8) и zfsprops(7).

23.4.12. Резервирования

Резервирования гарантируют всегда доступный объём пространства в наборе данных. Зарезервированное пространство не будет доступно для других наборов данных. Эта полезная функция обеспечивает наличие свободного места для важных наборов данных или файлов журналов.

Общий формат свойства `reservation` — `reservation=размер`, поэтому, чтобы установить резервирование в 10 ГБ для storage/home/bob, используйте:

```
# zfs set reservation=10G storage/home/bob
```

Чтобы очистить любое резервирование:

```
# zfs set reservation=none storage/home/bob
```

Тот же принцип применяется к свойству `refreservation` для установки Референсного резервирования, с общим форматом `refreservation=размер`.

Эта команда показывает все и резервирования (`reservation`), и референсные резервирования (`refreservation`), существующие в storage/home/bob:

```
# zfs get reservation storage/home/bob
# zfs get refreservation storage/home/bob
```

23.4.13. Сжатие

ZFS предоставляет прозрачное сжатие. Сжатие данных на уровне блоков экономит место и увеличивает пропускную способность диска. Если данные сжимаются на 25%, то сжатые данные записываются на диск с той же скоростью, что и несжатые, что приводит к эффективной скорости записи в 125%. Сжатие также может быть отличной альтернативой Дедупликации, так как не требует дополнительной памяти.

ZFS предлагает различные алгоритмы сжатия, каждый со своими компромиссами. В OpenZFS 2.2 и новее для вновь создаваемых пулов и наборов данных по умолчанию установлено значение `compression=on`, которое выбирает алгоритм LZ4. Сжатие LZ4 позволяет сжимать весь пул без значительного снижения производительности, характерного для других алгоритмов. Главное преимущество LZ4 — функция *раннего прерывания*. Если LZ4 не достигает как минимум 12,5% сжатия в заголовочной части данных, ZFS записывает блок без сжатия, чтобы избежать потерь процессорного времени на попытки сжать уже сжатые или несжимаемые данные. Изменение свойства `compression` влияет только на новые записи; существующие данные сохраняют свою старую кодировку до тех пор, пока не будут перезаписаны (см. «Перезапись существующих данных»). Подробнее о различных алгоритмах сжатия, доступных в ZFS, см. в разделе Сжатие терминологии.

Администратор может оценить эффективность сжатия, используя свойства набора данных.

```
# zfs get used,compressratio,compression,logicalused mypool/compressed_dataset
NAME                        PROPERTY       VALUE  SOURCE
mypool/compressed_dataset  used           449G   -
mypool/compressed_dataset  compressratio  1.11x  -
mypool/compressed_dataset  compression    lz4    local
mypool/compressed_dataset  logicalused    496G   -
```

Набор данных использует 449 ГБ пространства (свойство `used`). Без сжатия он занял бы 496 ГБ пространства (свойство `logicalused`). Это даёт коэффициент сжатия 1.11:1.

Сжатие может иметь неожиданный побочный эффект при использовании вместе с Квотами пользователей. Квоты пользователей ограничивают фактическое пространство, которое пользователь занимает на наборе данных *после сжатия*. Если у пользователя есть квота в 10 ГБ, и он записывает 10 ГБ сжимаемых данных, он всё равно сможет сохранить больше данных. Если позже пользователь обновит файл, например базу данных, более или менее сжимаемыми данными, количество доступного ему пространства изменится. Это может привести к необычной ситуации, когда пользователь не увеличил фактический объём данных (свойство `logicalused`), но изменение степени сжатия привело к достижению предела его квоты.

Сжатие может иметь схожий непредвиденный эффект при взаимодействии с резервными копиями. Квоты часто используются для ограничения хранимых данных, чтобы гарантировать наличие достаточного места для резервного копирования. Поскольку квоты не учитывают сжатие, ZFS может записать больше данных, чем поместилось бы при резервном копировании без сжатия.

23.4.14. Сжатие алгоритмом Zstandard

В OpenZFS 2.0 был добавлен алгоритм сжатия Zstandard (Zstd). Zstd обеспечивает более высокие коэффициенты сжатия по сравнению с используемым по умолчанию LZ4, при этом работая значительно быстрее альтернативного gzip.

Zstd предоставляет широкий выбор уровней сжатия, обеспечивая детальный контроль над производительностью и степенью сжатия. Одним из основных преимуществ Zstd является то, что скорость распаковки не зависит от уровня сжатия. Для данных, которые записываются один раз, но часто читаются, Zstd позволяет использовать максимальные уровни сжатия без потери производительности при чтении.

Даже при частом обновлении данных включение сжатия часто обеспечивает более высокую производительность. Одно из главных преимуществ связано с функцией сжатого ARC. Адаптивный кэш замещения (ARC Adaptive Replacement Cache) в ZFS хранит сжатую версию данных в оперативной памяти, распаковывая их при каждом обращении. Это позволяет хранить больше данных и метаданных в том же объёме памяти, повышая коэффициент попадания в кэш.

ZFS предлагает 19 уровней сжатия Zstd, каждый из которых обеспечивает постепенное увеличение экономии места в обмен на более медленное сжатие. Уровень по умолчанию — `zstd-3`, который обеспечивает лучшее сжатие, чем LZ4, без значительного снижения скорости. Уровни выше 10 требуют большого объёма памяти для сжатия каждого блока, и системы с менее чем 16 ГБ ОЗУ не должны их использовать. ZFS также использует подмножество уровней Zstd *fast*, которые работают быстрее, но обеспечивают меньшую степень сжатия. ZFS поддерживает уровни от `zstd-fast-1` до `zstd-fast-10`, от `zstd-fast-20` до `zstd-fast-100` с шагом 10, а также `zstd-fast-500` и `zstd-fast-1000`, которые обеспечивают минимальное сжатие, но обладают высокой производительностью.

Если ZFS не может получить необходимую память для сжатия блока с помощью Zstd, он переходит к сохранению блока в несжатом виде. Это маловероятно, за исключением случаев использования максимальных уровней Zstd на системах с ограниченной памятью. ZFS подсчитывает, сколько раз это произошло с момента загрузки модуля ZFS, с помощью `kstat.zfs.misc.zstd.compress_alloc_fail`.

23.4.15. Перезапись существующих данных

`zfs rewrite` требует OpenZFS 2.4, впервые выпущенный во FreeBSD 15.0.

Свойства, такие как `compression`, `checksum`, `dedup` и `copies`, влияют только на вновь записываемые данные. Их изменение оставляет существующие блоки сохранёнными как есть, и применение новых значений традиционно требовало цикла отправки/получения или копирования каждого файла туда и обратно. `zfs rewrite` вместо этого перезаписывает

блоки существующих файлов на месте, как если бы они были атомарно прочитаны и записаны обратно, поэтому текущие значения свойств вступают в силу без этих обходных путей.

Команда работает с файлами и каталогами, а не с именами наборов данных. Чтобы применить вновь выбранный алгоритм сжатия ко всему набору данных, выполните рекурсивный обход от его точки монтирования:

```
# zfs set compression=zstd mypool/archive
# zfs rewrite -r /mypool/archive
```

Добавление `-v` выводит имя каждого перезаписанного файла. `-x` предотвращает переход рекурсии через точки монтирования в дочерние наборы данных, а `-o` и `-l` ограничивают перезапись диапазоном байт внутри файла. Изменения свойств, которые могли бы изменить логический размер блока, такие как `recordsize`, не влияют на перезаписанные файлы.

Переписанные блоки — это новые блоки: снимки, сделанные до перезаписи, продолжают ссылаться на старые копии, поэтому использование дискового пространства может расти, пока эти снимки не будут уничтожены. `zfs rewrite` не работает через снимки по той же причине. Подробности см. в zfs-rewrite(8).

23.4.16. Дедупликация

Когда включена перекрёстная дедупликация, она использует контрольную сумму каждого блока для обнаружения дублирующихся блоков. Когда новый блок является дубликатом существующего блока, ZFS записывает новую ссылку на существующие данные вместо всего дублирующегося блока. Возможна значительная экономия места, если данные содержат много дублирующихся файлов или повторяющейся информации. Предупреждение: дедупликация требует большого объёма памяти, а включение сжатия обеспечивает большую часть экономии места без дополнительных затрат.

`dedup` является свойством набора данных, а не свойством пула. Чтобы активировать дедупликацию, установите это свойство на наборе данных:

```
# zfs set dedup=on pool
```

Этот пример устанавливает свойство на корневом наборе данных пула *pool*, поэтому каждый набор данных в пуле наследует его. Ограничение дедупликации только наборами данных, которые действительно хранят данные с большим количеством дубликатов, позволяет поддерживать таблицу дедупликации меньшего размера.

Дедупликация затрагивает только новые данные, записываемые в набор данных. Простое включение этой опции не приведёт к дедупликации уже записанных данных. В FreeBSD 15.0 `zfs rewrite` позволяет перезаписать существующие данные так, чтобы они прошли через дедупликацию. Пул с только что активированным свойством дедупликации будет выглядеть следующим образом:

```
# zpool list
NAME  SIZE ALLOC  FREE   CKPOINT  EXPANDSZ   FRAG   CAP   DEDUP   HEALTH   ALTROOT
pool 2.84G 2.19M 2.83G        -         -     0%    0%   1.00x   ONLINE   -
```

Столбец DEDUP показывает фактический уровень дедупликации для пула. Значение 1.00x означает, что данные пока не дедуплицированы. В следующем примере некоторые системные двоичные файлы копируются три раза в разные каталоги в пуле с дедупликацией, созданном выше.

```
# for d in dir1 dir2 dir3; do
> mkdir $d && cp -R /usr/bin $d &
> done
```

Для наблюдения за дедупликацией избыточных данных используйте:

```
# zpool list
NAME SIZE  ALLOC  FREE   CKPOINT  EXPANDSZ   FRAG  CAP   DEDUP   HEALTH   ALTROOT
pool 2.84G 20.9M 2.82G        -         -     0%   0%   3.00x   ONLINE   -
```

Столбец DEDUP показывает коэффициент 3.00x. Обнаружение и дедупликация копий данных используют треть пространства. Потенциальная экономия пространства может быть огромной, но достигается за счет наличия достаточного объёма памяти для отслеживания дедуплицированных блоков. ZFS хранит запись для каждого дедуплицированного блока в таблице дедупликации (DDT). Эмпирическое правило гласит, что требуется 5–6 ГБ оперативной памяти на 1 ТБ дедуплицированных данных. Когда таблица перестаёт помещаться в памяти, каждая запись вынуждает считывать элементы таблицы с диска, и производительность резко падает.

Дедупликация не всегда полезна, если данные в пуле не содержат избыточности. ZFS может показать потенциальную экономию пространства, имитируя дедупликацию на существующем пуле:

```
# zdb -S pool
Simulated DDT histogram:

bucket              allocated                       referenced
______   ______________________________   ______________________________
refcnt   blocks   LSIZE   PSIZE   DSIZE   blocks   LSIZE   PSIZE   DSIZE
------   ------   -----   -----   -----   ------   -----   -----   -----
     1    2.58M    289G    264G    264G    2.58M    289G    264G    264G
     2     206K   12.6G   10.4G   10.4G     430K   26.4G   21.6G   21.6G
     4    37.6K    692M    276M    276M     170K   3.04G   1.26G   1.26G
     8    2.18K   45.2M   19.4M   19.4M    20.0K    425M    176M    176M
    16      174   2.83M   1.20M   1.20M    3.33K   48.4M   20.4M   20.4M
    32       40   2.17M    222K    222K    1.70K   97.2M   9.91M   9.91M
    64        9     56K   10.5K   10.5K      865   4.96M    948K    948K
```

```
    128        2   9.50K      2K      2K       419   2.11M    438K    438K
    256        5   61.5K     12K     12K     1.90K   23.0M   4.47M   4.47M
     1K        2      1K      1K      1K     2.98K   1.49M   1.49M   1.49M
  Total    2.82M    303G    275G    275G     3.20M    319G    287G    287G

dedup = 1.05, compress = 1.11, copies = 1.00, dedup * compress / copies = 1.16
```

После завершения анализа пула командой `zdb -S` отображается коэффициент сокращения пространства, который был бы достигнут при активации дедупликации. В данном случае значение `1.16` указывает на низкий уровень экономии пространства, в основном обеспечиваемый сжатием. Активация дедупликации для этого пула не сэкономит значительного объёма пространства и не оправдает объём памяти, необходимый для её включения. Используя формулу *ratio = dedup * compress / copies*, системные администраторы могут планировать распределение хранилища, определяя, будет ли рабочая нагрузка содержать достаточное количество дублирующихся блоков, чтобы оправдать требования к памяти. Если данные достаточно хорошо сжимаемы, экономия пространства может быть значительной.

Быстрое дедуплицирование, переработанная реализация, снижающая эти затраты, требует OpenZFS 2.3 или более поздней версии, впервые доступной в FreeBSD 15.0. Быстрое дедуплицирование группирует обновления таблицы дедупликации в журнале на диске и сбрасывает их в таблицу в фоновом режиме, снижая случайный ввод-вывод, из-за которого классическая дедупликация была медленной. Свойство пула `dedup_table_quota` задаёт ограничение на размер таблицы на диске; ZFS прекращает добавлять новые записи, когда пул достигает лимита, а при значении по умолчанию `auto` размер выделенного vdev для дедупликации служит квотой. Свойство пула `dedup_table_size`, доступное только для чтения, сообщает текущий размер таблицы. `zpool ddtprune -p 30 pool` удаляет самые старые 30% записей таблицы, которые не дают экономии, поскольку на них ссылается только один блок; `-d days` вместо этого удаляет записи с одной ссылкой, старше указанного количества дней. `zpool status -DD pool` отображает статистику таблицы дедупликации, включая гистограмму блоков по количеству ссылок. Хранение таблицы на выделенном для дедупликации vdev, состоящем из быстрых устройств, обеспечивает быстрый поиск даже когда таблица превышает объём оперативной памяти.

Хорошей практикой является сначала включить сжатие, так как сжатие также обеспечивает значительно повышенную производительность. Включайте дедупликацию в случаях, когда экономия значительна и имеется достаточно доступной памяти для DDT.

23.4.17. ZFS и клетки

Используйте `zfs jail` и соответствующее свойство `jailed`, чтобы делегировать набор данных ZFS в Клетку. `zfs jail идентификатор_клетки` присоединяет набор данных к указанной клетке, а `zfs unjail` отсоединяет его. Для управления набором данных изнутри клетки установите свойство `jail`. ZFS запрещает монтирование на хосте набора данных со свойством jail, так как его точки монтирования могут нарушить безопасность хоста.

Делегированный набор данных передает управление всем поддеревом администратору клетки, который может монтировать его, создавать дочерние наборы данных, делать

снимки и изменять свойства, все это ограничено этим поддеревом. Для этого клетке необходимо включить параметры `allow.mount` и `allow.mount.zfs`, а параметр `enforce_statfs` установить на значение ниже 2. Обратитесь к jail(8) за подробностями об этих параметрах.

Этот пример создаёт набор данных, помечает его как находящийся в клетке и прикрепляет его к запущенной клетке *myjail*:

```
# zfs create mypool/jaildata
# zfs set jailed=on mypool/jaildata
# zfs jail myjail mypool/jaildata
```

Затем набор данных появляется в `zfs list` внутри клетки и доступен для управления там. `zfs unjail myjail mypool/jaildata` возвращает управление хосту. Дополнительную информацию см. в zfs-jail(8).

Начиная с FreeBSD 15.0, jail(8) автоматизирует подключение. Укажите наборы данных для делегирования в параметре клетки `zfs.dataset`, и jail(8) подключает их при запуске клетки. Наборы данных уже должны иметь установленный параметр `jailed=on`, а параметр требует `allow.mount.zfs`. Запись в /etc/jail.conf с его использованием выглядит так:

```
myjail {
    path = "/usr/local/jails/myjail";
    host.hostname = "myjail.example.org";
    exec.start = "/bin/sh /etc/rc";
    exec.stop = "/bin/sh /etc/rc.shutdown";
    allow.mount;
    allow.mount.zfs;
    enforce_statfs = 1;
    zfs.dataset = "mypool/jaildata";
}
```

Установка параметра клетки `zfs.mount_snapshot` в значение 1 дополнительно позволяет пользователям внутри клетки получать доступ к содержимому снимков ZFS в каталоге .zfs делегированных файловых систем.

23.5. Делегированное администрирование

Комплексная система делегирования прав позволяет непривилегированным пользователям выполнять функции администрирования ZFS. Например, если домашний каталог каждого пользователя является набором данных, пользователям нужно разрешение на создание и удаление снимков своих домашних каталогов. Пользователь, выполняющий резервное копирование, может получить разрешение на использование функций репликации. ZFS позволяет скрипту статистики использования работать с доступом только к данным о занятом пространстве для всех пользователей. Также возможно делегирование права на делегирование разрешений. Делегирование прав доступно для каждой подкоманды и большинства свойств.

23.5.1. Делегирование создания наборов данных

`zfs allow someuser create mydataset` предоставляет указанному пользователю разрешение на создание дочерних наборов данных в выбранном родительском наборе данных. Важное замечание: создание нового набора данных включает его монтирование. Для этого необходимо установить параметр `vfs.usermount` в sysctl(8) FreeBSD в значение 1, чтобы разрешить непривилегированным пользователям монтировать файловую систему. Ещё одно ограничение, направленное на предотвращение злоупотреблений: непривилегированные пользователи должны быть владельцами точки монтирования, куда монтируется файловая система.

23.5.2. Делегирование права делегировать разрешения

`zfs allow someuser allow mydataset` предоставляет указанному пользователю возможность назначать любые разрешения, которые у него есть для целевого набора данных или его дочерних элементов, другим пользователям. Если пользователь имеет разрешение `snapshot` и разрешение `allow`, он может предоставить разрешение `snapshot` другим пользователям.

23.6. Встроенное шифрование ZFS

ZFS поддерживает встроенное шифрование наборов данных и данных, хранящихся в них. Так было не всегда: предыдущие решения полагались на шифрование на основе GELI во FreeBSD. Хотя этот подход безопасен, он не был так легко переносим на системы, отличные от FreeBSD, в которых отсутствовала поддержка GELI. Благодаря встроенному шифрованию ZFS зашифрованные наборы данных можно использовать на других системах, поддерживающих эту функцию пула, без зависимости от сторонних компонентов или встроенного шифрования операционной системы.

Ещё одним преимуществом встроенного шифрования ZFS является то, что наборы данных не нужно расшифровывать для административных задач, таких как создание снимков, репликация или операции проверки целостности. Проверки целостности данных ZFS работают одинаково хорошо с нативно зашифрованными наборами данных, а сжатие также работает с зашифрованными наборами данных.

Разные наборы данных в одном пуле могут использовать отдельные ключи независимо. Рассмотрим центральный файловый сервер, где разные пользователи хранят свои данные в зашифрованных домашних наборах данных. У пользователя A ключ отличается от ключа пользователя B. Оба пользователя могут работать параллельно и расшифровывать свои наборы данных по мере необходимости, не зная ключей расшифровки друг друга. Метод расшифровки также может полностью различаться между пользователями, не влияя друг на друга. Это позволяет хранить конфиденциальные данные разных пользователей в одном пуле, оставаясь защищёнными отдельными ключами.

Перед использованием встроенного шифрования ZFS, учтите следующее:

- Шифрование применяется на уровне набора данных, а не на уровне пула.
- Загрузка с зашифрованных наборов данных ZFS пока не поддерживается в загрузчике FreeBSD.

- Шифрование может быть включено только при создании набора данных, а не после.
- Блочный шифр, длина ключа и режим шифрования не могут быть изменены после того, как они были установлены.
- ZFS не шифрует метаданные.

Последний пункт может показаться недостатком по сравнению с полнодисковым шифрованием. Однако он обеспечивает большую гибкость, позволяя разным ключам защищать наборы данных (datasets) в рамках одного пула, вместо того чтобы требовать один ключ или парольную фразу для разблокировки всего пула. В частности, ZFS шифрует следующее:

- данные файла и zvol,
- атрибуты файлов,
- ACL,
- биты разрешений,
- листинги каталогов,
- Соответствия FUID,
- используемые пользователем/групповые используемые данные, и
- Дедуплицированные данные

Некоторые метаданные и информация ZFS остаются незашифрованными, включая следующее:

- структура и имя пула,
- имена наборов данных и снимков,
- иерархия наборов данных,
- свойства,
- размер файла
- пропуски в файлах, и
- Таблицы дедупликации

Этот набор незашифрованных метаданных позволяет ZFS выполнять рутинные операции обслуживания, такие как `zpool scrub`, без предварительной расшифровки данных. В результате ZFS может обнаруживать и исправлять повреждённые зашифрованные данные, не обращаясь к незашифрованному содержимому файлов.

Чтобы зашифровать существующие незашифрованные данные, скопируйте их в набор данных с включенным шифрованием. Это позволяет перенести данные в защищенное место в пуле, при условии предоставления парольной фразы для авторизации доступа. Обычно это делается путем загрузки ключа шифрования в память после подтверждения знания секрета, такого как парольная фраза. После загрузки ключа содержимое набора данных становится доступным для чтения и записи. Чтобы снова защитить набор данных, выгрузите ключ из памяти. Тогда ZFS сделает набор данных недоступным до тех пор, пока

ключ не будет загружен снова.

Когда шифрование включено, некоторые операции ZFS ведут себя иначе. ZFS применяет сжатие перед шифрованием, чтобы сохранить коэффициенты сжатия. Хотя ZFS обычно использует 256-битные контрольные суммы, шифрование заменяет их на 128-битную контрольную сумму и 128-битный код аутентификации сообщения (MAC), предоставляемый набором шифрования. Это обеспечивает дополнительную защиту от злонамеренного изменения данных.

Когда дедупликация используется вместе с шифрованием, ZFS выполняет дедупликацию только в пределах зашифрованного набора данных, его снимков и его клонов. Это предотвращает утечку информации между зашифрованными наборами данных. Обратной стороной является более низкая эффективность дедупликации, поскольку ZFS не может сравнивать контрольные суммы по всему пулу. Тем не менее, дедупликация в сочетании с шифрованием всё равно раскрывает, какие блоки идентичны, и добавляет дополнительные накладные расходы на ЦП для каждого записываемого блока.

У шифрования ZFS также есть и другие ограничения. Возможность `embedded_data` не может использоваться с шифрованием. Наборы данных с включенным шифрованием также не могут иметь свойство `copies`, установленное в 3, поскольку реализация хранит зашифрованные метаданные в том месте, где обычно размещалась бы третья копия.

23.6.1. Создание зашифрованного набора данных

Шифрование ZFS включается путем установки свойства `encryption=on` при создании набора данных. Установка этого свойства вне команды `zfs create` не включает шифрование задним числом, поскольку оно действует только во время создания набора данных. В следующем примере для монтирования зашифрованного набора данных после его создания используется парольная фраза.

```
# zfs create -o encryption=on -o keyformat=passphrase -o keylocation=prompt
zroot/secretdata
Enter passphrase:
Re-enter passphrase:
```

Если парольная фраза короче восьми символов, ZFS отклоняет её и не создаёт набор данных. После того как парольная фраза была введена правильно дважды, ZFS создаёт и монтирует набор данных в пуле.

Любые данные, хранящиеся в зашифрованном наборе данных, остаются зашифрованными, пока они там находятся. Копирование данных из зашифрованного набора данных в незашифрованное место расшифровывает данные во время операции копирования. Затем вновь созданный набор данных можно при необходимости смонтировать в другом месте.

```
# zfs set mountpoint=/secretdata zroot/secretdata
# echo "Hello FreeBSD!" > /secretdata/message
# cat /secretdata/message
```

```
Hello FreeBSD!
```

Как и ожидалось, данные в настоящее время доступны для чтения и записи любому, у кого есть доступ к набору данных.

23.6.2. Свойства зашифрованных наборов данных

Перечисление свойств собственного шифрования ZFS для набора данных выполняется с помощью следующей команды:

```
# zfs get encryption,keylocation,keyformat zroot/secretdata
NAME              PROPERTY     VALUE        SOURCE
zroot/secretdata  encryption   aes-256-gcm  -
zroot/secretdata  keylocation  prompt       local
zroot/secretdata  keyformat    passphrase   -
```

Свойство `encryption` показывает `aes-256-gcm`, который в настоящее время является методом шифрования, используемым ZFS по умолчанию. Как указано в столбце `SOURCE`, ни это свойство, ни свойство `keyformat` нельзя изменить с помощью `zfs set` после создания набора данных. Смотрите раздел изменение ключа шифрования для получения информации о том, как изменить некоторые из этих свойств. В следующей таблице перечислены свойства, связанные с шифрованием, используемые ZFS:

encryptionroot	Набор данных, из которого текущий набор данных наследует свой ключ шифрования. Клоны наборов данных разделяют ключ шифрования своего исходного набора данных.
keyformat	Формат ключа шифрования. Возможные значения: `raw`, `hex` и `passphrase`. Как `raw`, так и `hex` ключи должны иметь длину 32 байта и содержать случайные значения.
encryption	Шифр, используемый для шифрования, состоящий из блочного шифра, длины ключа и режима шифрования. Свойства `encryption` и `keyformat` должны быть указаны при создании набора данных. Возможные значения свойства `encryption`: • `off` (по умолчанию), • `on`, • `aes-128-ccm`, • `aes-192-ccm`, • `aes-256-ccm`, • `aes-128-gcm`, • `aes-192-gcm` и • `aes-256-gcm` (текущий стандарт при установке `on`). Это свойство нельзя изменить после того, как оно было установлено.

keystatus	Указывает, был ли успешно загружен ключ шифрования. Возможные значения: `none`, `available` и `unavailable`.
keylocation	Указывает местоположение, из которого следует загрузить ключ шифрования. Возможные значения: • `prompt`, • `file://path/to/file`, • `http://some.address` и • `https://other.address`
pbkdf2iters	Количество итераций, используемых при получении ключа шифрования из парольной фразы. Это вынуждает злоумышленника выполнять большое количество вычислительно затратных операций хеширования. По мере роста вычислительной мощности это значение также следует увеличивать с текущего значения по умолчанию, равного 350000.

23.6.3. Выгрузка ключа шифрования

Защита данных, хранящихся в зашифрованном наборе данных, требует двух шагов: сначала размонтировать набор данных, затем выгрузить ключ шифрования из памяти. Это различие важно. Размонтированный набор данных не защищён, если его ключ шифрования остаётся загруженным в памяти. Всегда выполняйте `zfs unload-key`, чтобы защитить данные. Чтобы снова получить доступ к набору данных, загрузите ключ и укажите парольную фразу, как описано в разделе загрузка ключа шифрования. ZFS предотвращает выполнение `zfs unload-key`, пока набор данных всё ещё смонтирован.

```
# zfs unmount zroot/secretdata
# zfs unload-key zroot/secretdata
```

Выполнение `zfs unload-key` несколько раз приводит к ошибке, указывающей на то, что ключ уже был выгружен.

23.6.4. Загрузка ключа шифрования

Прежде чем зашифрованный набор данных можно будет использовать, как и любой другой набор данных, необходимо загрузить его ключ шифрования и указать парольную фразу. На этом этапе свойство `keystatus` по-прежнему имеет значение `unavailable`, и монтирование набора данных завершается ошибкой, поскольку ключ ещё не был загружен:

```
# zfs get keystatus zroot/secretdata
NAME              PROPERTY   VALUE        SOURCE
zroot/secretdata  keystatus  unavailable  -
# zfs mount zroot/secretdata
cannot mount 'zroot/secretdata': encryption key not loaded
```

Это указывает на то, что набор данных зашифрован и требует ключ шифрования. Чтобы смонтировать набор данных, выполните `zfs load-key` и введите парольную фразу при появлении запроса:

```
# zfs load-key zroot/secretdata
Enter passphrase for 'zroot/secretdata':
# zfs get keystatus zroot/secretdata
NAME              PROPERTY   VALUE      SOURCE
zroot/secretdata  keystatus  available  -
```

Затем смонтируйте набор данных, чтобы сделать его доступным через иерархию файловой системы:

```
# zfs mount zroot/secretdata
# mount|grep secretdata
zroot/secretdata on /secretdata (zfs, local, noatime, nfsv4acls)
```

Чтобы рекурсивно загрузить несколько ключей шифрования, укажите параметр `-r` с командой `zfs load-key`. Если ключ уже был загружен, ZFS сообщает об этом сообщением об ошибке.

```
# zfs load-key zroot/secretdata
Key load error: Key already loaded for 'zroot/secretdata'.
```

Свойство `keystatus` также подтверждает, что ключ был загружен, сообщая значение `available`. Чтобы проверить, корректен ли пароль, не загружая ключ, используйте параметр `-n` для выполнения пробного запуска.

23.6.5. Изменение ключа шифрования

ZFS также поддерживает смену ключей шифрования, например, замену парольной фразы. Эта операция не требует повторного шифрования набора данных. Выполните следующую команду, чтобы изменить парольную фразу шифрования:

```
# zfs change-key zroot/secretdata
Enter new passphrase for 'zroot/secretdata':
Re-enter new passphrase for 'zroot/secretdata':
```

Выполнение `zfs change-key` требует, чтобы текущий ключ был загружен. Если он не загружен, ZFS всё равно запросит новый ключ, но затем выведет предупреждение о том, что текущий ключ не был загружен.

Команда `zfs change-key` также может изменять свойства `keylocation`, `keyformat` и `pbkdf2iters`, указывая их с помощью опции `-o`. Выполнение `zfs change-key` над зашифрованным дочерним набором данных делает его корнем шифрования, если он еще не является

таковым. Чтобы предотвратить такое поведение и продолжить наследование ключа родительского набора, укажите опцию `-i`.

Имейте в виду, что смена ключа шифрования родительского набора данных также меняет ключ для дочерних наборов данных, которые его наследуют. Следовательно, эти дочерние наборы данных продолжают использовать ключ шифрования родителя. Если дочерний набор данных должен использовать другой ключ, либо укажите другой `keyformat` при его создании, либо выполните `zfs change-key` для дочернего набора данных. Это создаст новый корень шифрования для дочернего элемента и разорвёт его наследование шифрования от родителя.

Клоны ZFS зашифрованного набора данных всегда используют ключ шифрования своего исходного набора данных. В результате свойства `keystatus`, `keyformat`, `keylocation` и `pbkdf2iters` не наследуются, как другие свойства набора данных. Вместо этого они используют значения, определённые корнем шифрования. Чтобы определить корень шифрования, используйте свойство `encryptionroot` только для чтения.

Если злоумышленник скомпрометировал ключ шифрования, смена парольной фразы с помощью `zfs change-key` не обязательно защитит существующие или вновь записанные данные. Новые данные продолжают шифроваться тем же мастер-ключом, что и существующие. Если злоумышленник получит как пользовательский ключ, так и соответствующий обёрнутый мастер-ключ, выполнение `zfs change-key` не перезапишет предыдущий мастер-ключ на диске. В результате старый мастер-ключ может оставаться доступным для анализа данных в течение неопределённого периода времени.

Если мастер-ключ был скомпрометирован, предпочтительным решением является безопасное стирание нижележащих устройств хранения и создание нового пула. После этого восстановите данные из резервной копии в новый пул. Или создайте новый зашифрованный набор данных, перенесите данные с помощью `zfs send` и `zfs receive`, а затем выполните `zpool trim --secure`, чтобы стереть освободившееся пространство. Если нижележащее оборудование не поддерживает безопасный TRIM, вместо этого используйте `zpool initialize`.

23.6.6. Отправка зашифрованных наборов данных

Репликация ZFS основана на отправке снимков наборов данных, которые также могут быть созданы из зашифрованных наборов данных. Полученные снимки остаются зашифрованными.

```
# zfs snapshot zroot/secretdata@snap1
```

В дополнение к отправке зашифрованных резервных копий через SSH, использование опции `-w` (raw) с командой `zfs send` передает зашифрованные блоки данных в целевой пул. Сырая отправка обеспечивает несколько преимуществ:

- принимающая система никогда не видит данные в открытом виде.
- принимающая система не требует кодовую фразу, поскольку расшифровка не выполняется.

- Резервные копии можно отправлять без предварительной загрузки ключа шифрования.
- Ненадёжная система может получить зашифрованные данные, но не может расшифровать или изменить их без ключа шифрования.

Чтобы отправить созданный выше снимок в виде необработанного потока в набор данных `secret` в пуле `backup`, введите следующую команду:

```
# zfs send -w zroot/secretdata@snap1 | zfs recv backup/secret
```

Без опции `-w` команда `zfs send` передает данные в незашифрованном виде. Принимающая система затем может повторно зашифровать данные, используя другой ключ. Однако это предотвращает будущие инкрементальные «сырые» отправки на это место назначения.

23.6.7. Загрузка ключей шифрования при загрузке

Скрипт `zfskeys` из rc(8) загружает ключи шифрования при загрузке для зашифрованных наборов данных, чьё свойство `keylocation` указывает на файл ключа с URL-адресом `file://`. Скрипт запускается после импорта пула и перед монтированием файловых систем, поэтому наборы данных, обрабатываемые таким образом, монтируются автоматически во время обычного запуска ZFS. Скрипт не обрабатывает наборы данных с `keylocation=prompt`; загружайте их ключи вручную с помощью `zfs load-key` после загрузки системы.

Чтобы переключить набор данных, созданный ранее, с парольной фразы на файл ключа, сгенерируйте 32 байта случайных данных в качестве нового ключа, ограничьте доступ к файлу ключа и измените ключ шифрования набора данных:

```
# dd if=/dev/random of=/root/secretdata.key bs=32 count=1
# chmod 600 /root/secretdata.key
# zfs change-key -o keyformat=raw -o keylocation=file:///root/secretdata.key
zroot/secretdata
```

Включите скрипт в файле /etc/rc.conf:

```
# sysrc zfskeys_enable="YES"
```

По умолчанию сценарий загружает ключи всех зашифрованных наборов данных с расположением ключей `file://`. Чтобы ограничить загрузку ключей конкретными наборами данных, перечислите их в `zfskeys_datasets`, разделяя пробелами:

```
# sysrc zfskeys_datasets="zroot/secretdata"
```

Любой, кто может прочитать файл ключа, сможет расшифровать набор данных. Держите файл ключа принадлежащим `root` с правами `600` и храните его на хранилище, которое само защищено, например, на зашифрованной

> корневой файловой системе или сменном носителе, который подключается только во время загрузки.

23.7. Загрузочные окружения

Загрузочное окружение — это загрузочный клон дерева наборов данных, которое содержит операционную систему. На системах, установленных с раскладкой Root-on-ZFS, эти наборы данных находятся в `zroot/ROOT`, а работающая система — в `zroot/ROOT/default`. Создание загрузочного окружения создаёт снимки и клоны корневых наборов данных, что занимает секунды и почти не потребляет места на диске, пока окружения не начнут расходиться.

Управление загрузочными окружениями с помощью bectl(8). Создайте новое окружение перед серьезным изменением, таким как обновление операционной системы или крупного пакета:

```
# bectl create beforeupgrade
# bectl list
BE            Active Mountpoint Space Created
default       NR     /          2.43G 2026-07-11 11:26
beforeupgrade -      -          328K  2026-07-11 11:31
```

В столбце `Active` метка `N` обозначает окружение, используемое в данный момент, а `R` — то, которое станет активным после перезагрузки.

Выполните обновление с помощью freebsd-update(8) или pkg(8) как обычно. Обновление изменяет активное окружение, тогда как `beforeupgrade` сохраняет систему в её прежнем состоянии. Если обновлённая система не загружается или работает некорректно, перезагрузитесь и выберите `beforeupgrade` из меню `Boot Environments` загрузчика FreeBSD. Выбор окружения в меню загрузчика влияет только на текущую загрузку. Чтобы сделать откат постоянным, активируйте это окружение:

```
# bectl activate beforeupgrade
```

Выполнение `bectl activate -t beName` вместо этого активирует окружение только для следующей загрузки, что полезно при тестировании изменений на удаленной системе.

Чтобы осмотреть или восстановить содержимое неактивного окружения без его загрузки, смонтируйте его во временное расположение:

```
# bectl mount beforeupgrade
/tmp/be_mount.c1Xk
# bectl umount beforeupgrade
```

Уничтожьте окружение, которое больше не нужно, чтобы освободить занимаемое окружением пространство:

```
# bectl destroy beforeupgrade
```

Создание загрузочного окружения перед каждым freebsd-update(8) upgrade или крупной операцией pkg(8) обеспечивает мгновенный способ вернуться к заведомо исправной системе без восстановления из резервных копий.

23.7.1. Загрузочные окружения и приложения

Загрузочное окружение содержит только наборы данных в `zroot/ROOT`. В стандартной компоновке Root-on-ZFS это включает всю корневую файловую систему с /usr/local и /var/db, поэтому установленные пакеты и их метаданные откатываются вместе с операционной системой. Наборы данных за пределами `zroot/ROOT`, такие как `zroot/home`, `zroot/var/log` и `zroot/var/mail`, являются общими: каждое окружение видит одни и те же файлы, и активация более старого окружения не возвращает их в более раннее состояние.

Спланируйте структуру набора данных с учётом этого разделения до того, как полагаться на загрузочные окружения. Данные, которые должны пережить откат, например базы данных приложения, установленного из пакетов, должны находиться в собственном наборе данных вне `zroot/ROOT`:

```
# zfs create -o mountpoint=/var/db/postgres zroot/pgdata
```

При такой компоновке откат к более старой среде возвращает бинарные файлы сервера баз данных к их предыдущей версии, в то время как содержимое базы данных остается нетронутым. Обратное также верно: файлы, записанные в общие наборы данных при тестировании новой среды, сохраняются после переключения обратно, поэтому загрузочные среды не заменяют снимки этих наборов данных.

23.8. Сложные темы

23.8.1. Настройка

Отрегулируйте параметры настройки, чтобы ZFS показывал наилучшую производительность для различных рабочих нагрузок. FreeBSD предоставляет параметры ядра ZFS как переменные sysctl(8) в разделе `vfs.zfs`; zfs(4) документирует каждый параметр и его значение по умолчанию. Установите значения, которые должны применяться с момента загрузки, в /boot/loader.conf; изменяйте значения, доступные для изменения во время работы, с помощью sysctl(8) и сделайте их постоянными в /etc/sysctl.conf.

- `vfs.zfs.arc.max`— верхний размер ARC. Значение по умолчанию `0` позволяет ZFS автоматически определять размер ARC на основе объёма установленной оперативной памяти. Используйте меньшее значение, если в системе работают другие демоны или процессы, которым может потребоваться память. Смотрите «Размер и мониторинг ARC» для получения информации. Изменяйте это значение во время работы с помощью sysctl(8) и задавайте его в /boot/loader.conf или /etc/sysctl.conf.

- `vfs.zfs.arc.min`— нижний размер ARC. Значение по умолчанию `0` позволяет ZFS автоматически выбирать небольшое минимальное значение. Увеличьте это значение, чтобы предотвратить вытеснение всего ARC другими приложениями. Настройка этого значения возможна во время выполнения с помощью sysctl(8), а также в /boot/loader.conf или /etc/sysctl.conf.
- `vfs.zfs.arc.meta_balance` — баланс между кэшированием метаданных и данных файлов в ARC; значения выше `100` всё больше отдают предпочтение метаданным, по умолчанию — `500`. Увеличьте это значение, если рабочая нагрузка связана с операциями над большим количеством файлов и каталогов или частыми операциями с метаданными, ценой уменьшения объёма данных файлов, помещающихся в ARC. Этот настраиваемый параметр заменяет `vfs.zfs.arc.meta_limit`, который был удалён в OpenZFS 2.2. Изменяйте это значение в любое время с помощью sysctl(8).
- `vfs.zfs.vdev.min_auto_ashift` - Минимальное значение `ashift` (размер сектора), автоматически используемое при создании пула. Значение является степенью двойки. Значение по умолчанию `9` соответствует `2^9 = 512`, то есть размеру сектора 512 байт. Чтобы избежать *усиления записи* (write amplification) и получить наилучшую производительность, установите это значение равным наибольшему размеру сектора среди устройств в пуле.

  См. Создание и уничтожение пулов хранения о том, почему пулы на дисках с секторами 4 КБ должны использовать `ashift` со значением 12.

- `vfs.zfs.prefetch.disable` — Отключает предварительную выборку. Значение `0` включает её, а `1` отключает. По умолчанию используется `0`. Предварительная выборка работает, считывая блоки большего размера, чем запрошено, в раздел ARC, в надежде, что данные скоро понадобятся. Если рабочая нагрузка включает большое количество случайных чтений, отключение предварительной выборки может улучшить производительность, сократив ненужные чтения. Это значение можно изменить в любое время с помощью sysctl(8).
- `vfs.zfs.txg.timeout` — Максимальное количество секунд между группами транзакций. Текущая группа транзакций записывается в пул, и начинается новая группа транзакций, если с момента предыдущей группы транзакций прошло указанное время. Группа транзакций может запуститься раньше при записи достаточного объёма данных. Значение по умолчанию составляет 5 секунд. Увеличение этого значения может улучшить производительность чтения за счет задержки асинхронных записей, но это может привести к неравномерной производительности при записи группы транзакций. Это значение можно изменить в любое время с помощью sysctl(8).
- `vfs.zfs.l2arc.write_max` — Ограничивает объём данных, записываемых в L2ARC в секунду. Этот параметр увеличивает срок службы SSD, ограничивая объём данных, записываемых на устройство. Значение можно изменить в любой момент с помощью sysctl(8).
- `vfs.zfs.l2arc.write_boost` — Добавляет значение этого параметра к `vfs.zfs.l2arc.write_max` и увеличивает скорость записи на SSD до вытеснения первого блока из L2ARC. Эта "Фаза турборазогрева" снижает потерю производительности из-за пустого L2ARC после перезагрузки. Значение можно изменить в любое время с помощью sysctl(8).
- `vfs.zfs.l2arc.rebuild_enabled` - Обеспечивает сохранение содержимого L2ARC при

перезагрузках. При значении по умолчанию 1, ZFS перестраивает L2ARC из журнальных блоков, хранящихся на устройстве кэша, при импорте пула, избегая длительного периода прогрева, который потребовался бы пустому кэшу. Установите это значение в 0, чтобы всегда начинать с пустого L2ARC.

- vfs.zfs.vdev.max_active — максимальное количество активных запросов ввода-вывода на каждом устройстве в пуле. Более высокое значение поддерживает очередь команд устройства заполненной и может увеличить пропускную способность. Более низкое значение уменьшает задержки. Это значение можно изменить в любое время с помощью sysctl(8).
- vfs.zfs.vdev.*_min_active и vfs.zfs.vdev.*_max_active — нижний и верхний предел количества одновременных запросов ввода-вывода, которые планировщик отправляет каждому устройству для каждого класса ввода-вывода: sync_read, sync_write, async_read, async_write, scrub и других. Эти ограничения для каждого класса заменяют основанное на задержках ограничение scrub и resilver, которое использовалось в более старых версиях ZFS. Например, увеличение vfs.zfs.vdev.scrub_max_active позволяет операциям проверки (scrub) использовать большую долю очереди каждого устройства, завершаясь быстрее за счет увеличения задержки операций ввода-вывода переднего плана.
- `*vfs.zfs.resilver_min_time_ms*`_ — минимальное количество времени в миллисекундах, которое resilver тратит на работу между сбросами групп транзакций; по умолчанию равно 3000 (3 секунды). Увеличьте это значение, чтобы завершить процесс восстановления (resilver) быстрее, когда деградировавший пул находится под угрозой потери другого устройства, ценой замедления нормальной работы пула. Изменяйте это значение в любое время с помощью sysctl(8).

23.8.2. Определение размера ARC и мониторинг

ZFS хранит недавно и часто используемые данные и метаданные в оперативной памяти в ARC. ARC растет по мере необходимости до vfs.zfs.arc.max и снова уменьшается, когда другим программам нужна память, поэтому на выделенном сервере хранения нормально, что большая часть оперативной памяти используется. ARC хранит блоки в их сжатом виде на диске и распаковывает их при доступе. Этот *сжатый ARC* позволяет тому же объему оперативной памяти кэшировать больше данных, увеличивая эффективную частоту попаданий.

top(1) отображает текущий размер ARC в строке сводки по памяти. Дерево kstat.zfs.misc.arcstats из sysctl(8) предоставляет подробную статистику, включая текущий размер, целевой максимум, а также счетчики попаданий и промахов:

```
% sysctl kstat.zfs.misc.arcstats.size kstat.zfs.misc.arcstats.c_max
kstat.zfs.misc.arcstats.hits kstat.zfs.misc.arcstats.misses
kstat.zfs.misc.arcstats.size: 8963041280
kstat.zfs.misc.arcstats.c_max: 15843122176
kstat.zfs.misc.arcstats.hits: 954912441
kstat.zfs.misc.arcstats.misses: 21356847
```

filesystems/zfs-stats обобщает эти статистические данные в удобочитаемом отчете.

ARC возвращает память при нехватке ресурсов, но на системах, где ОЗУ разделяется между ZFS и другими крупными потребителями, такими как базы данных, виртуальные машины или клетки, конкуренция может вызывать неравномерную производительность с обеих сторон. Ограничение ARC оставляет предсказуемый объём памяти для других рабочих нагрузок. Например, чтобы ограничить ARC до 8 ГБ, добавьте эту строку в /boot/loader.conf:

```
vfs.zfs.arc.max="8G"
```

Изменение `vfs.zfs.arc.max` с помощью sysctl(8) вступает в силу немедленно, хотя уменьшение теплого ARC освобождает память постепенно.

23.8.3. Синхронные записи, ZIL и SLOG

Каждый пул имеет ZIL. Приложения, запрашивающие синхронную семантику с помощью fsync(2) или флага `O_SYNC`, не получают подтверждение до тех пор, пока ZFS не зафиксирует соответствующую запись журнала намерений в стабильном хранилище. По умолчанию ZIL располагается на обычных vdev пула, поэтому каждая синхронная запись конкурирует с обычным вводом-выводом пула. Асинхронные записи обходят ZIL и записываются со следующей транзакционной группой.

Выделенный перекрестная ссылка:zfs[zfs-term-vdev-log,log] vdev, часто называемый *SLOG* (отдельный журнал), перемещает ZIL на быстрое устройство:

```
# zpool add mypool log mirror nda0 nda1
```

SLOG помогает только рабочим нагрузкам, которые генерируют много синхронных записей, например, NFS-серверам и базам данных. Для чисто асинхронных рабочих нагрузок он не имеет значения.

Несколько гигабайт ёмкости SLOG вполне достаточно: ZIL хранит максимум несколько секунд входящих записей, примерно объём одной группы транзакций, прежде чем ZFS запишет данные в их конечное расположение в пуле. Выбирайте SSD с защитой от потери питания и низкой задержкой устойчивой записи, а также зеркалируйте устройства журнала. ZFS читает ZIL только после сбоя, поэтому потеря незеркалированного SLOG вместе со сбоем системы приводит к потере записей, которые были подтверждены, но ещё не зафиксированы в пуле.

Свойство `sync` набора данных управляет поведением синхронной записи:

- `sync=standard` (по умолчанию) фиксирует синхронные записи в ZIL, а асинхронные записи — со следующей группой транзакций.
- `sync=always` обрабатывает каждую запись как синхронную, жертвуя производительностью ради безопасности, которая помогает только приложениям, которые не запрашивают синхронные записи, когда это необходимо.
- `sync=disabled` обрабатывает каждую запись как асинхронную.

С `sync=disabled` ZFS подтверждает синхронные записи до того, как они достигнут стабильного хранилища. Сам пул остается согласованным, но после сбоя или отключения питания приложения и NFS-клиенты незаметно теряют до нескольких секунд данных, которые, как они считали, были безопасно записаны. Не используйте `sync=disabled` для баз данных, NFS-экспортов или любых других важных данных.

Свойство `logbias` регулирует, как ZFS использует журнал: `logbias=latency` (по умолчанию) предпочитает SLOG для минимизации задержки фиксации, тогда как `logbias=throughput` записывает синхронные данные напрямую в основной пул, оставляя ёмкость SLOG для наборов данных, чувствительных к задержке. Подробности см. в zfsprops(7).

23.8.4. Прямой ввод-вывод

Прямой ввод-вывод требует OpenZFS 2.3 или более поздней версии, впервые доступной в FreeBSD 15.0.

Свойство набора данных `direct` управляет тем, как ZFS обрабатывает запросы, сделанные с флагом `O_DIRECT`:

- `direct=standard` (по умолчанию) учитывает `O_DIRECT`: корректно выровненные чтения и записи обходят ARC.
- `direct=always` обрабатывает каждое правильно выровненное чтение или запись как прямой запрос.
- `direct=disabled` беззвучно игнорирует `O_DIRECT` и обрабатывает каждый запрос через ARC.

Прямой ввод-вывод позволяет избежать копирования в памяти и накладных расходов на кэширование ARC. В пулах очень быстрых NVMe-устройств эти накладные расходы могут доминировать над стоимостью самого ввода-вывода, а приложения, поддерживающие собственные кэши, например базы данных, тратят оперативную память впустую, кэшируя одни и те же данные дважды. Для таких рабочих нагрузок прямой ввод-вывод может увеличить пропускную способность и снизить использование памяти.

ZFS по-прежнему проверяет контрольные суммы для прямых чтений и записей, поэтому защита целостности данных не изменяется. Прямые записи должны быть выровнены по значению `recordsize` набора данных; ZFS незаметно перенаправляет невыровненную часть запроса через ARC. См. zfsprops(7) для получения полных правил и ограничений по выравниванию.

23.8.5. `recordsize` и `volblocksize`

Свойство `recordsize` задаёт наибольший блок, который ZFS использует для хранения файла в наборе данных, по умолчанию 128 КБ. Согласование размера блока с шаблоном ввода-вывода рабочей нагрузки позволяет избежать чтения и записи большего объёма данных, чем запрашивает приложение.

Для баз данных, выполняющих небольшие случайные чтения и записи, установите

`recordsize` равным размеру страницы базы данных, обычно 8 КБ или 16 КБ, до создания файлов базы данных. Для наборов данных, хранящих большие медиафайлы или потоки резервных копий, которые читаются и записываются последовательно, `recordsize=1M` снижает накладные расходы на метаданные и улучшает сжатие. Тома используют свойство `volblocksize` вместо этого, которое по умолчанию составляет 16 КБ и не может быть изменено после создания тома.

Изменение `recordsize` влияет только на новые записанные файлы; существующие файлы сохраняют свой исходный размер блока. Даже `zfs rewrite` не применяет новое значение `recordsize` к существующим файлам, поскольку изменения свойств, влияющие на логический размер блока, не действуют на перезаписанные блоки; скопируйте файлы заново, чтобы сохранить их с новым значением.

23.9. Дополнительные ресурсы

- Документация OpenZFS, включая руководства по производительности и настройке
- OpenZFS
- zfs(4) описывает каждый настраиваемый параметр ядра, zpoolconcepts(7) описывает типы vdev и структуру пула, а zfsprops(7) описывает каждое свойство набора данных
- Инструменты автоматизации снимков и репликации из коллекции портов:
  - filesystems/zrepl - демон репликации, поддерживающий конфигурации push и pull, автоматическое управление снимками и возобновляемые передачи
  - sysutils/sanoid — управление снимками на основе политик, с входящим в комплект `syncoid` для репликации по SSH
  - filesystems/zfstools - автоматическая ротация снимков в стиле службы автоматических снимков OpenSolaris, запускаемая из cron(8)
  - filesystems/zap - простое обслуживание снапшотов и удаленная репликация, настраиваемое через свойства ZFS вместо файла конфигурации

23.10. Особенности и терминология ZFS

ZFS принципиально отличается от традиционных файловых систем. ZFS объединяет функции файловой системы и менеджера томов, позволяя добавлять новые устройства хранения в работающую систему и сразу же использовать новое пространство в существующих файловых системах пула. Благодаря объединению традиционно разделённых ролей, ZFS преодолевает прежние ограничения, которые мешали расширению RAID-групп. Устройство *vdev* — это устройство верхнего уровня в пуле, которое может быть простым диском или RAID-преобразованием, таким как зеркало или массив RAID-Z. Файловые системы ZFS (называемые *наборами данных*) имеют доступ к объединённому свободному пространству всего пула. Используемые блоки из пула уменьшают доступное пространство для каждой файловой системы. Такой подход позволяет избежать распространённой проблемы с разбиением на разделы, когда свободное пространство фрагментируется между разделами.

pool	Пул (*pool*) — это базовая строительная единица ZFS. Оно состоит из одного или нескольких vdev — устройств хранения данных. На основе хранилища создаются файловые системы (наборы данных, datasets) или блочные устройства (тома, volumes). Эти наборы данных и тома используют общий пул свободного пространства. Каждый пул имеет уникальное имя и GUID. Пулы, созданные современной OpenZFS, имеют версию пула 5000; начиная с этой версии, отдельные возможности управляются с помощью *функциональных флагов* (feature flags), а не номера версии. Подробнее см. zpool-features(7) и «Обновление пула хранения».

vdev Types	Пул состоит из одного или нескольких vdev, которые, в свою очередь, представляют собой отдельный диск или группу дисков, преобразованных в RAID. При использовании множества vdev ZFS распределяет данные между ними для повышения производительности и максимизации доступного пространства. Все vdev должны быть размером не менее 64 МБ. • *Диск* — Самый базовый тип vdev — это стандартное блочное устройство. Это может быть целый диск (например, /dev/ada0) или раздел (/dev/ada0p3). В FreeBSD нет потери производительности при использовании раздела вместо целого диска. Это отличается от рекомендаций документации Solaris. Настоятельно не рекомендуется использовать целый диск в составе загружаемого пула, так как это может сделать пул незагружаемым. Аналогично, не следует использовать целый диск в составе зеркала (mirror) или RAID-Z vdev. Надежное определение размера неразмеченного диска во время загрузки невозможно, и нет места для размещения загрузочного кода. • *Файл* — Обычные файлы могут составлять пулы ZFS, что полезно для тестирования и экспериментов. Используйте полный путь к файлу в качестве пути к устройству в команде `zpool create`. • *Зеркало (Mirror)* - При создании зеркала укажите ключевое слово `mirror`, за которым следует список устройств-участников зеркала. Зеркало состоит из двух или более устройств, и все данные записываются на все устройства-участники. Зеркальный vdev будет хранить столько данных, сколько может поместиться на его самом маленьком устройстве. Зеркальный vdev может пережить отказ всех устройств, кроме одного, без потери данных. Чтобы в любой момент обновить обычный vdev с одним диском до зеркального vdev, используйте команду `zpool attach`. • *RAID-Z* - ZFS использует RAID-Z — вариацию стандартного RAID-5, которая обеспечивает лучшее распределение чётности и устраняет проблему "записи дырки RAID-5", когда данные и информация о чётности становятся несогласованными после неожиданного перезапуска. ZFS поддерживает три уровня RAID-Z, которые обеспечивают разную степень избыточности в обмен на уменьшение доступного пространства. ZFS использует RAID-Z1, RAID-Z2 и RAID-Z3 в зависимости от количества устройств чётности в массиве и количества дисков, которые могут отказать до того, как пул перестанет работать. В конфигурации RAID-Z1 с четырьмя дисками по 1 ТБ доступное пространство составляет 3 ТБ, и пул сможет продолжать работу в деградированном режиме при отказе одного диска. Если другой диск выйдет из строя до замены и восстановления отказавшего диска, это приведет к потере всех данных пула. В конфигурации RAID-Z3 с восемью дисками по 1 ТБ том предоставит 5 ТБ

Группы транзакций (TXG)	Группы транзакций — это способ, которым ZFS объединяет изменения блоков и записывает их в пул. Группы транзакций являются атомарной единицей, используемой ZFS для обеспечения согласованности. ZFS назначает каждой группе транзакций уникальный 64-битный последовательный идентификатор. Одновременно может быть до трёх активных групп транзакций, каждая в одном из следующих состояний: * *Открытое (Open)* — Новая группа транзакций начинается в открытом состоянии и принимает новые операции записи. Всегда существует группа транзакций в открытом состоянии, но она может отказать в приёме новых записей, если достигнут лимит. Как только открытая группа транзакций достигает лимита или срабатывает `vfs.zfs.txg.timeout`, группа переходит в следующее состояние. * *Завершающееся (Quiescing)* — Краткое состояние, позволяющее завершить все ожидающие операции без блокировки создания новой открытой группы транзакций. Как только все транзакции в группе завершены, группа переходит в финальное состояние. * *Синхронизируемое (Syncing)* — Запись всех данных группы транзакций в устойчивое хранилище. Этот процесс, в свою очередь, изменяет другие данные, такие как метаданные и карты пространства, которые ZFS также записывает в устойчивое хранилище. Процесс синхронизации включает несколько проходов. На первом и самом большом записываются все изменённые блоки данных; затем идут метаданные, которые могут потребовать нескольких проходов. Поскольку выделение пространства для блоков данных генерирует новые метаданные, состояние синхронизации не может завершиться, пока не будет выполнен проход, не использующий новое пространство. В состоянии синхронизации также завершаются *синхронизационные задачи*. Синхронизационные задачи — это административные операции, такие как создание или удаление снимков и наборов данных, завершающие изменение uberblock. Как только состояние синхронизации завершается, группа транзакций в Завершающемся состоянии переходит в Синхронизируемое. Все административные функции, такие как `снимок`, записываются как часть группы транзакций. ZFS добавляет созданную синхронизационную задачу в открытую группу транзакций, и эта группа как можно быстрее переходит в синхронизируемое состояние, чтобы уменьшить задержку административных команд.

Adaptive Replacement Cache (ARC)	ZFS использует Adaptive Replacement Cache (ARC), а не более традиционный кэш Least Recently Used (LRU). LRU-кэш — это простой список элементов в кэше, отсортированный по времени последнего использования объекта, при этом новые элементы добавляются в начало списка. Когда кэш заполнен, освобождение места для более активных объектов происходит за счёт удаления элементов из конца списка. ARC состоит из четырёх списков: Most Recently Used (MRU) и Most Frequently Used (MFU), а также дополнительных "теневых" списков для каждого из них. Эти теневые списки отслеживают удалённые объекты, чтобы предотвратить их повторное добавление в кэш. Это увеличивает процент попаданий в кэш, исключая объекты, которые использовались лишь изредка. Ещё одно преимущество использования как MRU, так и MFU заключается в том, что сканирование всей файловой системы вытеснило бы все данные из MRU- или LRU-кэша в пользу только что прочитанного содержимого. В ZFS также присутствует MFU, который отслеживает наиболее часто используемые объекты, и кэш наиболее часто запрашиваемых блоков остаётся неизменным.
L2ARC	L2ARC — это второй уровень системы кэширования ZFS. Основной кэш (ARC) хранится в оперативной памяти. Поскольку объём доступной оперативной памяти часто ограничен, ZFS также может использовать кэширующие vdev. Твердотельные накопители (SSD) часто используются в качестве таких кэширующих устройств благодаря их более высокой скорости и меньшей задержке по сравнению с традиционными жёсткими дисками. L2ARC полностью опционален, но его наличие увеличит скорость чтения для кэшированных файлов на SSD, избавляя от необходимости читать с обычных дисков. L2ARC также может ускорить дедупликацию, поскольку таблица дедупликации (DDT - deduplication table), которая не помещается в оперативную память, но помещается в L2ARC, будет работать значительно быстрее, чем DDT, которую должны считать с диска. Начиная с OpenZFS 2.0, L2ARC является постоянным: после перезагрузки ZFS восстанавливает содержимое кэша из заголовков, хранящихся на кэширующем устройстве, вместо того чтобы начинать с пустого кэша. Ограничения на скорость записи данных на кэширующие устройства предотвращают преждевременный износ SSD из-за дополнительных операций записи. Пока кэш не заполнен (первый блок вытеснен для освобождения места), записи в L2ARC ограничиваются суммой лимита записи и лимита ускорения, а после - только лимитом записи. Пара значений sysctl(8) управляет этими ограничениями скорости. `vfs.zfs.l2arc.write_max` контролирует количество байт, записываемых в кэш в секунду,а `vfs.zfs.l2arc.write_boost` добавляется к этому лимиту до первого вытеснения из кэша. Подробнее смотрите раздел Настройка.

ZIL	Журнал намерений ZFS (ZIL) — это запись на диске, которую ZFS использует для воспроизведения синхронных записей после сбоя. Каждый пул имеет ZIL; по умолчанию он занимает небольшой объём пространства на обычных устройствах пула. Когда приложение запрашивает синхронную запись (гарантию того, что данные сохранены на диске, а не просто закэшированы для последующей записи), ZFS сначала фиксирует данные в ZIL, а затем сбрасывает их в обычное хранилище со следующей группой транзакций. Перемещение ZIL на выделенное журнальное виртуальное устройство (SLOG) на устройстве, более быстром, чем основное хранилище, например SSD, значительно снижает задержку и повышает производительность. Синхронные рабочие нагрузки, такие как базы данных, выиграют от одного только быстрого ZIL. Обычные асинхронные записи, например копирование файлов, вообще не используют ZIL. Подробнее см. Синхронные записи, ZIL и SLOG.
Копирование при Записи (Copy-On-Write)	В отличие от традиционной файловой системы, ZFS записывает новый блок вместо перезаписи старых данных на том же месте. После завершения записи метаданные обновляются, указывая на новое местоположение. Если происходит обрыв записи (сбой системы или отключение питания во время записи файла), исходное содержимое файла остаётся доступным, а ZFS отменяет незавершённую запись. Это также означает, что ZFS не требует выполнения fsck(8) после неожиданного выключения.
Набор данных (Dataset)	*Набор данных* — это общий термин для файловой системы ZFS, тома, снимка или клона. Каждый набор данных имеет уникальное имя в формате *имяпула/путь@снимок*. Корень пула также является набором данных. Дочерние наборы данных имеют иерархические имена, подобные каталогам. Например, *mypool/home*, набор данных home, является дочерним для *mypool* и наследует его свойства. Это можно расширить, создав *mypool/home/user*. Этот внучатый набор данных будет наследовать свойства от родительского и вышестоящего наборов данных. Установка свойств для дочернего набора данных позволяет переопределить значения по умолчанию, унаследованные от родительских наборов. Управление наборами данных и их дочерними элементами может быть делегировано.
Файловая система	Набор данных ZFS чаще всего используется как файловая система. Как и большинство других файловых систем, файловая система ZFS монтируется в определённое место иерархии каталогов системы и содержит собственные файлы и каталоги с правами доступа, флагами и другими метаданными.
Том	ZFS также может создавать тома, которые отображаются как дисковые устройства. Тома обладают многими функциями, аналогичными наборам данных, включая копирование при записи, снимки, клоны и контрольные суммы. Тома могут быть полезны для работы других файловых систем поверх ZFS, таких как виртуализация UFS или экспорт областей iSCSI.

Снимок (Snapshot)	Дизайн copy-on-write (COW) в ZFS позволяет создавать почти мгновенные, согласованные снимки с произвольными именами. После создания снимка набора данных или рекурсивного снимка родительского набора, который включит все дочерние наборы, новые данные записываются в новые блоки, но без освобождения старых блоков как свободного пространства. Снимок содержит исходную версию файловой системы, а активная файловая система — все изменения, сделанные после создания снимка, не используя дополнительного пространства. Новые данные, записанные в активную файловую систему, сохраняются в новых блоках. Снимок будет расти по мере того, как блоки перестают использоваться в активной файловой системе и остаются только в снимке. Монтирование этих снимков в режиме только для чтения позволяет восстановить предыдущие версии файлов. Откат активной файловой системы к определённому снимку возможен, отменяя все изменения, произошедшие после создания снимка. Каждый блок в пуле имеет счетчик ссылок, который отслеживает использование этого блока снимками, клонами, наборами данных или томами. По мере удаления файлов и снимков счетчик ссылок уменьшается, освобождая пространство, когда блок больше не используется. Если пометить снимки с помощью удержания (hold), то это приведет к тому, что любая попытка удалить его вернет ошибку `EBUSY`. Каждый снимок может иметь удержания с уникальными именами. Команда release удаляет удержание, позволяя удалить снимок. Снимки, клонирование и откат работают с томами, но независимое монтирование — нет.
Клон (Clone)	Также возможно клонирование снимка. Клон — это доступная для записи версия снимка, позволяющая файловой системе разветвляться как новый набор данных. Как и снимок, клон изначально не занимает дополнительного пространства. По мере записи новых данных в клон используются новые блоки, и размер клона увеличивается. При перезаписи блоков в клонированной файловой системе или томе счетчик ссылок на предыдущий блок уменьшается. Удалить снимок, на котором основан клон, невозможно, потому что клон зависит от него. Снимок является родителем, а клон — потомком. Клоны можно *повышать*, меняя эту зависимость местами, делая клон родителем, а предыдущего родителя — потомком. Эта операция не требует дополнительного пространства. Поскольку объём пространства, используемого родителем и потомком, меняется местами, это может повлиять на существующие квоты и резервирования.
Bookmark	Закладка фиксирует момент времени, в который был сделан снапшот, не сохраняя при этом никаких данных снапшота. Закладки служат источником инкрементальных потоков `zfs send`, что позволяет уничтожить снапшот на отправляющей стороне, сохраняя возможность отправлять дальнейшие инкременты. Создайте закладку из существующего снапшота с помощью `zfs bookmark`. Подробности см. в Закладки.

Hold	Удержание — это именованная метка, помещаемая на снимок, которая предотвращает его уничтожение. Попытки уничтожить удерживаемый снимок завершаются с ошибкой `EBUSY`, пока не будут сняты все удержания с помощью `zfs release`. Удержания защищают снимки, от которых всё ещё зависят инструменты резервного копирования и репликации. Подробнее см. Удержания снимков.
Контрольная сумма	Каждый блок также имеет контрольную сумму. Используемый алгоритм контрольной суммы является свойством для каждого набора данных, см. `set`. Контрольная сумма каждого блока прозрачно проверяется при чтении, что позволяет ZFS обнаруживать скрытые повреждения. Если прочитанные данные не соответствуют ожидаемой контрольной сумме, ZFS попытается восстановить данные из любого доступного резервирования, например, зеркал или RAID-Z. Запуск проверки всех контрольных сумм выполняется с помощью `scrub`. Доступные алгоритмы контрольных сумм включают: * `fletcher2` * `fletcher4` * `sha256` * `sha512` * `skein` * `edonr` * `blake3` (since FreeBSD 14.0) Алгоритмы `fletcher` работают быстрее, но криптографические хеши (`sha256`, `sha512`, `skein`, `edonr` и `blake3`) имеют значительно меньшую вероятность коллизий ценой некоторого снижения производительности. Значение по умолчанию `on` в настоящее время выбирает `fletcher4`; дедупликация требует криптографического хеша, например `sha256`. Возможно отключение контрольных сумм, но это крайне не рекомендуется.

Сжатие (compression)	Каждый набор данных имеет свойство сжатия (compression). Начиная с OpenZFS 2.2, для новых наборов данных сжатие включено по умолчанию (`compression=on`, что выбирает алгоритм LZ4). Установите это свойство в один из доступных алгоритмов сжатия. Это приведёт к сжатию всех новых данных, записываемых в набор данных. Помимо уменьшения используемого пространства, пропускная способность при чтении и записи часто увеличивается, поскольку требуется читать или записывать меньше блоков. * *LZ4* — добавлен в версии 5000 (флаги функций) пула ZFS, LZ4 — это алгоритм, который выбирается параметром `compression=on`, и является рекомендуемым выбором для общего назначения. LZ4 работает примерно на 50% быстрее, чем LZJB, при работе с сжимаемыми данными и более чем в три раза быстрее при работе с несжимаемыми данными. LZ4 также распаковывает данные примерно на 80% быстрее, чем LZJB. На современных процессорах LZ4 часто может сжимать данные со скоростью более 500 МБ/с и распаковывать со скоростью более 1,5 ГБ/с (на одно ядро CPU). * *ZSTD* — Zstandard обеспечивает более высокую степень сжатия, чем LZ4, с настраиваемым компромиссом между скоростью и степенью сжатия, начиная от быстрых вариантов `zstd-fast` и заканчивая `zstd-19` для максимальной экономии места. Подробнее см. «Сжатие Zstandard». * *LZJB* — устаревший алгоритм сжатия, создан Джеффом Бонвиком (одним из первоначальных создателей ZFS). LZJB обеспечивает хорошее сжатие с меньшей нагрузкой на CPU по сравнению с GZIP. LZ4 заменил его в качестве алгоритма по умолчанию, для новых наборов данных предпочитайте LZ4 или ZSTD. * *GZIP* — популярный алгоритм потокового сжатия, доступный в ZFS. Одним из основных преимуществ использования GZIP является настраиваемый уровень сжатия. При установке свойства `compress` администратор может выбрать уровень сжатия от `gzip1` (минимальный уровень сжатия) до `gzip9` (максимальный уровень сжатия). Это позволяет администратору контролировать баланс между использованием CPU и экономией дискового пространства. * *ZLE* — Zero Length Encoding (кодирование нулевой длины) — это специальный алгоритм сжатия, который сжимает только непрерывные последовательности нулей. Этот алгоритм полезен, когда набор данных содержит большие блоки нулей.
Копии (copies)	Когда свойству `копии (copies)` присваивается значение больше 1, ZFS сохраняет копии каждого блока в файловой системе или томе. Установка этого свойства для важных наборов данных обеспечивает дополнительную избыточность, позволяющую восстановить блок, контрольная сумма которого не совпадает. В пулах без избыточности функция копирования является единственной формой избыточности. Функция копирования позволяет восстановиться после повреждения отдельного сектора или других незначительных повреждений, но не защищает пул от потери всего диска.

Дедубликация (Deduplication)	Контрольные суммы позволяют обнаруживать дублирующиеся блоки при записи данных. При дедупликации счетчик ссылок существующего идентичного блока увеличивается, что экономит место на диске. ZFS хранит таблицу дедупликации (DDT) в памяти для обнаружения дублирующихся блоков. Таблица содержит список уникальных контрольных сумм, расположение этих блоков и счетчик ссылок. При записи новых данных ZFS вычисляет контрольные суммы и сравнивает их со списком. При обнаружении совпадения используется существующий блок. Использование алгоритма контрольной суммы SHA256 с дедупликацией обеспечивает безопасное криптографическое хеширование. Дедупликация настраивается. Если `dedup` установлен в `on`, то совпадение контрольных сумм означает, что данные идентичны. Если `dedup` установлен в `verify`, ZFS выполняет побайтовую проверку данных, гарантируя их полное совпадение. Если данные не идентичны, ZFS зафиксирует коллизию хешей и сохранит два блока отдельно. Поскольку DDT должна хранить хеш каждого уникального блока, она потребляет значительный объём памяти. Общее правило — 5–6 ГБ оперативной памяти на 1 ТБ дедуплицированных данных. В ситуациях, когда невозможно иметь достаточно памяти для хранения всей DDT в оперативной памяти, производительность сильно снизится, так как DDT будет считываться с диска перед записью каждого нового блока. Дедупликация может использовать L2ARC для хранения DDT, что представляет собой компромисс между быстрой системной памятью и медленными дисками. Рекомендуется также рассмотреть использование сжатия, которое часто обеспечивает почти такую же экономию места без увеличения потребления памяти. *Быстрая дедупликация* (fast dedup) группирует обновления DDT в журнале, позволяет удалять устаревшие записи с помощью `zpool ddtprune` и ограничивает размер таблицы свойством пула `dedup_table_quota`; для неё требуется OpenZFS 2.3 или новее, впервые доступная в FreeBSD 15.0. Подробнее см. «Дедупликация».
Block Cloning	Блочное клонирование позволяет копии файла ссылаться на блоки исходного файла, существующие на диске, вместо их дублирования, поэтому копия выполняется почти мгновенно и не занимает дополнительное место, пока ни один из файлов не изменится. На FreeBSD cp(1) использует copy_file_range(2), поэтому обычные копии файлов становятся блочными клонами, когда эта функция включена (по умолчанию в FreeBSD 15.0). Свойства пула `bcloneused`, `bclonesaved` и `bcloneratio` сообщают о полученной экономии места. Подробности см. в Блочное клонирование.

Scrub	Вместо проверки согласованности, такой как fsck(8), в ZFS используется `scrub`. `scrub` читает все блоки данных в пуле и проверяет их контрольные суммы по сравнению с известными корректными контрольными суммами, хранящимися в метаданных. Периодическая проверка всех данных в пуле гарантирует восстановление повреждённых блоков до того, как они понадобятся. Проведение `scrub` не требуется после некорректного завершения работы, но рекомендуется выполнять её хотя бы раз в месяц. ZFS проверяет контрольные суммы каждого блока при обычном использовании, но `scrub` обеспечивает проверку даже редко используемых блоков на тихую порчу данных. ZFS повышает безопасность данных в архивных системах хранения. Приостановите выполняющуюся проверку (scrub) с помощью `zpool scrub -p` и возобновите её позже, снова запустив `zpool scrub`. Подробнее см. «Проверка пула».
Checkpoint	Контрольная точка сохраняет полное состояние пула в единый момент времени, включая изменения конфигурации всего пула, которые снимки не могут зафиксировать. Возврат к контрольной точке с помощью `zpool import --rewind-to-checkpoint` отменяет всё, что произошло после её создания, например, включение функций пула, добавление или удаление vdev, или уничтожение наборов данных. Пул может иметь не более одной контрольной точки одновременно. Подробности см. в Контрольные точки пула.
TRIM	TRIM уведомляет SSD и другие накопители на флеш-памяти о блоках, которые пул больше не использует, позволяя устройству заранее стирать их и поддерживать производительность записи. Запустите `zpool trim` вручную или установите свойство пула `autotrim` в значение `on`, чтобы непрерывно выполнять TRIM для освобожденного пространства. Подробности см. в TRIM и инициализация.
Квота набора данных	ZFS обеспечивает быстрый и точный учёт пространства для наборов данных, пользователей и групп, а также квоты и резервирование пространства. Это даёт администратору детальный контроль над распределением пространства и позволяет резервировать место для критически важных файловых систем. ZFS поддерживает различные типы квот: квоту набора данных, референтную квоту (refquota), пользовательскую квоту и групповую квоту. Квоты ограничивают общий размер набора данных и его потомков, включая снимки набора данных, дочерние наборы данных и снимки этих наборов данных. Тома не поддерживают квоты, так как свойство `volsize` действует как неявная квота.
Квота ссылки	Референтная квота ограничивает объём пространства, который может занимать набор данных, устанавливая жёсткий лимит. Этот жёсткий лимит включает только пространство, на которое ссылается сам набор данных, и не учитывает пространство, используемое его потомками, такими как файловые системы или снимки.

Квота пользователя	Пользовательские квоты полезны для ограничения объёма пространства, используемого указанным пользователем.
Квота группы	Квота группы ограничивает объём пространства, который может использовать указанная группа.
Резервирование для набора данных	Свойство `reservation` позволяет гарантировать определённый объём пространства для конкретного набора данных и его потомков. Это означает, что установка резерва в 10 ГБ для storage/home/bob предотвращает использование всего свободного пространства другими наборами данных, резервируя как минимум 10 ГБ для этого набора. В отличие от обычного `reservation`, при `refreservation` пространство, используемое снимками и потомками наборов данных, не учитывается в резерве. Резервирование любого типа полезно в таких ситуациях, как планирование и тестирование распределения дискового пространства в новой системе, или для обеспечения достаточного места в файловых системах для аудио-логов, процедур восстановления системы и файлов.
Референтное резервирование	Свойство `refreservation` позволяет гарантировать определённый объём пространства для использования конкретным набором данных *исключая* его потомков. Это означает, что если установить резервирование в 10 ГБ для storage/home/bob, а другой набор данных попытается использовать свободное пространство, то он оставит как минимум 10 ГБ, зарезервированных для storage/home/bob. В отличие от обычного резервирования, пространство, используемое снимками и наборами-потомками, не учитывается в резервировании. Например, при создании снимка storage/home/bob для успешного выполнения операции должно быть достаточно места на диске помимо объёма, указанного в `refreservation`. Наборы данных-потомки не учитываются в объёме `refreservation` и, следовательно, не уменьшают зарезервированное пространство.
Boot Environment	Загрузочное окружение — это загрузочная копия наборов данных, содержащих систему FreeBSD, управляемая с помощью bectl(8). Создание загрузочного окружения перед обновлением системы позволяет загрузить предыдущую, заведомо рабочую систему из меню загрузчика, если обновление вызовет проблемы. Подробности см. в Загрузочные окружения.
Ресильверинг	При замене вышедшего из строя диска ZFS должен заполнить новый диск утерянными данными. *Ресилверинг* — это процесс записи этих данных на заменяющее устройство: зеркала выполняют ресилверинг путём копирования данных с выживших участников, а RAID-Z восстанавливает их из информации о чётности, распределённой по оставшимся дискам. В отличие от традиционной перестройки RAID, ресилверинг копирует только выделенные данные, а не каждый сектор устройства. *Последовательный ресилверинг* (`zpool attach -s` или `zpool replace -s`) перестраивает зеркальные и dRAID vdev-ы ещё быстрее, копируя блоки в порядке диска, пропуская проверку контрольных сумм и откладывая её на автоматически запланированную проверку (scrab). Подробности см. в Dealing with Failed Devices.

Online	Пул или vdev в состоянии `Online` имеет подключённые и полностью работоспособные устройства-участники. Отдельные устройства в состоянии `Online` функционируют.
Offline	Администратор переводит отдельные устройства в состояние `Offline`, если существует достаточная избыточность, чтобы избежать перевода пула или vdev в состояние Faulted. Администратор может отключить диск для подготовки к его замене или для упрощения идентификации.
Degraded	Пул или vdev в состоянии `Degraded` имеет один или несколько дисков, которые исчезли или вышли из строя. Пул по-прежнему можно использовать, но если другие устройства выйдут из строя, пул может стать невосстановимым. Повторное подключение отсутствующих устройств или замена неисправных дисков вернет пул в состояние Online после того, как повторно подключённое или новое устройство завершит процесс ресильверинга.
Faulted	Пул или vdev в состоянии `Faulted` больше не функционирует. Доступ к данным становится невозможным. Пул или vdev переходит в состояние `Faulted`, когда количество отсутствующих или неисправных устройств превышает уровень избыточности vdev. Если отсутствующие устройства будут снова подключены, пул вернётся в состояние Online. Недостаточная избыточность для компенсации количества неисправных дисков приводит к потере содержимого пула и требует восстановления из резервных копий.
Unavail	Устройство в состоянии `Unavail` не может быть открыто, поскольку оно отсутствует или недоступно. Пул продолжает работать, если имеется достаточная избыточность для компенсации, функционируя в состоянии Degraded (деградировавшем). Если вышестоящий vdev находится в состоянии `Unavail`, доступ к содержимому пула невозможен.
Removed	Устройство в состоянии `Removed` было физически отсоединено во время работы системы. Обнаружение удаления устройства зависит от аппаратного обеспечения и не поддерживается на некоторых платформах. Когда устройство переподключается, zfsd(8) возвращает его в онлайн; если устройство не возвращается, zfsd активирует доступный горячий резерв вместо него.

Глава 24. Поддержка файловых систем

24.1. Обзор

Файловые системы являются фундаментальным компонентом любой операционной системы. Они позволяют пользователям сохранять, управлять и получать доступ к данным, делая устройства хранения, такие как жёсткие диски, флеш-накопители и USB-устройства, практичными для повседневного использования. Разные операционные системы используют разные файловые системы в своей основе.

Традиционно FreeBSD использует Unix File System (UFS), а её современную версию UFS2 — в качестве основной родной файловой системы. FreeBSD также поддерживает Файловую Систему Z (Z File System — ZFS), известную благодаря своим расширенным возможностям, надёжности и отказоустойчивости. Подробнее см. The Z File System (ZFS).

Помимо собственных файловых систем, FreeBSD поддерживает широкий спектр файловых систем из других операционных систем. Поддержка этих файловых систем варьируется: для некоторых требуется загрузка модулей ядра, в то время как для других необходимы дополнительные пользовательские утилиты.

Прежде чем читать эту главу, необходимо:

- Знать концепции UNIX® и основы FreeBSD.
- Свободно устанавливать программное обеспечение через установку программ в FreeBSD.
- Иметь некоторое представление о дисках, устройствах хранения данных и соглашениях FreeBSD по именованию устройств.

Прочитайте эту главу, чтобы:

- Понять различия между родными и поддерживаемыми файловыми системами.
- Знать, какие файловые системы поддерживаются FreeBSD и уровень доступной поддержки.
- Узнать, как включить, настроить, получить доступ и работать с файловыми системами, не входящими в стандартную поставку.

24.2. Файловые системы Linux®

FreeBSD предоставляет встроенную поддержку нескольких файловых систем Linux®. В этом разделе показано, как загрузить поддержку и подключить поддерживаемые файловые системы Linux®.

24.2.1. Расширенная файловая система (EXT)

Поддержка файловых систем Extended File System (EXT) на уровне ядра доступна в FreeBSD начиная с версии 2.2. Драйвер ext2fs(4) позволяет ядру FreeBSD читать и записывать данные в файловые системы ext2, ext3 и ext4.

Журналирование и шифрование пока не поддерживаются.

Для доступа к файловой системе ext смонтируйте раздел ext, указав его имя раздела в FreeBSD и существующую точку монтирования. В этом примере монтируется /dev/ada1s1 в /mnt:

```
# mount -t ext2fs /dev/ada1s1 /mnt
```

24.3. Файловые системы Windows®

FreeBSD поддерживает файловые системы FAT, exFAT и NTFS, обеспечивая доступ к хранилищам, отформатированным в Windows.

24.3.1. Файловая система FAT

Файловая система FAT — это простая и надёжная файловая система. Хотя она уступает современным аналогам в производительности, надёжности и масштабируемости, её доступность во многих операционных системах делает её распространённым выбором для обмена данными между устройствами.

Для доступа к файловой системе FAT необходимо подключить том FAT, указав имя раздела FreeBSD и существующую точку монтирования. В этом примере подключается /dev/ada0s1 в /mnt:

```
# mount -t msdosfs /dev/ada0s1 /mnt
```

24.3.2. Файловая система exFAT

exFAT (Extended File Allocation Table) — это облегчённая файловая система, оптимизированная для флеш-накопителей, таких как USB-диски и SD-карты. Она поддерживает файлы большого размера и широко используется на различных платформах, что делает её идеальной для внешних накопителей.

Для использования exFAT в FreeBSD установите пакет filesystems/exfat, загрузите модуль ядра FUSE и смонтируйте файловую систему, как показано ниже:

Установите пакет exFAT:

```
# pkg install fusefs-exfat
```

Прежде чем использовать файловую систему FUSE, загрузите модуль ядра fusefs(4):

```
# kldload fusefs
```

Используйте sysrc(8) для загрузки модуля при запуске:

```
# sysrc kld_list+=fusefs
```

Смонтируйте том exFAT, указав его имя раздела FreeBSD и существующую точку монтирования. В этом примере монтируется /dev/ada0s1 в /mnt:

```
# mount.exfat /dev/ada0s1 /mnt
```

24.3.3. Файловая система NTFS

NTFS — это надёжная файловая система, разработанная Microsoft® и широко используемая в операционных системах Windows. FreeBSD обеспечивает полную поддержку чтения и записи NTFS через пакет filesystems/ntfs, что упрощает доступ и изменение хранилищ данных с форматированием NTFS.

Для использования NTFS в FreeBSD установите пакет filesystems/ntfs, загрузите модуль ядра FUSE и смонтируйте файловую систему, как показано ниже:

Установите пакет NTFS:

```
# pkg install fusefs-ntfs
```

Прежде чем использовать файловую систему FUSE, загрузите модуль ядра fusefs(4):

```
# kldload fusefs
```

Используйте sysrc(8) для загрузки модуля при запуске:

```
# sysrc kld_list+=fusefs
```

Смонтируйте том NTFS, указав его имя раздела в FreeBSD и существующую точку монтирования. В этом примере монтируется /dev/ada0s1 в /mnt:

```
# ntfs-3g /dev/ada0s1 /mnt
```

24.4. Файловые системы MacOS®

FreeBSD обеспечивает поддержку файловых систем MacOS®, включая HFS/HFS+, что позволяет получать доступ к устройствам хранения, отформатированным для систем Apple®.

24.4.1. Файловая система HFS/HFS+

HFS/HFS+ была основной файловой системой для MacOS до APFS, часто использовалась на старых устройствах Mac и внешних накопителях. FreeBSD предоставляет поддержку только для чтения HFS/HFS+ через пакет filesystems/hfsfuse.

Для использования HFS/HFS+ в FreeBSD установите пакет filesystems/hfsfuse, загрузите модуль ядра FUSE и смонтируйте файловую систему, как показано ниже:

Установите пакет HFS/HFS+:

```
# pkg install fusefs-hfsfuse
```

Прежде чем использовать файловую систему FUSE, загрузите модуль ядра fusefs(4):

```
# kldload fusefs
```

Используйте sysrc(8) для загрузки модуля при запуске:

```
# sysrc kld_list+=fusefs
```

Смонтируйте том HFS/HFS+, указав его имя раздела FreeBSD и существующую точку монтирования. В этом примере монтируется /dev/ada0s1 в /mnt:

```
# hfsfuse /dev/ada0s1 /mnt
```

Глава 25. Виртуализация

25.1. Обзор

Программное обеспечение виртуализации позволяет нескольким операционным системам работать одновременно на одном компьютере. Такие программные системы для ПК часто включают основную операционную систему, которая запускает ПО виртуализации и поддерживает произвольное количество гостевых операционных систем.

Прочтите эту главу, чтобы узнать:

- Разница между основной операционной системой и гостевой операционной системой.
- Как установить FreeBSD на следующие платформы виртуализации:
  - Parallels Desktop(Apple® macOS®)
  - VMware Fusion(Apple® macOS®)
  - VirtualBox™(Microsoft® Windows®, Apple® macOS® на базе Intel®, Linux)
  - QEMU(FreeBSD)
  - bhyve(FreeBSD)
- Как настроить систему FreeBSD для достижения наилучшей производительности при виртуализации.

Прежде чем читать эту главу, необходимо:

- Понимать основы UNIX® и FreeBSD.
- Знать, как установить FreeBSD.
- Уметь настраивать сетевое соединение, как описано в разделе по расширенной настройке сети.
- Знать, как установить дополнительное стороннее программное обеспечение.

25.2. FreeBSD в качестве гостевой системы в Parallels Desktop для macOS®

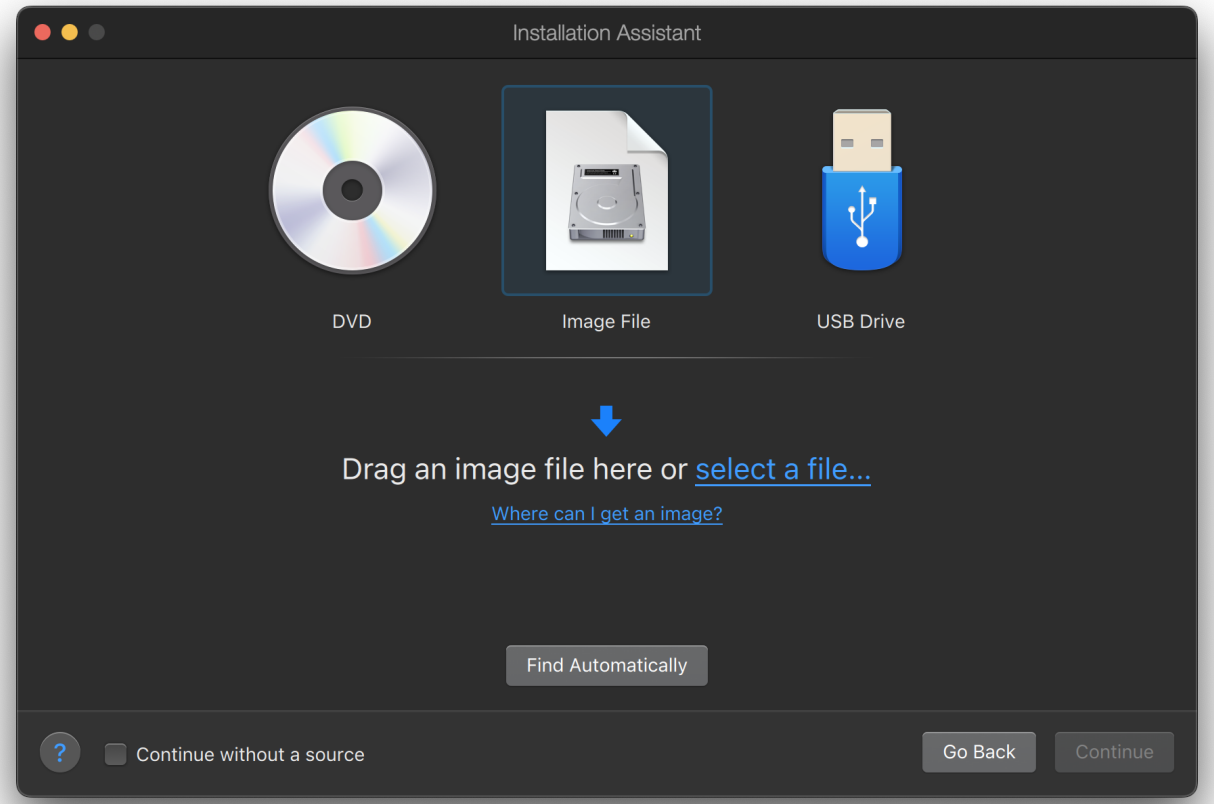
Parallels Desktop for Mac® - это коммерческий программный продукт, доступный для компьютеров Apple® Mac® под управлением macOS® 10.14.6 или новее. FreeBSD является полностью поддерживаемой гостевой операционной системой. После установки Parallels на macOS® пользователь должен настроить виртуальную машину и затем установить желаемую гостевую операционную систему.

25.2.1. Установка FreeBSD на Parallels Desktop на Mac®

Первым шагом при установке FreeBSD на Parallels является создание новой виртуальной машины для установки FreeBSD.

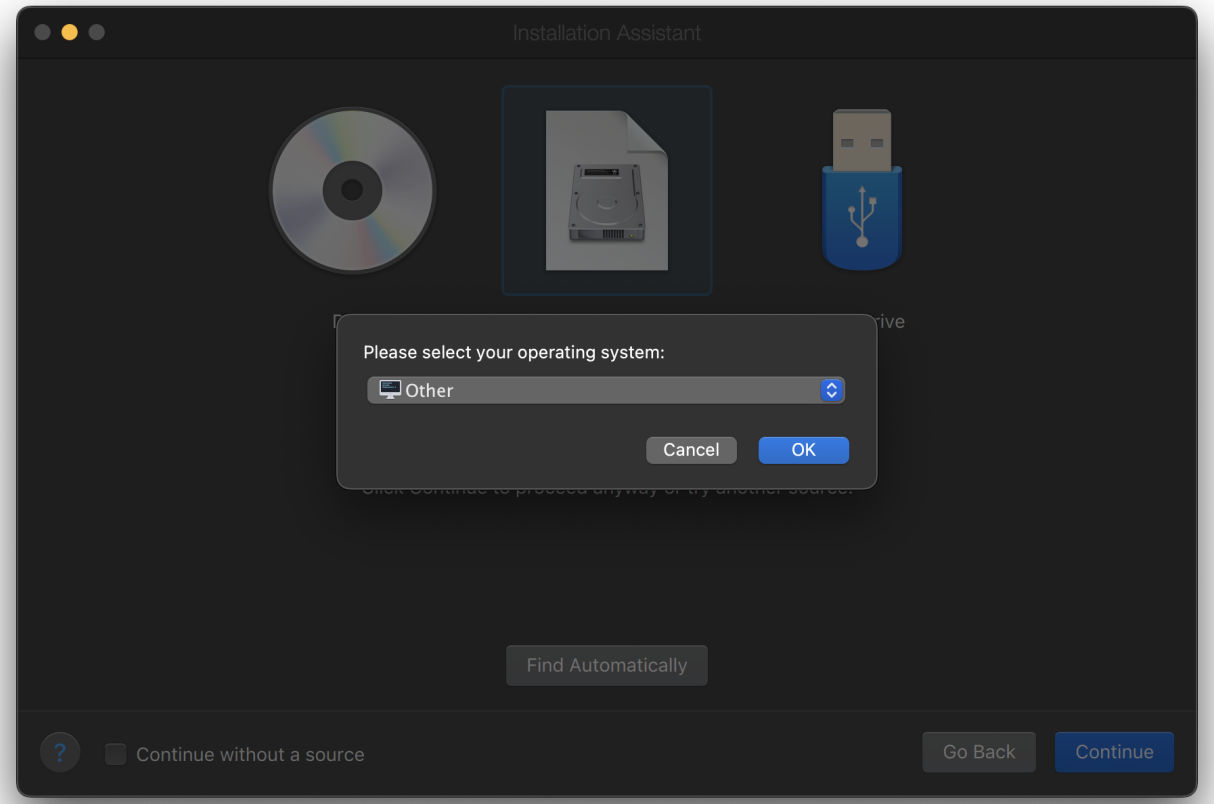
Выберите **Установка Windows или другой ОС с DVD или файла образа** и продолжите.

Выберите файл образа FreeBSD.

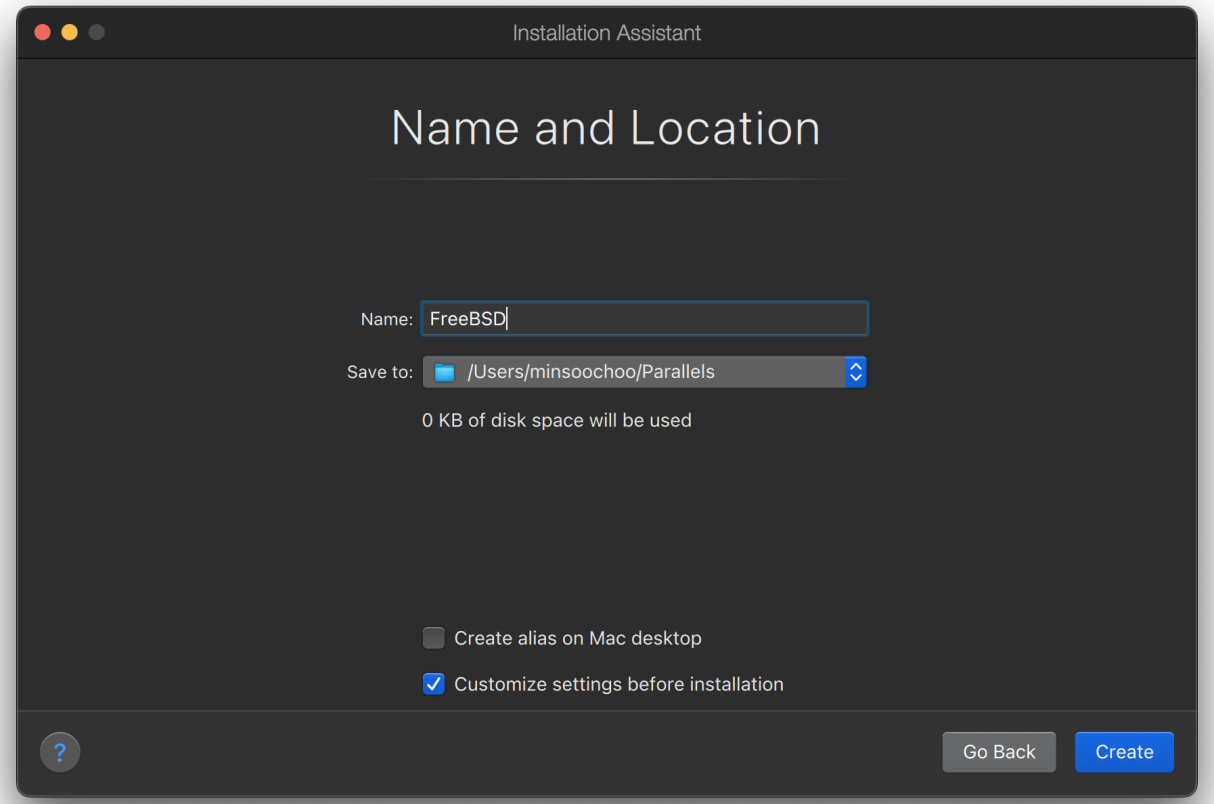


Выберите **Другое в качестве операционной системы**.

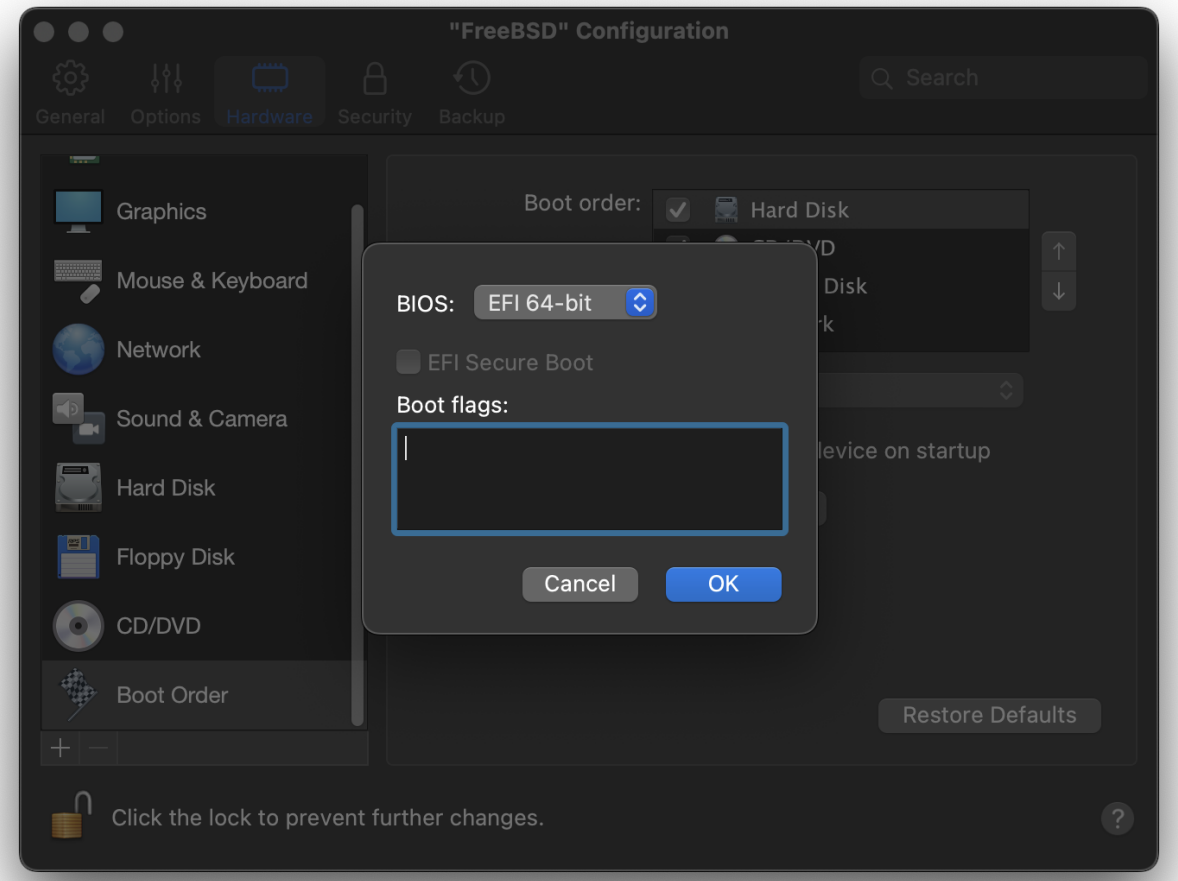
Выбор FreeBSD приведет к ошибке загрузки при запуске.



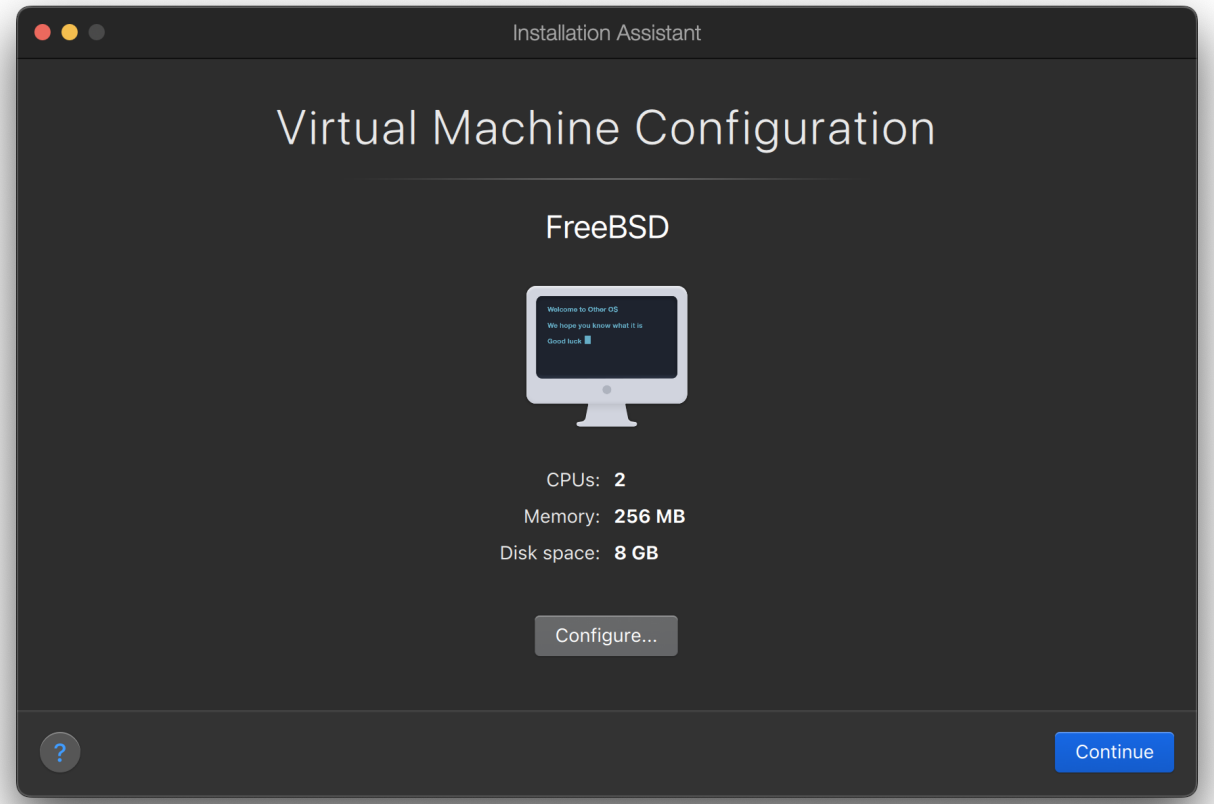
Назовите виртуальную машину и отметьте пункт **Настроить параметры перед установкой (Customize settings before installation)**



Когда появится окно конфигурации, перейдите на вкладку **Hardware**, выберите **Boot order** и нажмите **Advanced**. Затем выберите **EFI 64-bit** в качестве **BIOS**.



Нажмите **OK**, закройте окно конфигурации и нажмите **Continue**.



Виртуальная машина автоматически загрузится. Установите FreeBSD, следуя общим шагам.

25.2.2. Настройка FreeBSD на Parallels

После успешной установки FreeBSD на macOS® X с использованием Parallels, можно выполнить ряд шагов по настройке для оптимизации системы в виртуальной среде.

1. Установка переменных загрузчика

   Самый важный шаг — уменьшить параметр `kern.hz`, чтобы снизить использование CPU FreeBSD в среде Parallels. Это достигается добавлением следующей строки в /boot/loader.conf:

   ```
   kern.hz=100
   ```

   Без этой настройки бездействующая гостевая система FreeBSD в Parallels будет использовать около 15% процессора однопроцессорного iMac®. После внесения изменений использование снизится до примерно 5%.

   Если установлена FreeBSD 14.0 или более поздняя версия, а использование ЦП по-прежнему высокое, добавьте следующую строку в /boot/loader.conf:

```
debug.acpi.disabled="ged"
```

2. Создание нового файла конфигурации ядра

   Все драйверы SCSI, FireWire и USB могут быть удалены из конфигурационного файла собственного ядра. Parallels предоставляет виртуальный сетевой адаптер, используемый драйвером ed(4), поэтому все сетевые устройства, кроме ed(4) и miibus(4), могут быть удалены из ядра.

3. Настройка сети

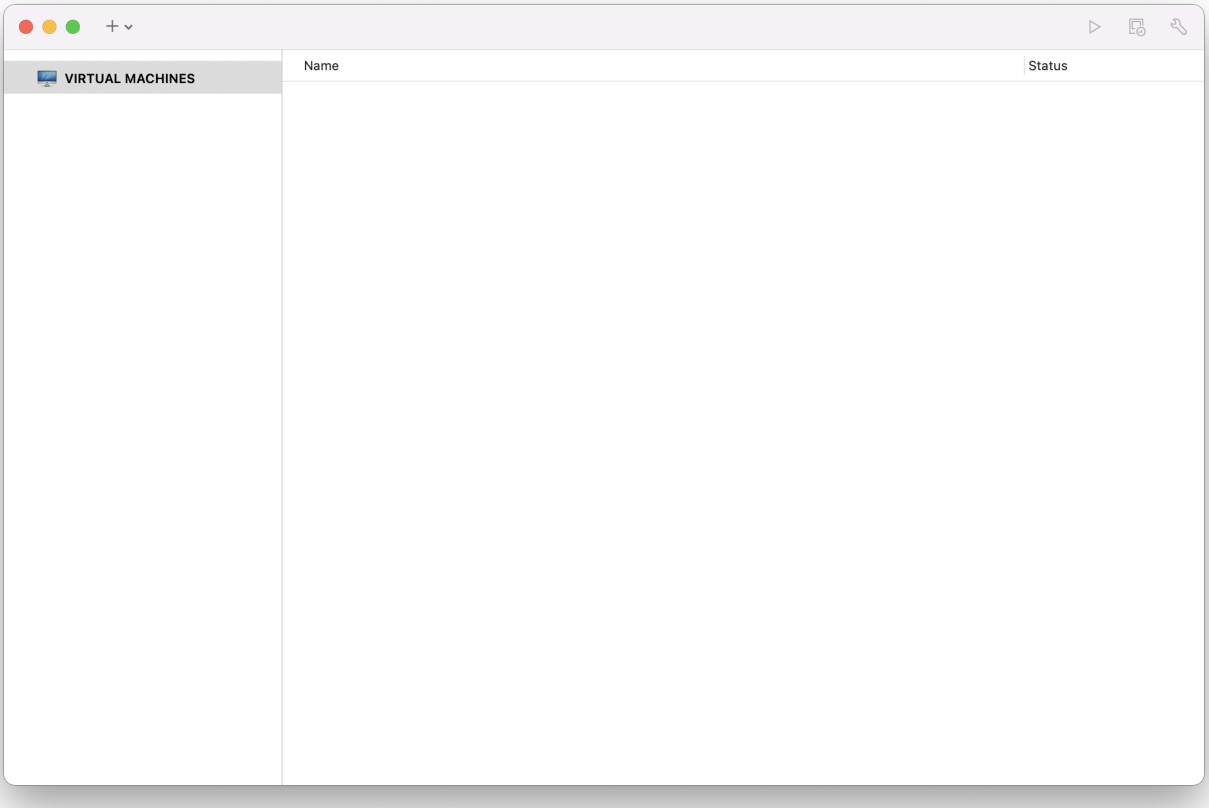
   Самая базовая настройка сети использует DHCP для подключения виртуальной машины к той же локальной сети, что и хост-компьютер Mac®. Это можно сделать, добавив `ifconfig_ed0="DHCP"` в /etc/rc.conf. Более сложные настройки сети описаны в Расширенная настройка сети.

25.3. FreeBSD в качестве гостевой системы на VMware Fusion для macOS®

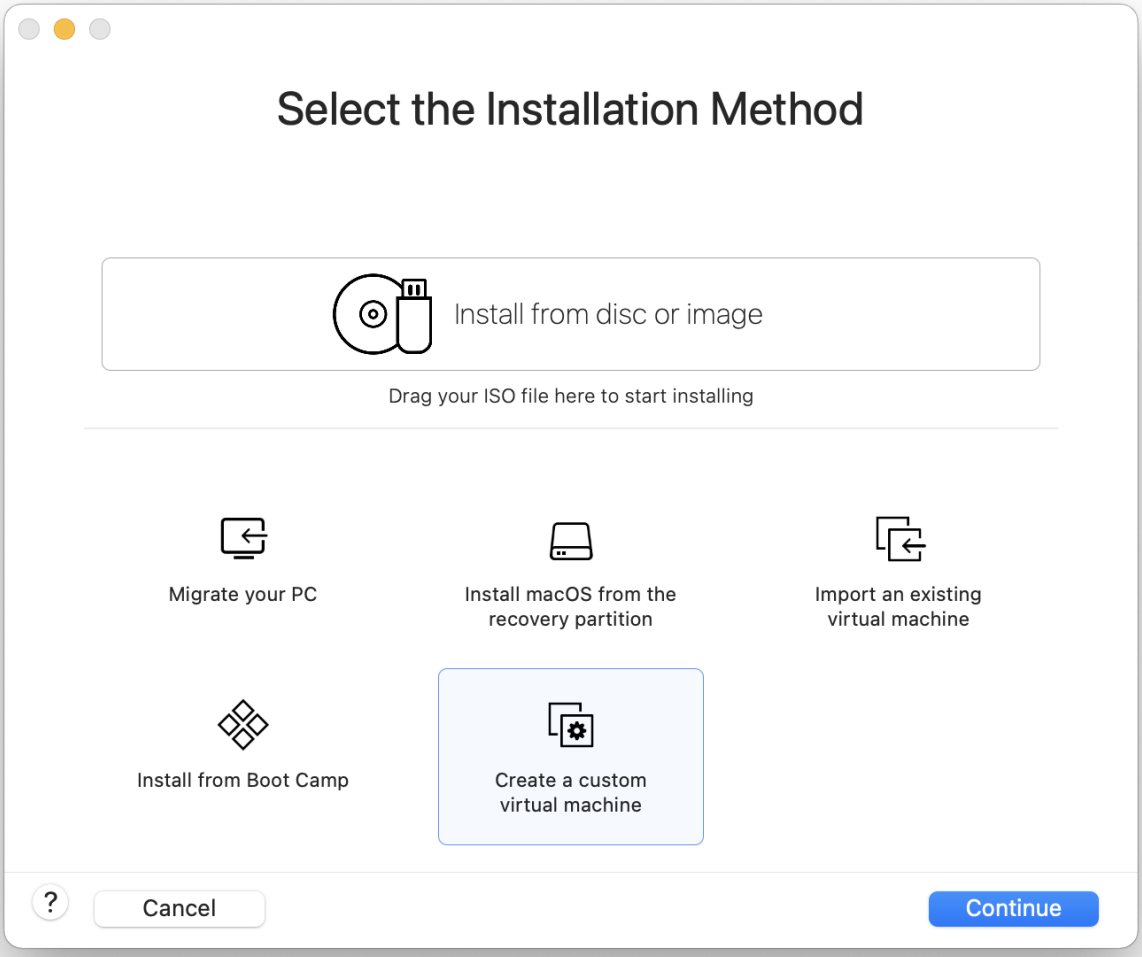
VMware Fusion для Mac® — это коммерческий программный продукт, доступный для компьютеров Apple® Mac® под управлением macOS® 12 или новее. FreeBSD является полностью поддерживаемой гостевой операционной системой. После установки VMware Fusion на macOS® пользователь может настроить виртуальную машину и установить желаемую гостевую операционную систему.

25.3.1. Установка FreeBSD на VMware Fusion

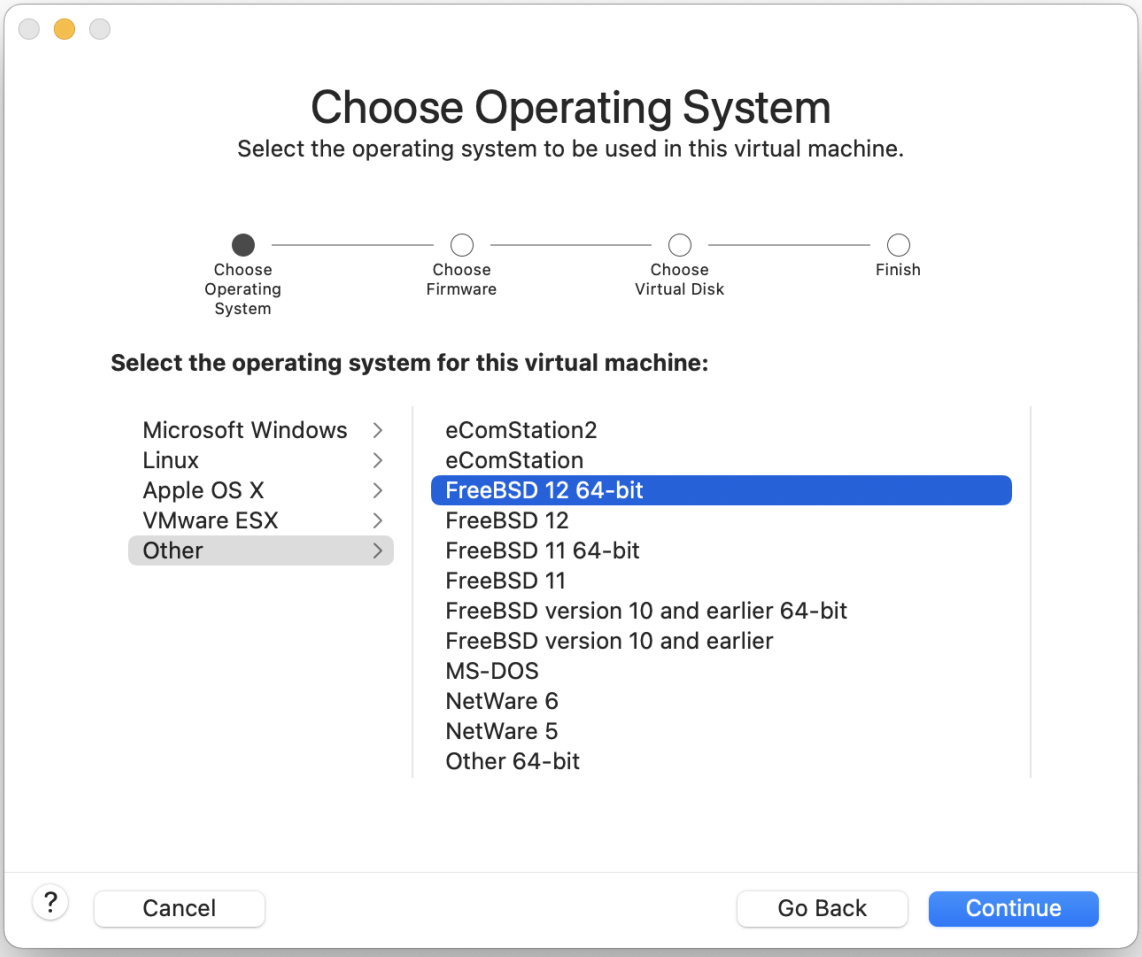
Первым шагом является запуск VMware Fusion, который загрузит библиотеку виртуальных машин. Нажмите + → Создать, чтобы создать виртуальную машину:



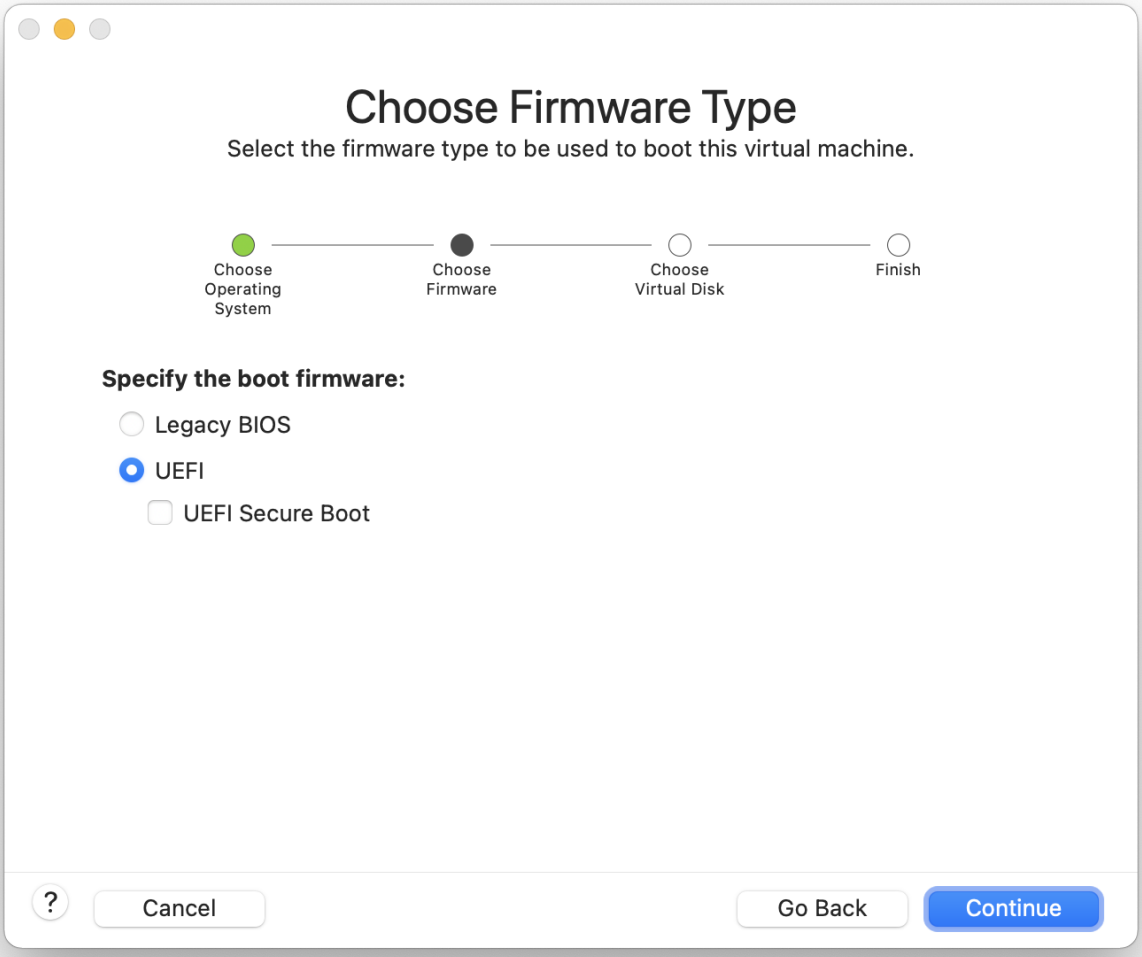
Это запустит Мастер создания новой виртуальной машины. Выберите Создать пользовательскую виртуальную машину и нажмите Продолжить, чтобы продолжить:



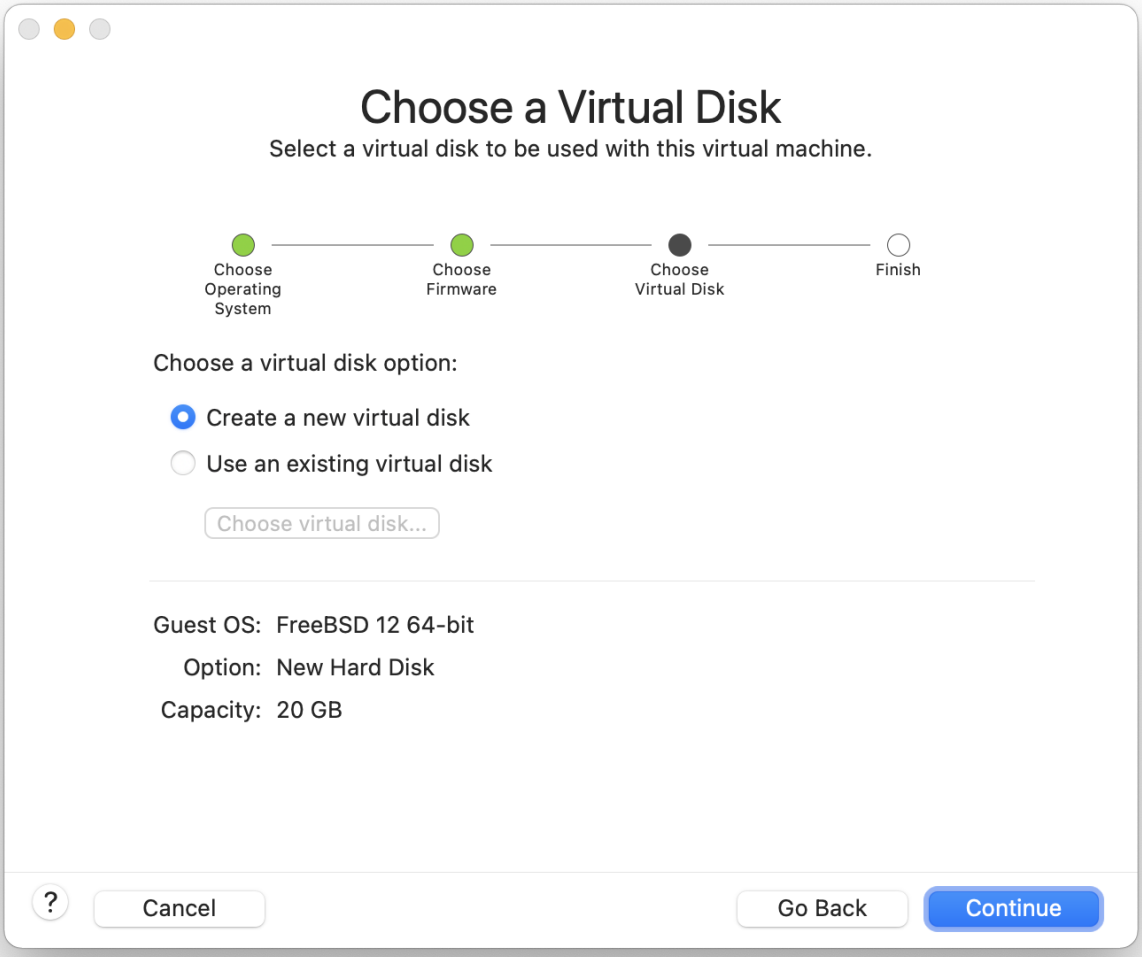
Выберите Другое в качестве Операционной системы и либо FreeBSD X, либо FreeBSD X 64-bit в качестве **Версия** при запросе:



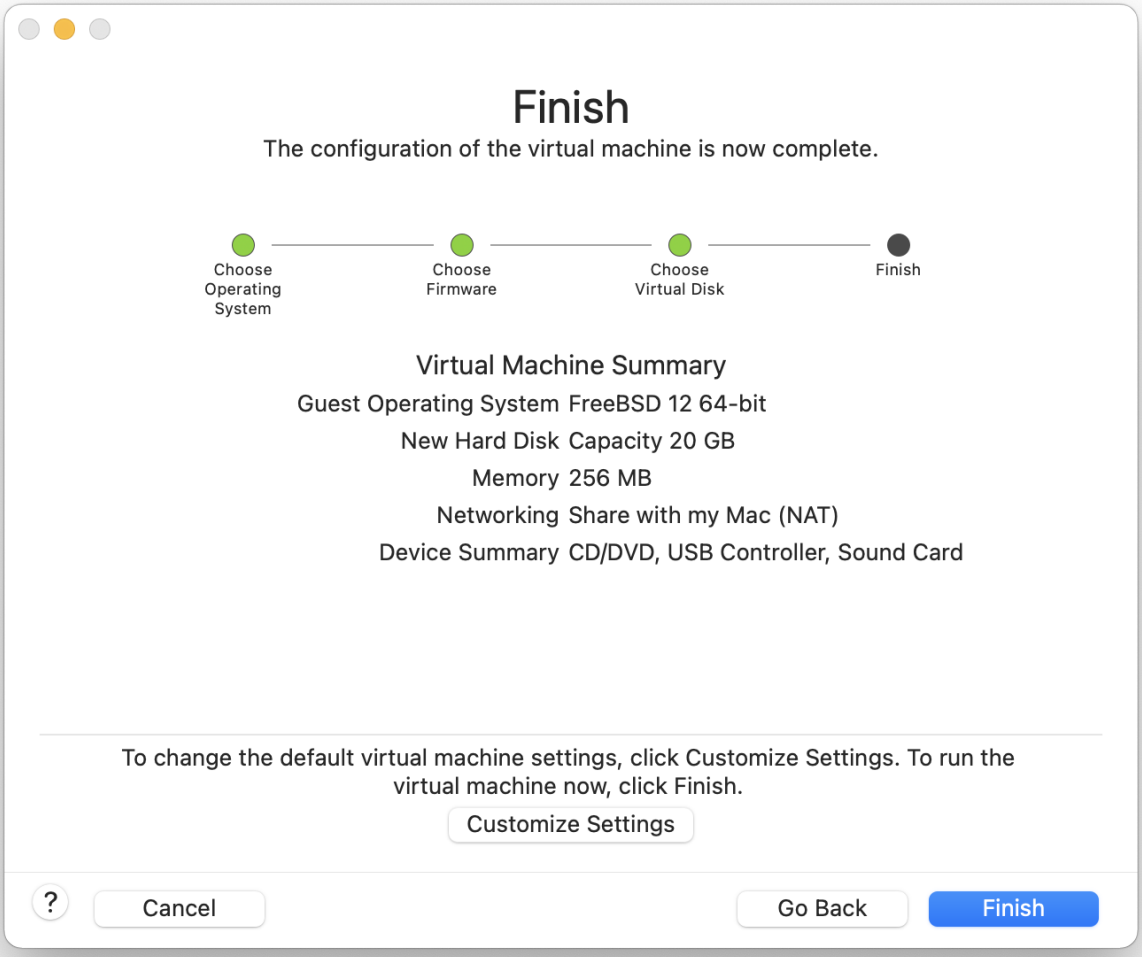
Выберите прошивку (рекомендуется UEFI):



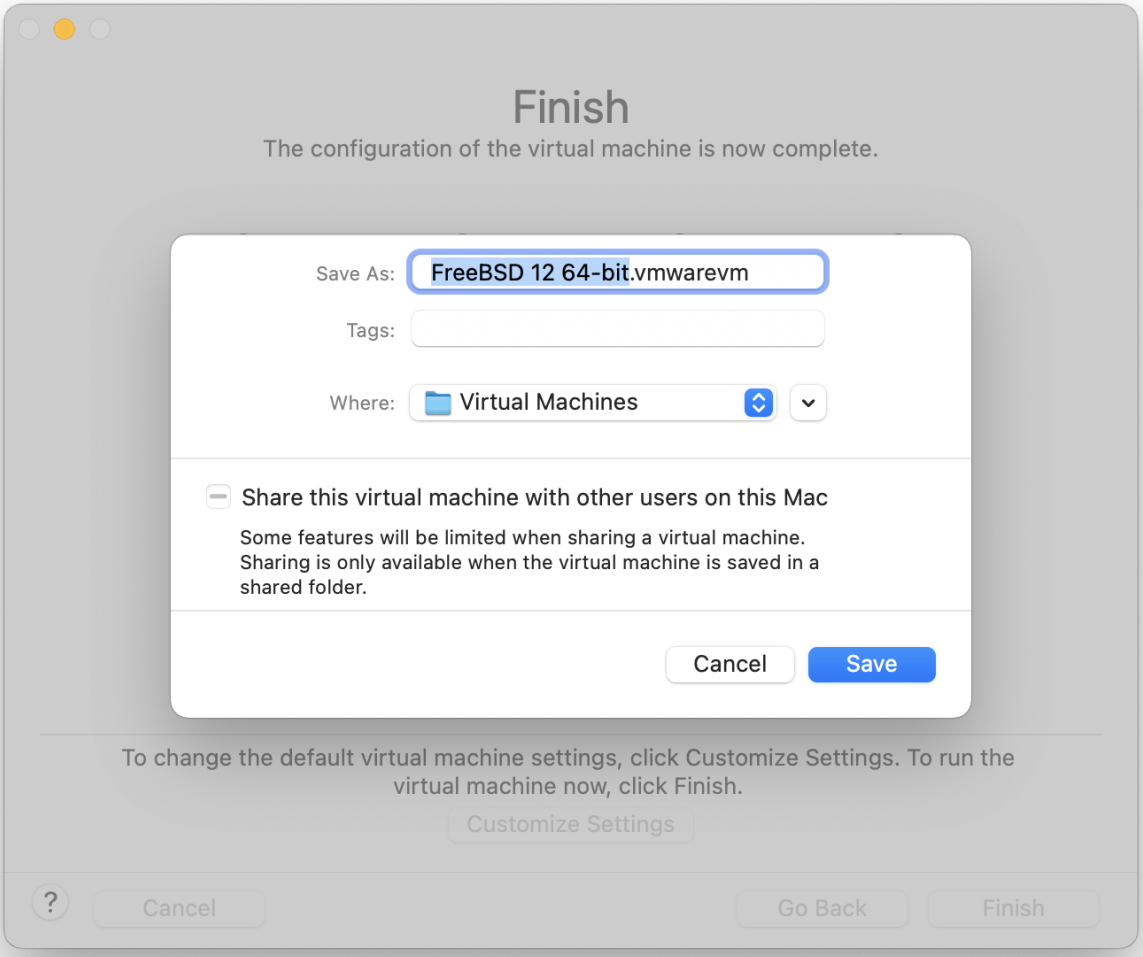
Выберите Создать новый виртуальный диск и нажмите Продолжить:



Проверьте конфигурацию и нажмите Завершить:

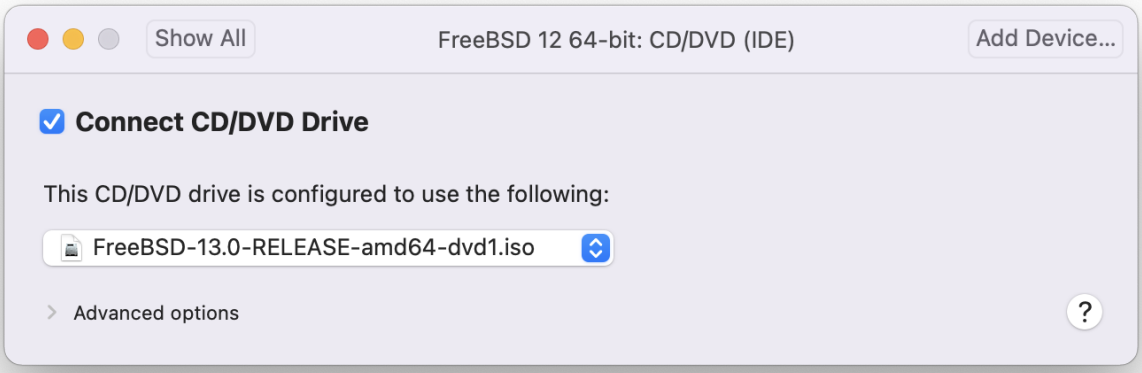


Выберите имя виртуальной машины и каталог, в котором она будет сохранена:



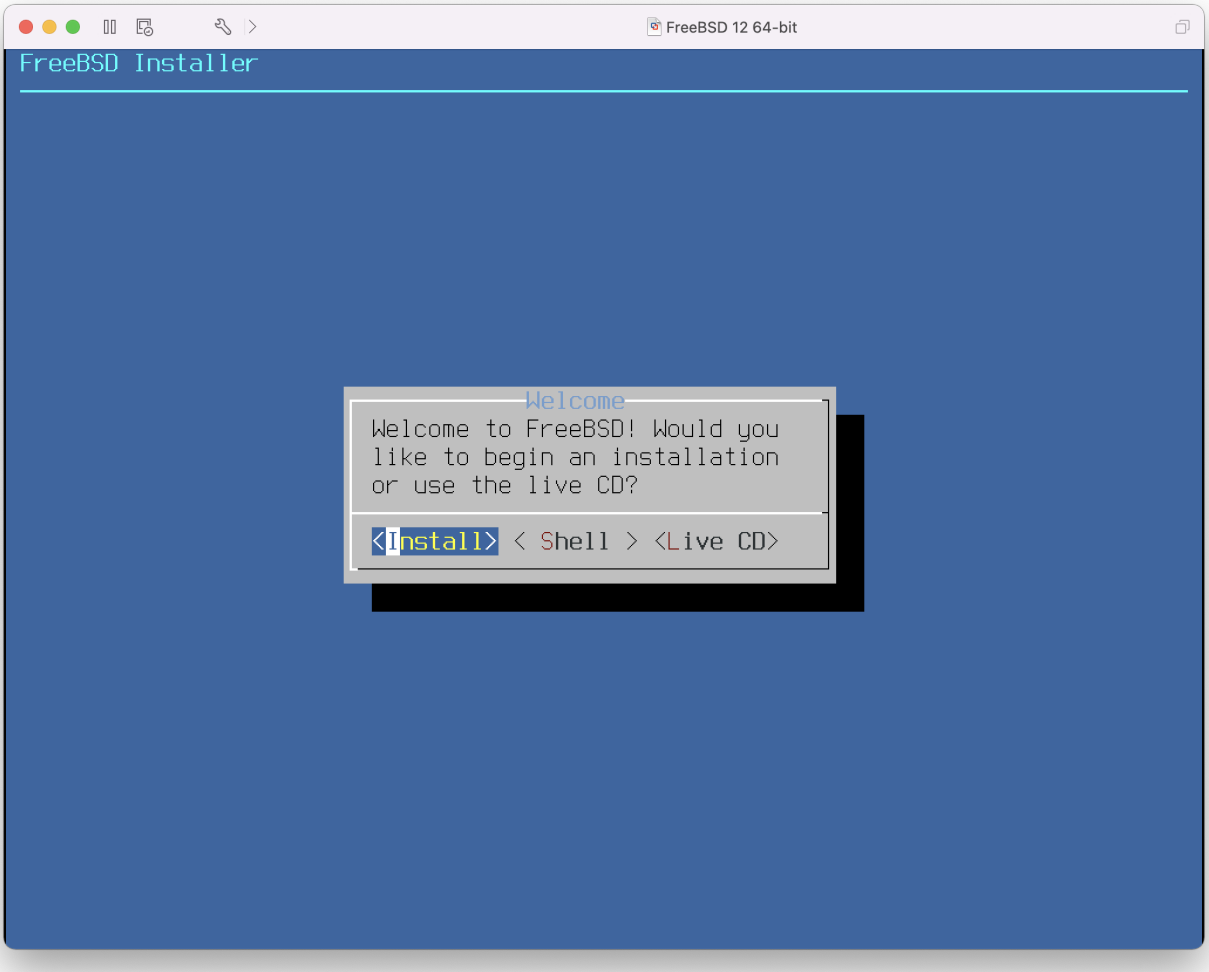
Нажмите command+E, чтобы открыть настройки виртуальной машины, и выберите CD/DVD:

Выберите образ FreeBSD в формате ISO или с CD/DVD:



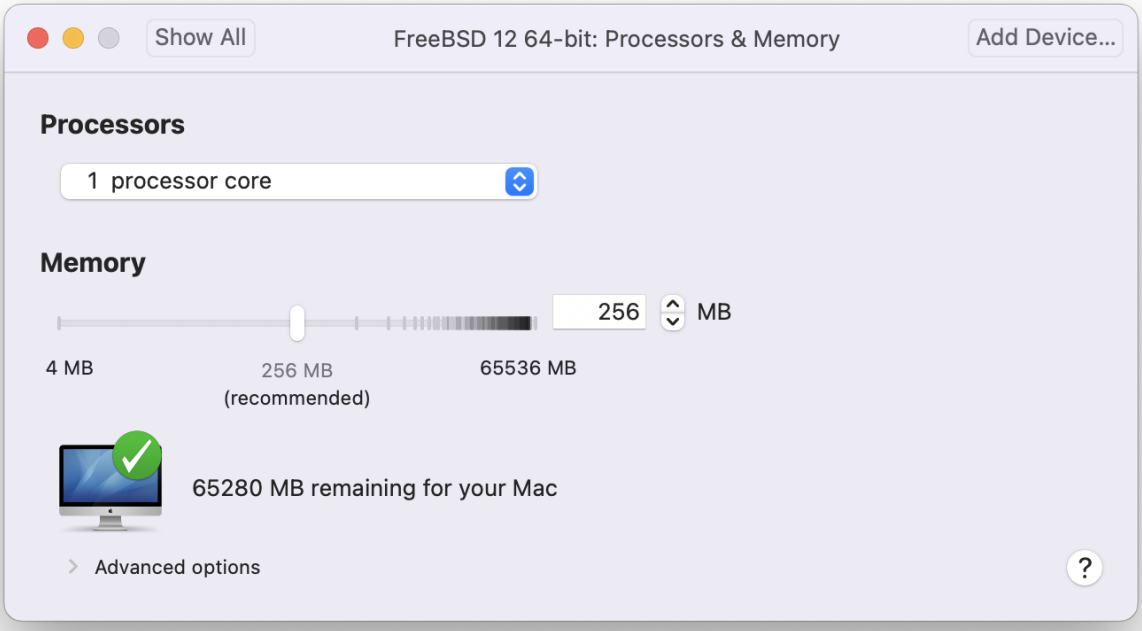
Запустите виртуальную машину:

Установите FreeBSD как обычно:

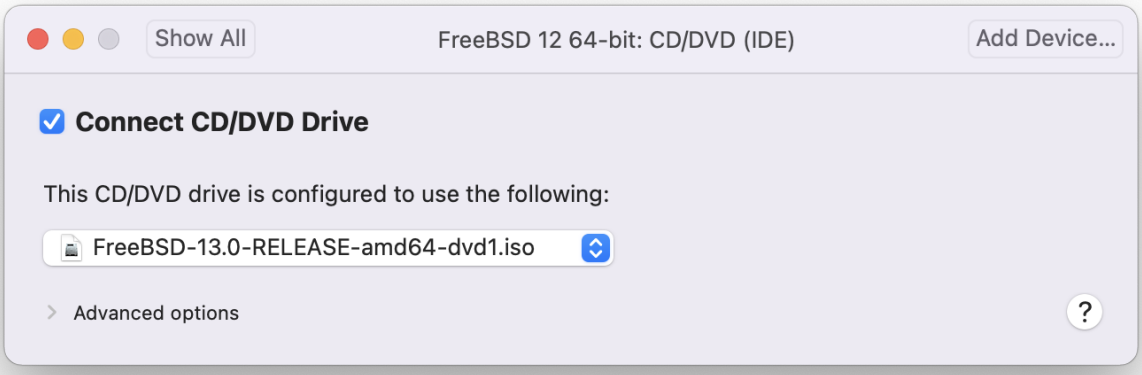


После завершения установки можно изменить параметры виртуальной машины, такие как использование памяти и количество процессоров, доступных виртуальной машине:

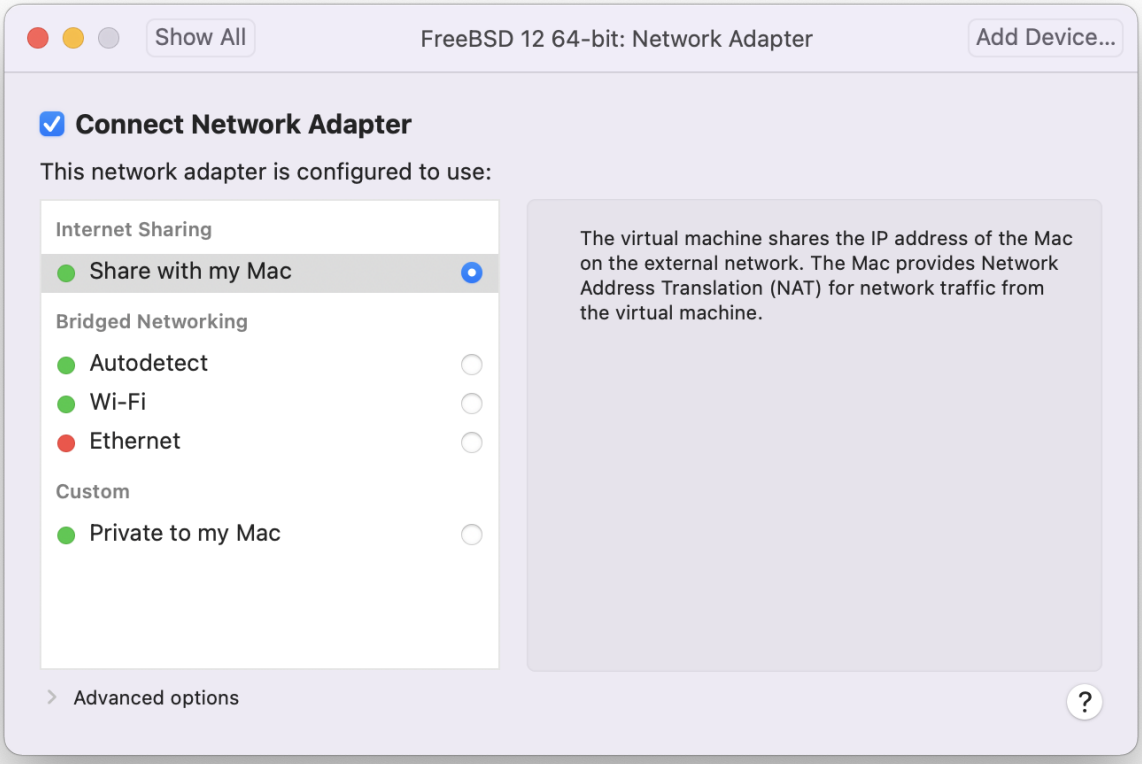
Настройки аппаратного обеспечения системы виртуальной машины нельзя изменить во время её работы.



Состояние устройства CD-ROM. Обычно CD/DVD/ISO отключается от виртуальной машины, когда в нём больше нет необходимости.



Последнее, что нужно изменить, — это способ подключения виртуальной машины к сети. Чтобы разрешить подключения к виртуальной машине с других машин, кроме хоста, выберите Подключиться напрямую к физической сети (Bridged). В противном случае предпочтительнее выбрать Использовать интернет-подключение хоста (NAT), чтобы виртуальная машина имела доступ в Интернет, но сеть не могла получить доступ к виртуальной машине.



После изменения настроек загрузите новую виртуальную машину с установленной FreeBSD.

25.3.2. Настройка FreeBSD на VMware Fusion

После успешной установки FreeBSD на macOS® X с использованием VMware Fusion можно выполнить ряд шагов по настройке для оптимизации системы в виртуальной среде.

1. Установка переменных загрузчика

   Самый важный шаг — уменьшить параметр `kern.hz`, чтобы снизить использование CPU FreeBSD в среде VMware Fusion. Это достигается добавлением следующей строки в /boot/loader.conf:

   ```
   kern.hz=100
   ```

   Без этой настройки бездействующая гостевая система FreeBSD в VMware Fusion будет использовать около 15% процессора однопроцессорного iMac®. После внесения изменений использование снизится до примерно 5%.

2. Создание нового файла конфигурации ядра

   Все драйверы устройств FireWire и USB могут быть удалены из файла конфигурации собственного ядра. VMware Fusion предоставляет виртуальный сетевой адаптер, используемый драйвером em(4), поэтому все сетевые устройства, кроме em(4), могут

быть удалены из ядра.

3. Настройка сети

   Самая базовая настройка сети использует DHCP для подключения виртуальной машины к той же локальной сети, что и хост-компьютер Mac®. Это можно сделать, добавив `ifconfig_em0="DHCP"` в /etc/rc.conf. Более сложные настройки сети описаны в Сложные вопросы работы в сети.

4. Установите драйверы и open-vm-tools

   Для бесперебойной работы FreeBSD на VMWare необходимо установить драйверы:

   ```
   # pkg install xf86-video-vmware xf86-input-vmmouse open-vm-tools
   ```

   Пакеты xf86 доступны только для гостевых систем FreeBSD на архитектуре x86.

Включение мыши

+ Некоторые пользователи сообщали о проблемах с использованием мыши в виртуальной машине. Мышь можно включить, добавив следующее в /boot/loader.conf:

+

```
# ums_load="YES"
```

25.4. FreeBSD в качестве гостевой системы в VirtualBox™

FreeBSD хорошо работает в качестве гостевой системы в VirtualBox™. Это программное обеспечение виртуализации доступно для большинства распространённых операционных систем, включая саму FreeBSD.

Дополнения гостевой ОС VirtualBox™ обеспечивают поддержку:

- Общий буфер обмена.
- Интеграция указателя мыши.
- Синхронизация времени хоста.
- Масштабирование окна.
- Бесшовный режим.

Для запуска Xorg убедитесь, что в настройках раздела `Display` конфигурации виртуальной машины для параметра `Screen` установлено значение **VBoxSVGA** — и сделайте это **до**

установки Guest Additions. Не выбирайте VMSVGA, так как этот драйвер предназначен для гостевых систем VMware и не работает в FreeBSD. Его использование приводит к некорректной работе Xorg. VBoxSVGA лишён этой проблемы и обеспечивает корректный запуск X11 без ошибок.

Эти команды выполняются в гостевой системе FreeBSD.

Установите пакет или порт emulators/virtualbox-ose-additions` в гостевой системе FreeBSD. Это установит порт:

```
# cd /usr/ports/emulators/virtualbox-ose-additions && make install clean
```

Добавьте следующие строки в файл /etc/rc.conf:

```
vboxguest_enable="YES"
vboxservice_enable="YES"
```

Если используется ntpd(8) или ntpdate(8), отключите синхронизацию времени хоста:

```
vboxservice_flags="--disable-timesync"
```

Xorg автоматически распознает драйвер `vboxvideo`. Его также можно вручную добавить в /etc/X11/xorg.conf:

```
Section "Device"
    Identifier "Card0"
    Driver "vboxvideo"
    VendorName "InnoTek Systemberatung GmbH"
    BoardName "VirtualBox Graphics Adapter"
EndSection
```

Чтобы использовать драйвер `vboxmouse`, измените раздел мыши в /etc/X11/xorg.conf:

```
Section "InputDevice"
    Identifier "Mouse0"
    Driver "vboxmouse"
EndSection
```

Общие папки для передачи файлов между хостом и виртуальной машиной доступны путем их монтирования с помощью `mount_vboxvfs`. Общую папку можно создать на хосте через графический интерфейс VirtualBox или с помощью `vboxmanage`. Например, чтобы создать общую папку с именем *myshare* в /mnt/bsdboxshare для виртуальной машины с именем *BSDBox*, выполните:

```
# vboxmanage sharedfolder add 'BSDBox' --name myshare --hostpath /mnt/bsdboxshare
```

Обратите внимание, что имя общей папки не должно содержать пробелов. Подключите общую папку из гостевой системы следующим образом:

```
# mount_vboxvfs -w myshare /mnt
```

25.5. FreeBSD в качестве хоста с VirtualBox™

VirtualBox™ — это активно развивающийся комплексный пакет виртуализации, доступный для большинства операционных систем, включая Windows®, macOS®, Linux® и FreeBSD. Он одинаково хорошо подходит для запуска гостевых систем Windows® или UNIX®-подобных ОС. Программа распространяется как открытое программное обеспечение, но с проприетарными компонентами, доступными в отдельном пакете расширений. Эти компоненты включают поддержку устройств USB 2.0. Дополнительную информацию можно найти на странице загрузок вики VirtualBox™. В настоящее время эти расширения недоступны для FreeBSD.

25.5.1. Установка VirtualBox™

VirtualBox™ доступен в FreeBSD как пакет или порт в emulators/virtualbox-ose. Порт можно установить с помощью следующих команд:

```
# cd /usr/ports/emulators/virtualbox-ose
# make install clean
```

Одна полезная опция в меню настройки порта — набор программ `GuestAdditions`. Они предоставляют ряд полезных функций в гостевых операционных системах, таких как интеграция указателя мыши (позволяя использовать мышь совместно между хостом и гостем без необходимости нажатия специальной комбинации клавиш для переключения) и ускоренный рендеринг видео, особенно в гостевых системах Windows®. Гостевые дополнения доступны в меню **Devices** после завершения установки гостевой системы.

Для начала работы с VirtualBox™ необходимо внести несколько изменений в конфигурацию. Порт устанавливает модуль ядра в /boot/modules, который должен быть загружен в работающее ядро:

```
# kldload vboxdrv
```

Чтобы модуль всегда загружался после перезагрузки, добавьте эту строку в /boot/loader.conf:

```
vboxdrv_load="YES"
```

Для использования модулей ядра, которые позволяют организовать мостовую или гостевую сеть, добавьте следующую строку в /etc/rc.conf и перезагрузите компьютер:

```
vboxnet_enable="YES"
```

Группа `vboxusers` создаётся во время установки VirtualBox™. Все пользователи, которым необходим доступ к VirtualBox™, должны быть добавлены в эту группу. Для добавления новых участников можно использовать `pw`:

```
# pw groupmod vboxusers -m username
```

Настройки прав доступа по умолчанию для /dev/vboxnetctl являются ограничительными и требуют изменения для работы мостового соединения:

```
# chown root:vboxusers /dev/vboxnetctl
# chmod 0660 /dev/vboxnetctl
```

Чтобы сделать это изменение прав постоянным, добавьте следующие строки в /etc/devfs.conf:

```
own     vboxnetctl root:vboxusers
perm    vboxnetctl 0660
```

Для запуска VirtualBox™ введите в сеансе Xorg:

```
% VirtualBox
```

Для получения дополнительной информации о настройке и использовании VirtualBox™ обратитесь к официальному сайту. Информация, относящаяся к FreeBSD, и инструкции по устранению проблем доступны на соответствующей странице wiki FreeBSD.

25.5.2. Поддержка USB в VirtualBox™

В VirtualBox™ можно настроить передачу USB-устройств в гостевую операционную систему. Хост-контроллер версии OSE ограничен эмуляцией USB 1.1 устройств до тех пор, пока расширение, поддерживающее USB 2.0 и 3.0 устройства, не станет доступным в FreeBSD.

Чтобы VirtualBox™ мог обнаруживать USB-устройства, подключённые к машине, пользователь должен быть членом группы `operator`.

```
# pw groupmod operator -m username
```

Затем добавьте следующее в /etc/devfs.rules или создайте этот файл, если он ещё не

существует:

```
[system=10]
add path 'usb/*' mode 0660 group operator
```

Чтобы загрузить эти новые правила, добавьте следующее в /etc/rc.conf:

```
devfs_system_ruleset="system"
```

Затем перезапустите devfs:

```
# service devfs restart
```

Перезапустите сеанс входа и VirtualBox™, чтобы изменения вступили в силу, и при необходимости создайте фильтры USB.

25.5.3. Виртуальный DVD/CD-доступ хоста VirtualBox™

Доступ к DVD/CD-приводам хоста из гостевых систем обеспечивается путем совместного использования физических приводов. В VirtualBox™ это настраивается в окне Хранилище (Storage) в настройках (Settings) виртуальной машины. При необходимости сначала создайте пустое устройство IDECD/DVD. Затем выберите Привод хоста (Host drive) из всплывающего меню для выбора виртуального привода CD/DVD. Появится флажок с меткой Сквозной доступ (Passthrough). Это позволяет виртуальной машине использовать оборудование напрямую. Например, аудио-CD или записывающее устройство будут работать только при выборе этой опции.

Для того чтобы пользователи могли использовать функции VirtualBox™ для работы с DVD/CD, им необходим доступ к /dev/xpt0, /dev/cdN и /dev/passN. Обычно это достигается путем добавления пользователя в группу operator. Права доступа к этим устройствам можно настроить, добавив следующие строки в /etc/devfs.conf:

```
perm cd* 0660
perm xpt0 0660
perm pass* 0660
```

```
# service devfs restart
```

25.6. Виртуализация с QEMU на FreeBSD

QEMU — это универсальный эмулятор и виртуализатор машин, представляющий собой полностью открытое программное обеспечение. Он разрабатывается большим активным сообществом и поддерживает FreeBSD, OpenBSD, NetBSD, а также другие операционные

системы.

Документация QEMU:

- QEMU можно использовать несколькими способами. Наиболее распространённый — это эмуляция системы, при которой предоставляется виртуальная модель всей машины (процессор, память и эмулируемые устройства) для запуска гостевой ОС. В этом режиме процессор может быть полностью эмулирован или работать с гипервизором, таким как `KVM`, `Xen` или `Hypervisor.Framework`, что позволяет гостевой системе выполняться непосредственно на процессоре хоста.
- Второй поддерживаемый способ использования QEMU — это эмуляция пользовательского режима, при которой QEMU может запускать процессы, скомпилированные для одного ЦП, на другом ЦП. В этом режиме ЦП всегда эмулируется.
- QEMU также предоставляет ряд автономных утилит командной строки, таких как утилита для работы с образами дисков qemu-img(1), которая позволяет создавать, преобразовывать и изменять образы дисков.

QEMU может эмулировать широкий спектр архитектур, включая `Arm™`, `i386`, `x86_64`, `MIPS™`, `s390X`, `SPARC™` (Sparc™ и Sparc64™), и другие. Список целевых систем QEMU System Emulator регулярно обновляется.

Этот раздел описывает, как использовать QEMU для системной эмуляции и эмуляции пользовательского режима в FreeBSD, а также содержит примеры использования команд QEMU и утилит командной строки.

25.6.1. Установка программы QEMU

QEMU доступен в виде пакета FreeBSD или порта в emulators/qemu. Сборка пакета включает разумные опции и настройки по умолчанию для большинства пользователей и является рекомендуемым способом установки.

```
# pkg install qemu
```

Установка пакета включает несколько зависимостей. После завершения установки создайте ссылку на основную версию QEMU для хоста, которая будет использоваться чаще всего. Если хост — 64-битная система Intel™ или AMD™, это будет:

```
# ln -s /usr/local/bin/qemu-system-x86_64 /usr/local/bin/qemu
```

Протестируйте установку, выполнив следующую команду от имени непривилегированного пользователя:

```
% qemu
```

Открывается окно, в котором QEMU активно пытается загрузиться с жёсткого диска,

дискеты, DVD/CD и PXE. Пока ничего не настроено, поэтому команда выдаст несколько ошибок и завершится сообщением «No bootable device» (Нет загрузочного устройства), как показано на рисунке 1. Тем не менее, это подтверждает, что ПО QEMU установлено корректно.

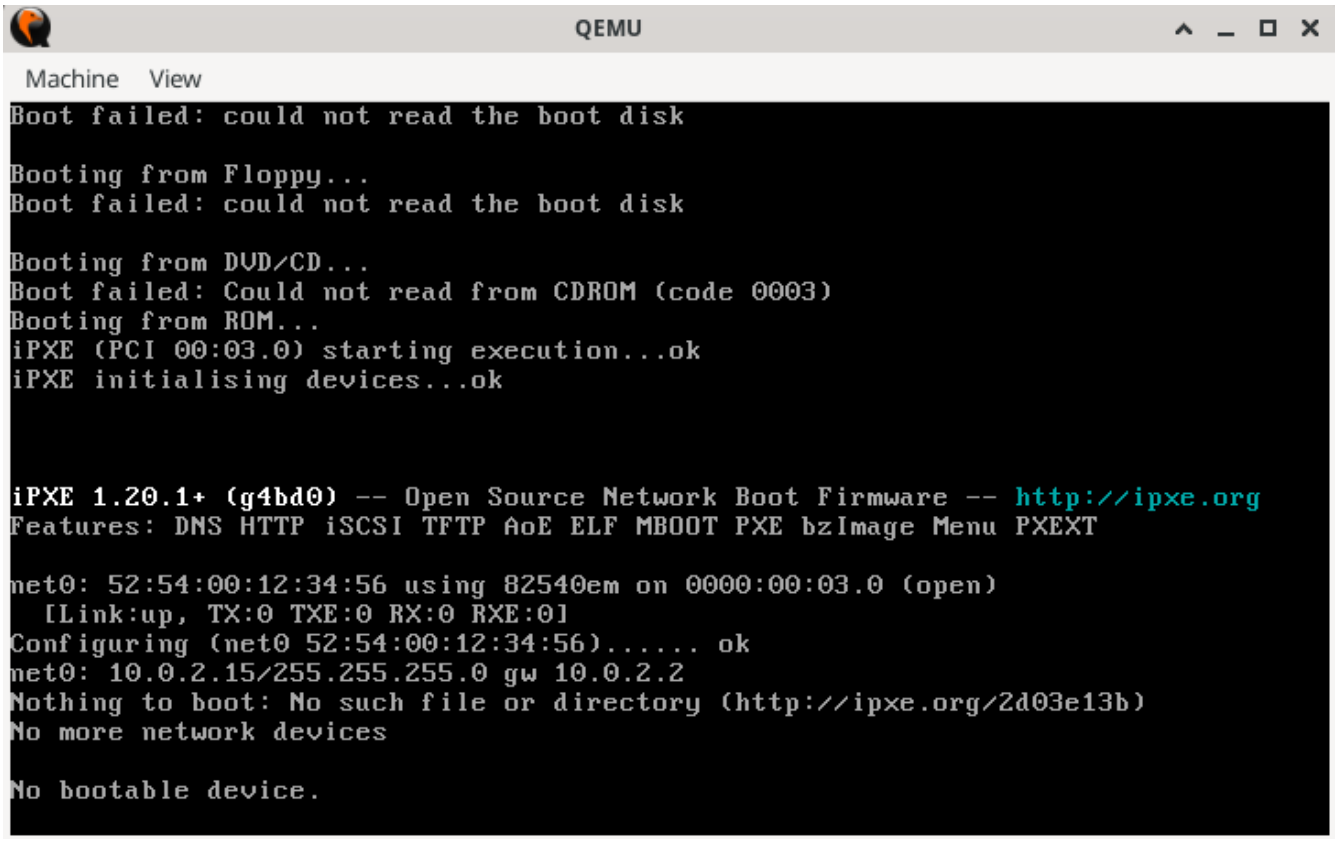


Рисунок 61. QEMU без загрузочного образа

25.6.2. Установка виртуальной машины

QEMU находится в стадии активной разработки. Возможности и параметры команд могут меняться от одной версии к другой. В этом разделе приведены примеры, разработанные с использованием QEMU версии 9.0.1 (лето 2024 года). В случае сомнений всегда обращайтесь к документации QEMU, особенно к странице О QEMU, где содержатся ссылки на поддерживаемые платформы сборки, эмуляцию, устаревшие и удалённые возможности.

Выполните следующие шаги, чтобы создать две виртуальные машины с именами `left` и `right`. Большинство команд можно выполнять без привилегий root.

1. Создайте тестовую среду для работы с QEMU:

```
% mkdir -p ~/QEMU  ~/QEMU/SCRIPTS  ~/QEMU/ISO  ~/QEMU/VM
```

   Каталог SCRIPTS предназначен для стартовых скриптов и утилит. Каталог ISO содержит загрузочные ISO-образы для гостевых систем. В каталоге VM хранятся образы виртуальных машин (`BM`).

2. Скачайте свежую копию FreeBSD в ~/QEMU/ISO:

```
% cd ~/QEMU/ISO
% fetch https://download.freebsd.org/releases/ISO-IMAGES/14.1/FreeBSD-14.1-RELEASE-amd64-bootonly.iso
```

После завершения загрузки создайте сокращенную ссылку. Эта сокращенная ссылка используется в скриптах запуска ниже.

```
% ln -s FreeBSD-14.1-RELEASE-amd64-bootonly.iso  fbsd.iso
```

3. Перейдите в каталог для виртуальных машин (~/QEMU/VM). Запустите qemu-img(1) для создания образов дисков виртуальной машины "left":

```
% cd ~/QEMU/VM
% qemu-img create -f raw  left.img   15G
```

Формат `raw` в QEMU предназначен для обеспечения высокой производительности. Этот формат прост и не имеет накладных расходов, что делает его быстрее, особенно в сценариях с высокой производительностью или высокой пропускной способностью. Он используется в случаях, когда требуется максимальная производительность, а дополнительные функции, такие как снимки состояния, не нужны. Этот формат используется в скрипте для виртуальной машины "left", приведённой ниже.

Отдельным форматом является `qcow2`, который использует технологию QEMU "копирования при записи" для управления дисковым пространством. Эта технология не требует полного диска размером 15G, а лишь заготовку, которой напрямую управляет виртуальная машина. Диск растёт динамически по мере записи данных виртуальной машиной. Этот формат поддерживает снимки состояния, сжатие и шифрование. Его применение целесообразно в разработке, тестировании и сценариях, требующих этих расширенных возможностей. Данный формат используется в скрипте для виртуальной машины "right" ниже.

Снова выполните qemu-img(1), чтобы создать образ диска для ВМ "right", используя `qcow2`:

```
% qemu-img create -f qcow2 -o preallocation=full,cluster_size=512K,lazy_refcounts=on right.qcow2 20G
```

Чтобы увидеть фактический размер файла, используйте:

```
% du -Ah right.qcow2
```

4. Настройте сеть для обеих виртуальных машин с помощью следующих команд. В этом примере сетевой интерфейс хоста — `em0`. При необходимости измените его в

соответствии с интерфейсом вашей системы. Это необходимо выполнять после каждой перезагрузки хоста, чтобы обеспечить возможность обмена данными для гостевых QEMU ВМ.

```
# ifconfig tap0 create
# ifconfig tap1 create
# sysctl net.link.tap.up_on_open=1
net.link.tap.up_on_open: 0 -> 1
# sysctl net.link.tap.user_open=1
net.link.tap.user_open: 0 -> 1
# ifconfig bridge0 create
# ifconfig bridge0 addm tap0 addm tap1 addm em0
# ifconfig bridge0 up
```

Приведенные выше команды создают два устройства tap(4) (`tap0`, `tap1`) и одно устройство if_bridge(4) (`bridge0`). Затем они добавляют устройства `tap` и интерфейс локального хоста (`em0`) в `bridge`, а также устанавливают две записи sysctl(8), чтобы разрешить обычным пользователям открывать устройство tap. Эти команды позволят виртуальным машинам взаимодействовать с сетевым стеком на хосте.

5. Перейдите в ~/QEMU/SCRIPTS и используйте следующий скрипт для запуска первой виртуальной машины — "left". Этот скрипт использует диск QEMU в формате raw.

```
/usr/local/bin/qemu-system-x86_64  -monitor none \
  -cpu qemu64 \
  -vga std \
  -m 4096 \
  -smp 4   \
  -cdrom ../ISO/fbsd.iso \
  -boot order=cd,menu=on \
  -blockdev driver=file,aio=threads,node-name=imgleft,filename=../VM/left.img \
  -blockdev driver=raw,node-name=drive0,file=imgleft \
  -device virtio-blk-pci,drive=drive0,bootindex=1  \
  -netdev tap,id=nd0,ifname=tap0,script=no,downscript=no,br=bridge0 \
  -device e1000,netdev=nd0,mac=02:20:6c:65:66:74 \
  -name \"left\"
```

Сохраните приведённый выше код в файл (например, `left.sh`) и просто выполните: `/bin/sh left.sh`

QEMU запустит виртуальную машину в отдельном окне и загрузит FreeBSD iso, как показано на рисунке 2. Все параметры команд, такие как `-cpu` и `-boot`, полностью описаны в man-странице QEMU qemu(1).

Рисунок 62. Меню загрузчика FreeBSD

Если в окне консоли QEMU кликнуть мышкой , QEMU «захватит» мышь, как показано в Рисунок 4. Нажмите `Ctrl`+`Alt`+`G`, чтобы освободить мышь.

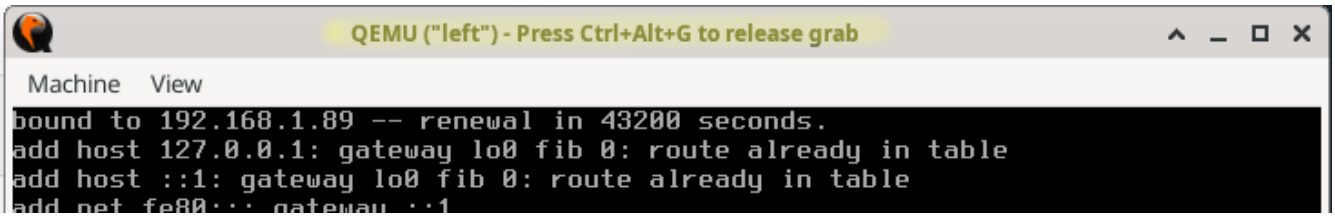


Рисунок 63. Когда QEMU захватил мышь

На FreeBSD первоначальная установка QEMU может быть несколько медленной. Это происходит потому, что эмулятор записывает форматирование файловой системы и метаданные при первом использовании диска. Последующие операции, как правило, выполняются значительно быстрее.

Во время установки следует обратить внимание на несколько моментов:

- Выберите использование UFS в качестве файловой системы. ZFS работает неэффективно при небольшом объёме памяти.
- Для использования сети настройте DHCP. При необходимости настройте IPv6, если он поддерживается локальной сетью.
- При добавлении пользователя по умолчанию убедитесь, что он является членом группы **wheel**.

После завершения установки виртуальная машина перезагружается в только что

установленный образ FreeBSD.

Войдите как `root` и обновите систему следующим образом:

```
# freebsd-update fetch
# freebsd-update install
# shutdown -r now
```

После успешной установки QEMU загрузит операционную систему, установленную на диске, а не программу установки.

QEMU поддерживает параметр `-runas`. Для повышения безопасности добавьте параметр "-runas user_name" в приведённый выше скрипт. Подробности см. в qemu(1).

Войдите снова как `root` и добавьте любые необходимые пакеты. Чтобы использовать систему X Window в гостевой системе, см. раздел «Использование системы X Window» ниже.

Завершена настройка виртуальной машины "left".

Для установки виртуальной машины "right" выполните следующий скрипт. Этот скрипт содержит необходимые изменения для tap1, format=qcow2, имени файла образа, MAC-адреса и названия окна терминала. При желании можно добавить параметр "-runas", как описано в примечании выше.

```
/usr/local/bin/qemu-system-x86_64  -monitor none \
  -cpu qemu64 \
  -vga cirrus \
  -m 4096  -smp 4   \
  -cdrom ../ISO/fbsd.iso \
  -boot order=cd,menu=on \
  -drive
if=none,id=drive0,cache=writeback,aio=threads,format=qcow2,discard=unmap,file=../VM/ri
ght.qcow2 \
  -device virtio-blk-pci,drive=drive0,bootindex=1  \
  -netdev tap,id=nd0,ifname=tap1,script=no,downscript=no,br=bridge0 \
  -device e1000,netdev=nd0,mac=02:72:69:67:68:74 \
  -name \"right\"
```

После завершения установки обе машины "left" и "right" могут взаимодействовать друг с другом и с хостом. Если на хосте действуют строгие правила брендмауэра, рассмотрите возможность добавления или изменения правил для разрешения взаимодействия между мостом (bridge) и устройствами tap.

25.6.3. Советы по использованию

Использование системы X Window

Установка Xorg описывает, как настроить систему X Window. Обратитесь к этому руководству для первоначальной настройки X Window, затем ознакомьтесь с Среды рабочего стола, чтобы настроить полноценную рабочую среду.

Этот раздел демонстрирует использование рабочего стола XFCE.

После завершения установки войдите в систему как обычный пользователь, затем введите:

```
% startx
```

Менеджер окон XFCE4 должен запуститься и предоставить рабочий графический рабочий стол, как на рисунке 5. При первом запуске отображение рабочего стола может занять до минуты. Подробности использования см. в документации на сайте XFCE.

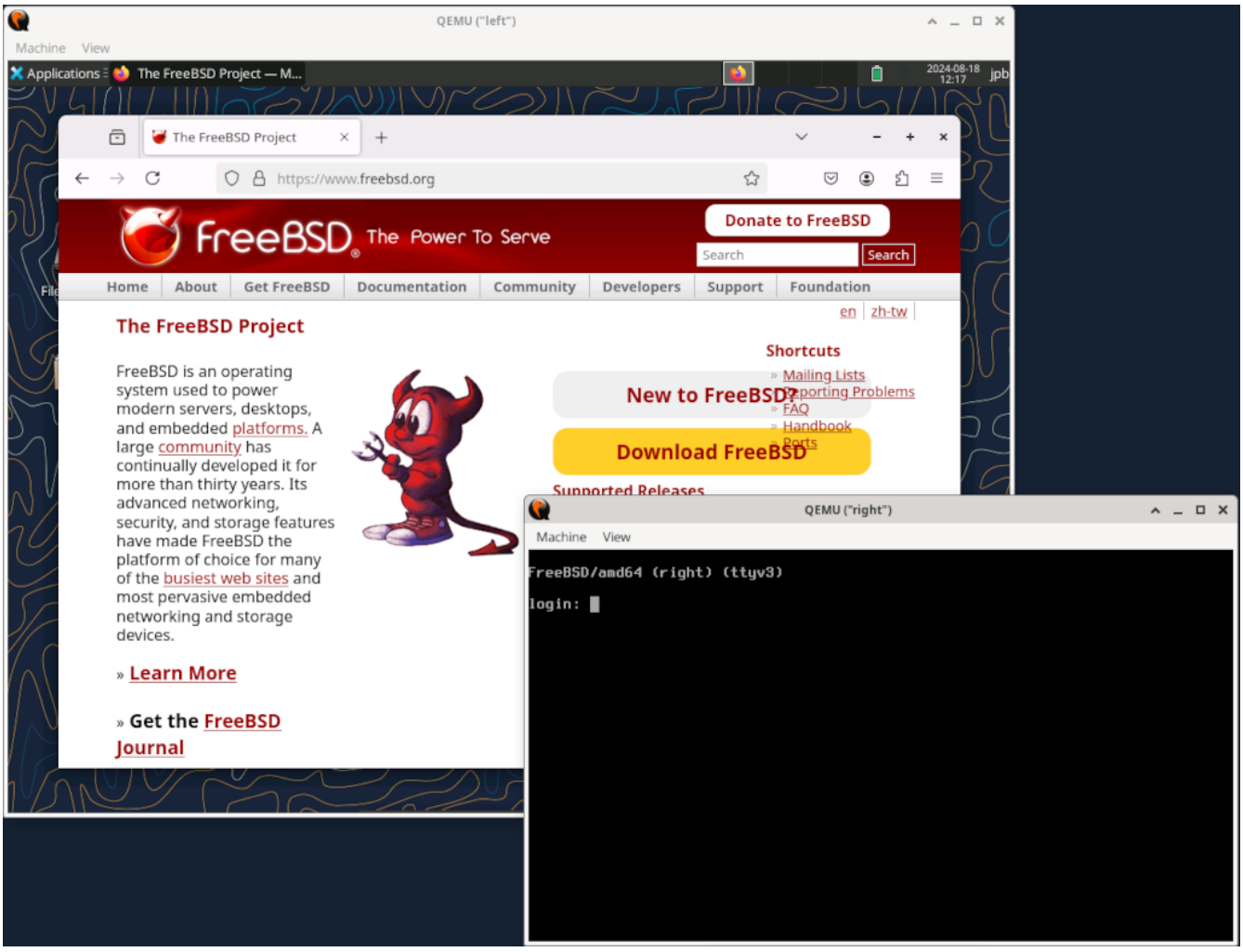


Рисунок 64. Обе виртуальные машины QEMU

Добавление дополнительной памяти в гостевую систему может ускорить работу графического интерфейса пользователя.

Здесь на виртуальной машине "left" установлена система X Window, а виртуальная машина "right" всё ещё находится в текстовом режиме.

Использование окна QEMU

Окно QEMU функционирует как полноценная консоль FreeBSD и способно работать с несколькими виртуальными терминалами, как и система на реальном оборудовании.

Для переключения на другую виртуальную консоль кликните в окно QEMU и нажмите `Alt` + `F2` или `Alt` + `F3`. FreeBSD должна переключиться на другую виртуальную консоль. Рисунок 6 показывает ВМ "left" с виртуальной консолью на `ttyv3`.

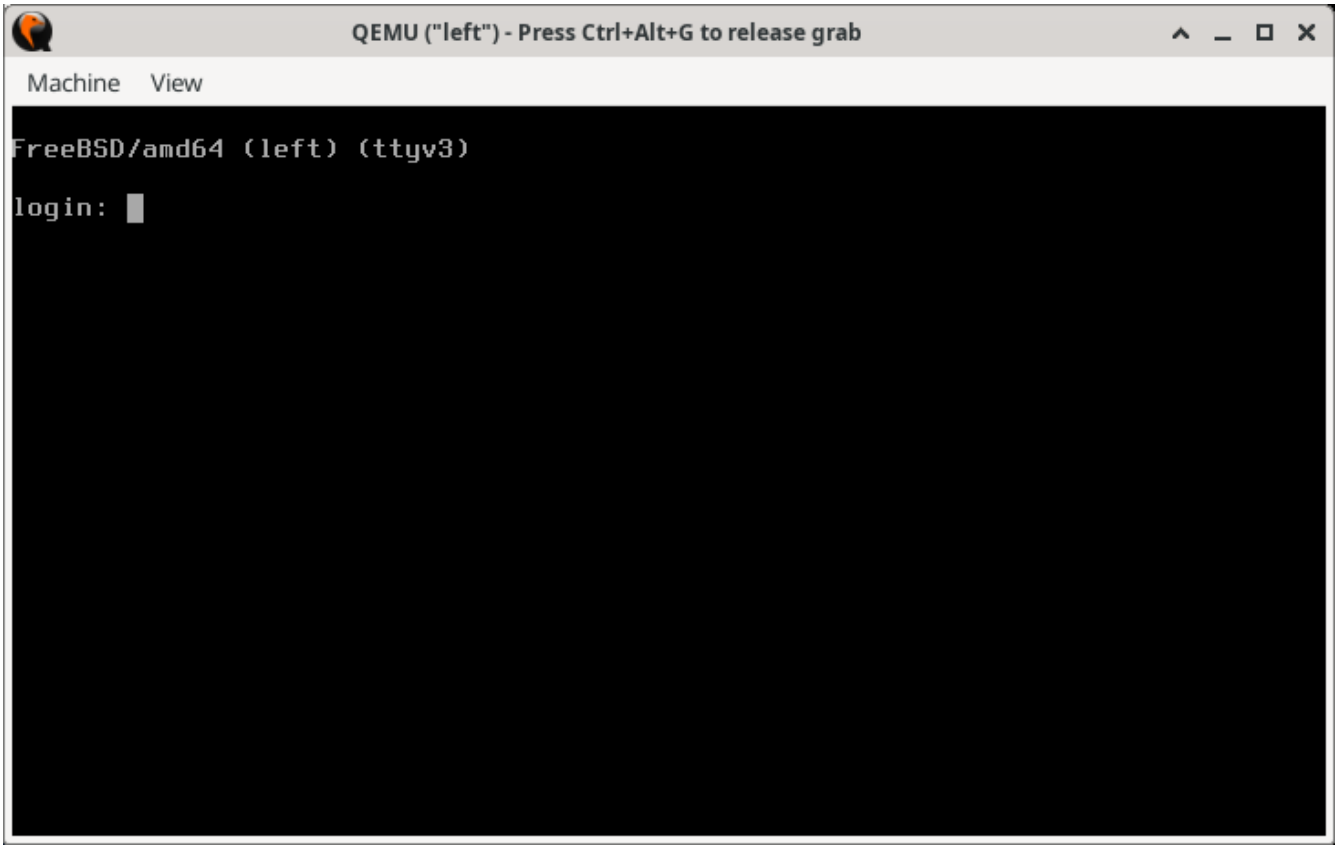


Рисунок 65. Переключение на другую виртуальную консоль в окне QEMU

Текущий менеджер рабочего стола или оконный менеджер на хосте может быть уже настроен на другое действие для комбинаций клавиш `Alt` + `F1`, `Alt` + `F2`. В таком случае попробуйте нажать `Ctrl` + `Alt` + `F1`, `Ctrl` + `Alt` + `F2` или другую подобную комбинацию. Подробности смотрите в документации вашего оконного менеджера или менеджера рабочего стола.

Использование меню окна QEMU

Еще одна особенность окна QEMU — это меню `View` и элементы управления масштабом. Наиболее полезным является пункт `Zoom to Fit`. При выборе этого пункта меню появляется возможность изменить размер окна QEMU, перетаскивая его углы. На рисунке 7 показан эффект изменения размера окна "left" в графическом режиме.

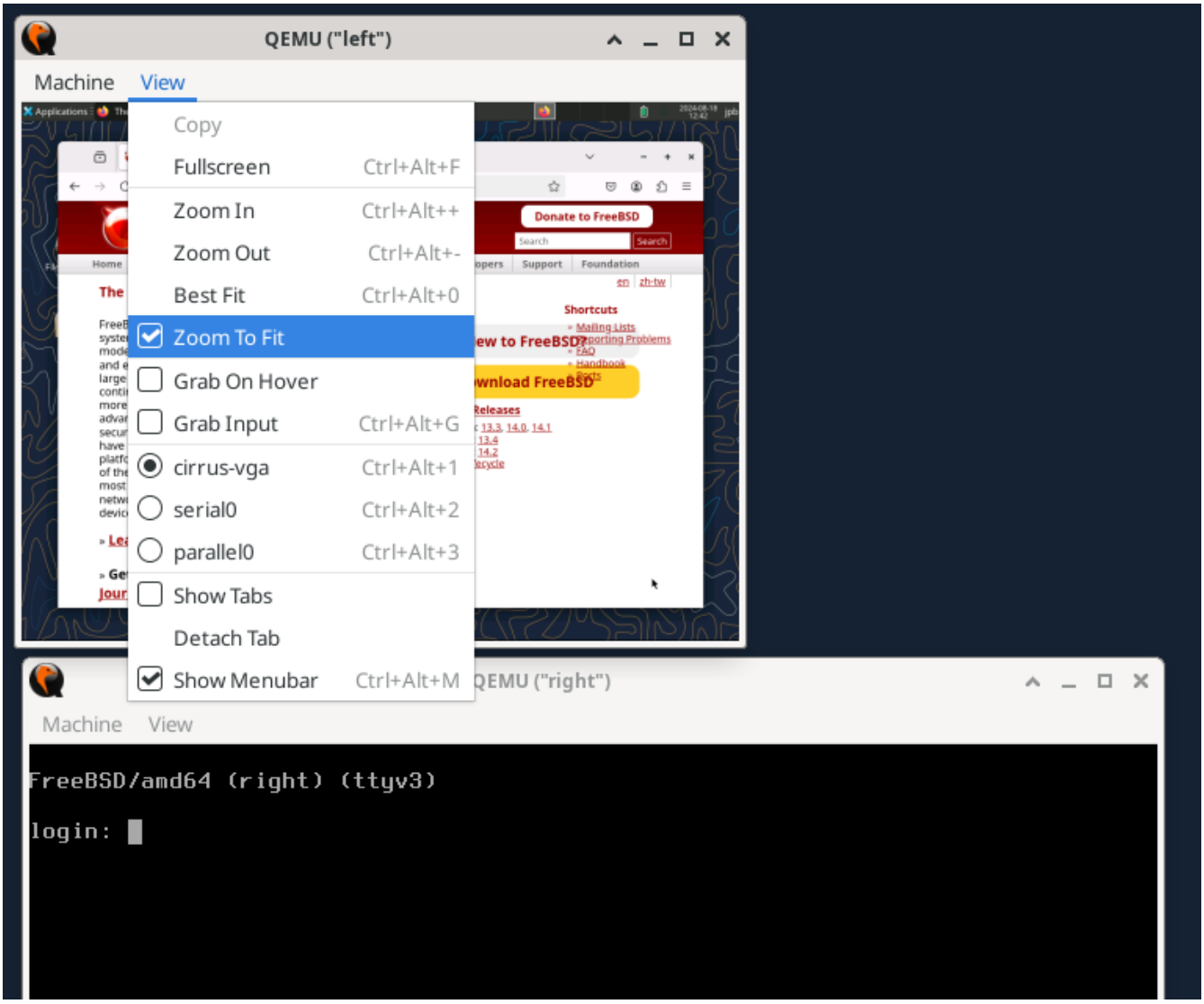


Рисунок 66. Использование опции `Zoom to Fit` в меню View

Другие пункты меню окна QEMU

Также в меню `View` отображаются

- Опции `cirrus-vga`, `serial0` и `parallel0`. Они позволяют переключать ввод/вывод на выбранное устройство.

Окно QEMU в меню `Machine` предоставляет четыре типа управления гостевой виртуальной машиной:

- `Pause` позволяет приостановить виртуальную машину QEMU. Это может быть полезно для остановки быстро прокручивающегося окна.
- `Reset` немедленно возвращает виртуальную машину в исходное состояние, как "при включении питания". Как и с реальной машиной, это не рекомендуется, если в этом нет крайней необходимости.
- `Power Down` имитирует сигнал отключения ACPI, и операционная система выполняет аккуратное завершение работы.
- `Quit` немедленно выключает виртуальную машину — также не рекомендуется, если в этом нет необходимости.

25.6.4. Добавление интерфейса последовательного порта к гостевой ВМ

Для использования последовательной консоли гостевая ВМ под управлением FreeBSD должна добавить

```
console="comconsole"
```

в /boot/loader.conf для разрешения использования последовательной консоли FreeBSD.

Обновлённая конфигурация ниже показывает, как реализовать последовательную консоль на гостевой ВМ. Запустите скрипт для старта ВМ.

```
# left+serial.sh
echo
echo "NOTE: telnet startup server running on guest VM!"
echo "To start QEMU, start another session and telnet to localhost port 4410"
echo

/usr/local/bin/qemu-system-x86_64  -monitor none \
  -serial telnet:localhost:4410,server=on,wait=on\
  -cpu qemu64 \
  -vga std \
  -m 4096 \
  -smp 4   \
  -cdrom ../ISO/fbsd.iso \
  -boot order=cd,menu=on \
  -blockdev driver=file,aio=threads,node-name=imgleft,filename=../VM/left.img \
  -blockdev driver=raw,node-name=drive0,file=imgleft \
  -device virtio-blk-pci,drive=drive0,bootindex=1  \
  -netdev tap,id=nd0,ifname=tap0,script=no,downscript=no,br=bridge0 \
  -device e1000,netdev=nd0,mac=02:20:6c:65:66:74 \
  -name \"left\"
```

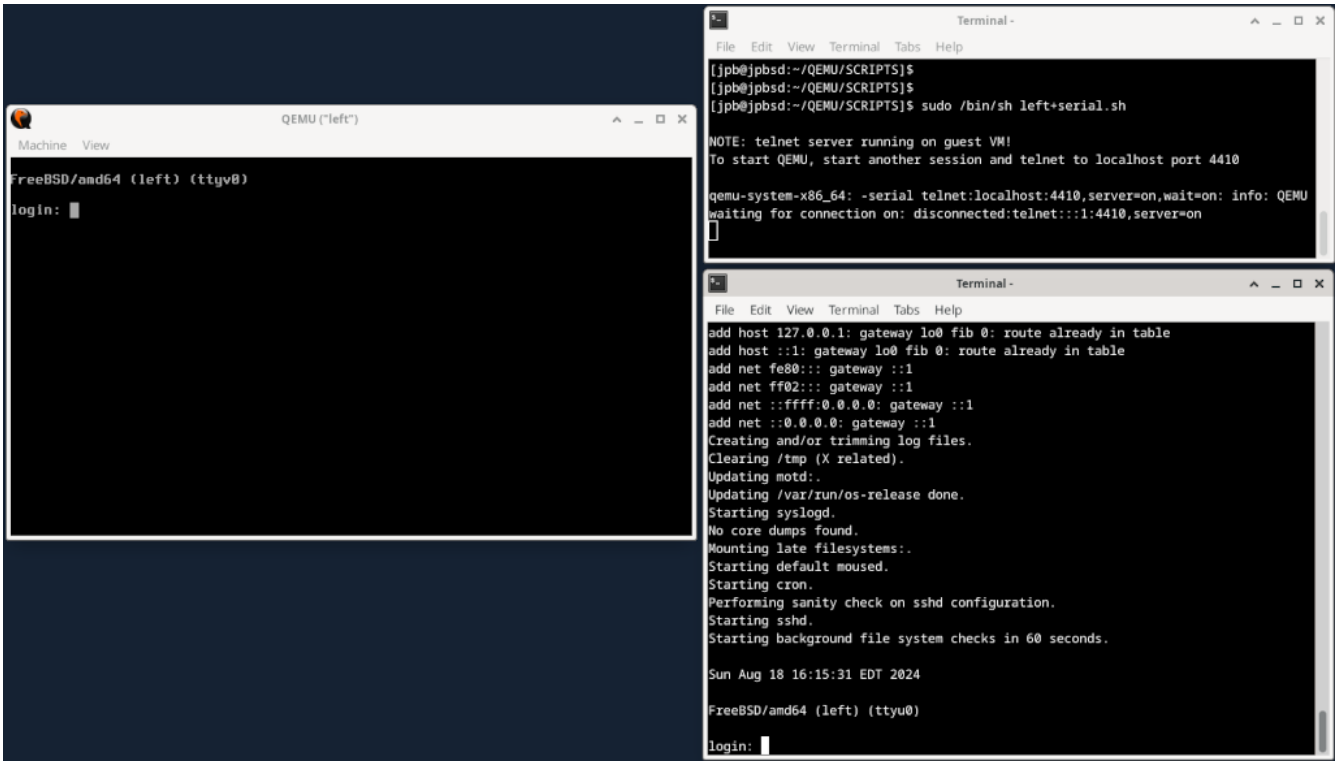


Рисунок 67. Включение последовательного порта через TCP

На рисунке 8 последовательный порт перенаправляется на TCP-порт хост-системы при запуске ВМ, а монитор QEMU ожидает (`wait=on`), активируя гостевую ВМ только после установления соединения telnet(1) с указанным портом localhost. После получения соединения из отдельного сеанса система FreeBSD начинает загрузку и ищет директиву консоли в /boot/loader.conf. С директивой "console=comconsole" FreeBSD запускает консольный сеанс на последовательном порту. Монитор QEMU обнаруживает это и направляет необходимый символьный ввод-вывод с этого последовательного порта в telnet-сеанс на хосте. Система загружается, и после завершения загрузки приглашения для входа в систему становятся доступными на последовательном порту (`ttyu0`) и на консоли (`ttyv0`).

Важно отметить, что этот последовательный перенаправление через TCP происходит вне виртуальной машины. Нет взаимодействия с какой-либо сетью внутри виртуальной машины, и поэтому оно не подчиняется никаким правилам брандмауэра. Представьте это как простой терминал, подключённый к RS-232 или USB-порту на реальном компьютере.

Заметки об использовании последовательной консоли

На последовательной консоли, если размер окна изменен, выполните resizewin(1), чтобы обновить размер терминала.

Может быть желательно (или даже необходимо) остановить отправку сообщений syslog на консоль (как консоль QEMU, так и последовательный порт). Подробности о перенаправлении сообщений консоли можно найти в syslog.conf(5).

После обновления файла /boot/loader.conf для разрешения использования последовательной консоли гостевая ВМ будет пытаться загружаться с последовательного порта при каждом запуске. Убедитесь, что последовательный порт включен, как показано в приведённом выше

листинге, или измените файл /boot/loader.conf, чтобы не требовать использования последовательной консоли.

25.6.5. Эмуляция пользовательского режима QEMU

QEMU также поддерживает выполнение приложений, скомпилированных для архитектуры, отличной от архитектуры основного процессора. Например, можно запустить операционную систему архитектуры Sparc64 на хосте с архитектурой x86_64. Это демонстрируется в следующем разделе.

Настройка гостевой ВМ SPARC64 на хосте x86_64

Настройка новой виртуальной машины с архитектурой, отличной от архитектуры хоста, включает несколько шагов:

- Получение программного обеспечения, которое будет работать на гостевой ВМ
- Создание нового образа диска для гостевой ВМ
- Настройка нового скрипта QEMU с новой архитектурой
- Выполнение установки

В следующей процедуре используется копия программного обеспечения OpenBSD 6.8 SPARC64 для этого упражнения по эмуляции пользовательского режима QEMU.

Не все версии OpenBSD Sparc64 работают в QEMU. Известно, что OpenBSD версии 6.8 работает, и она была выбрана в качестве примера для этого раздела.

1. Скачать OpenBSD 6.8 Sparc64 из архива OpenBSD.

   На сайтах загрузки OpenBSD поддерживаются только самые последние версии. Для получения предыдущих выпусков необходимо искать в архиве.

   ```
   % cd ~/QEMU/ISO
   % fetch https://mirror.planetunix.net/pub/OpenBSD-archive/6.8/sparc64/install68.iso
   ```

2. Создание нового образа диска для виртуальной машины Sparc64 аналогично описанному выше для "правильной" виртуальной машины. В данном случае используется формат диска QEMU qcow2:

   ```
   % cd ~/QEMU/VM
   qemu-img create -f qcow2 -o preallocation=full,lazy_refcounts=on sparc64.qcow2 16G
   ```

3. Используйте приведённый ниже скрипт для новой архитектуры Sparc64. Как и в предыдущем примере, запустите скрипт, затем начните новую сессию и подключитесь через `telnet` к localhost на указанном порту:

```
echo
echo "NOTE: telnet startup server running on guest VM!"
echo "To start QEMU, start another session and telnet to localhost port 4410"
echo

/usr/local/bin/qemu-system-sparc64 \
  -serial telnet:localhost:4410,server=on,wait=on \
  -machine sun4u,usb=off \
  -smp 1,sockets=1,cores=1,threads=1 \
  -rtc base=utc \
  -m 1024 \
  -boot d \
  -drive file=../VM/sparc64.qcow2,if=none,id=drive-ide0-0-1,format=qcow2,cache=none
\
  -cdrom ../ISO/install68.iso \
  -device ide-hd,bus=ide.0,unit=0,drive=drive-ide0-0-1,id=ide0-0-1 \
  -msg timestamp=on \
  -net nic,model=sunhme -net user \
  -nographic \
  -name \"sparc64\"
```

Обратите внимание на следующее:

- Опция `-boot d` загружает систему с устройства CDROM QEMU, которое указано как `-cdrom ../ISO/install68.iso`.
- Как и ранее, параметр сервера `telnet` настроен на ожидание отдельного подключения на порту 4410. Запустите ещё один сеанс и используйте telnet(1) для подключения к localhost на порту 4410.
- Скрипт устанавливает опцию `-nographic`, что означает использование только ввода/вывода через последовательный порт. Графический интерфейс отсутствует.
- Сетевое взаимодействие не настраивается через комбинацию tap(4) / if_bridge(4). В этом примере используется отдельный метод сетевого взаимодействия QEMU, известный как "Serial Line Internet Protocol" (SLIRP), иногда называемый "User Mode Networking". Документация по этому и другим методам сетевого взаимодействия QEMU находится здесь: Документация по сетевому взаимодействию QEMU

Если все настроено правильно, система загрузится, как показано на рисунке 9.

Рисунок 68. QEMU Загрузка OpenBSD 6.8 Sparc64 с CDROM с эмуляцией пользовательского режима

После установки системы измените скрипт и замените параметр загрузки на `-boot c`. Это укажет QEMU загружаться с предоставленного жёсткого диска, а не с CDROM.

Установленная система может использоваться как любая другая гостевая виртуальная машина. Однако, базовая архитектура гостевой системы — Sparc64, а не x86_64.

Если система остановлена на приглашении консоли OpenBios `0 >`, введите `power-off`, чтобы завершить работу системы.

На рисунке 10 показан вход в систему от имени root в установленной системе и

выполнение uname(1).

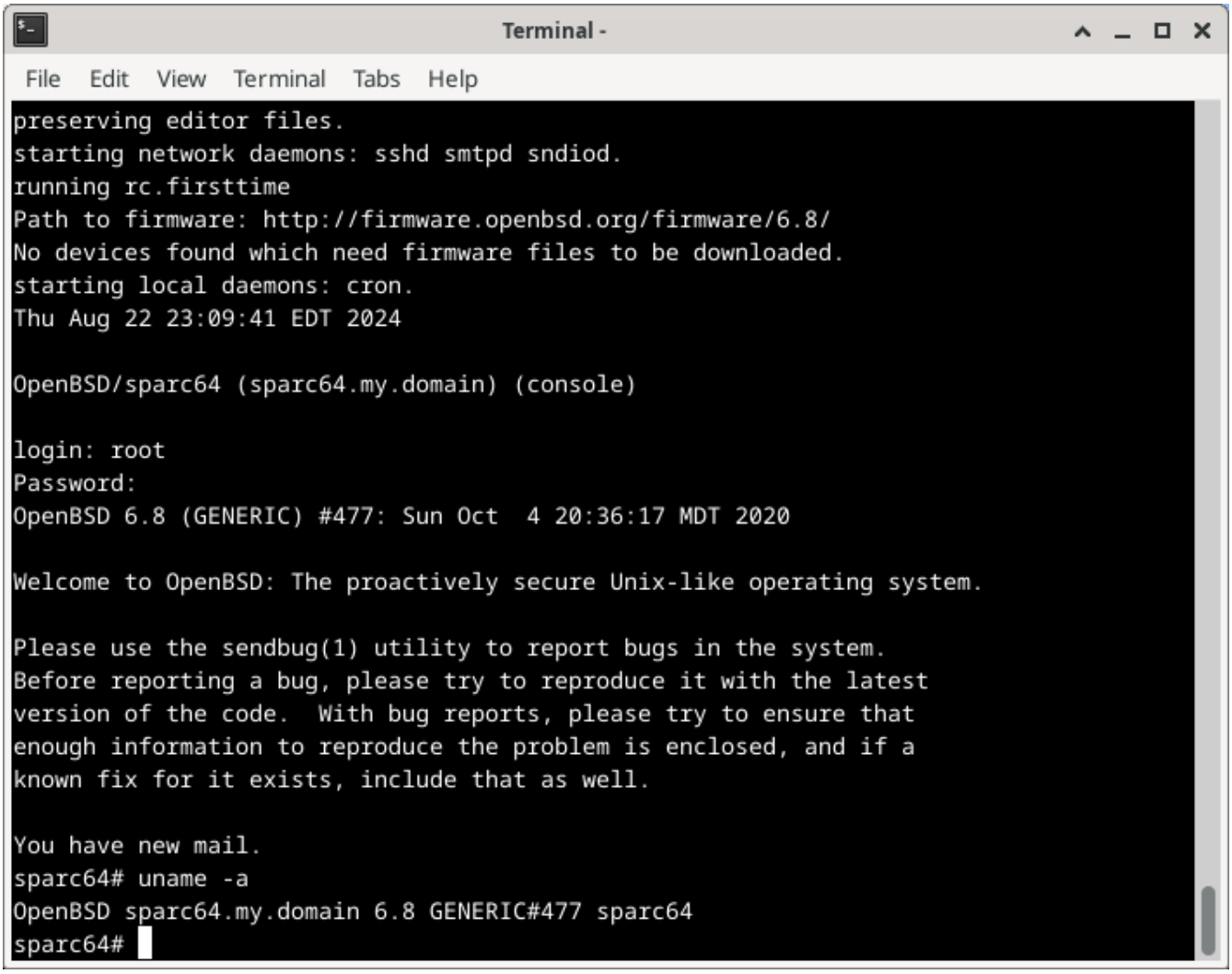


Рисунок 69. QEMU Загрузка с CDROM с эмуляцией пользовательского режима

25.6.6. Использование монитора QEMU

Монитор QEMU управляет работающим эмулятором QEMU (гостевой виртуальной машиной).

Используя монитор, можно:

- Динамически удалять или добавлять устройства, включая диски, сетевые интерфейсы, CD-ROM или дисководы гибких дисков
- Заморозить/разморозить гостевую ВМ и сохранить или восстановить её состояние из файла на диске
- Собрать информацию о состоянии ВМ и устройств
- Изменить настройки устройств на лету

А также сделать множество других операций.

Наиболее распространённые способы использования монитора — это проверка состояния виртуальной машины, а также добавление, удаление или изменение устройств. Некоторые операции, такие как миграция, доступны только при использовании гипервизоров с

ускорением, например KVM, Xen и т.д., и не поддерживаются на хостах FreeBSD.

При использовании графической среды рабочего стола самый простой способ доступа к монитору QEMU — это опция `-monitor stdio` при запуске QEMU из терминала.

```
# /usr/local/bin/qemu-system-x86_64  -monitor stdio \
  -cpu qemu64 \
  -vga cirrus \
  -m 4096  -smp 4   \
  ...
```

Это приводит к появлению новой строки приглашения `(qemu)` в окне терминала, как показано на рисунке 11.

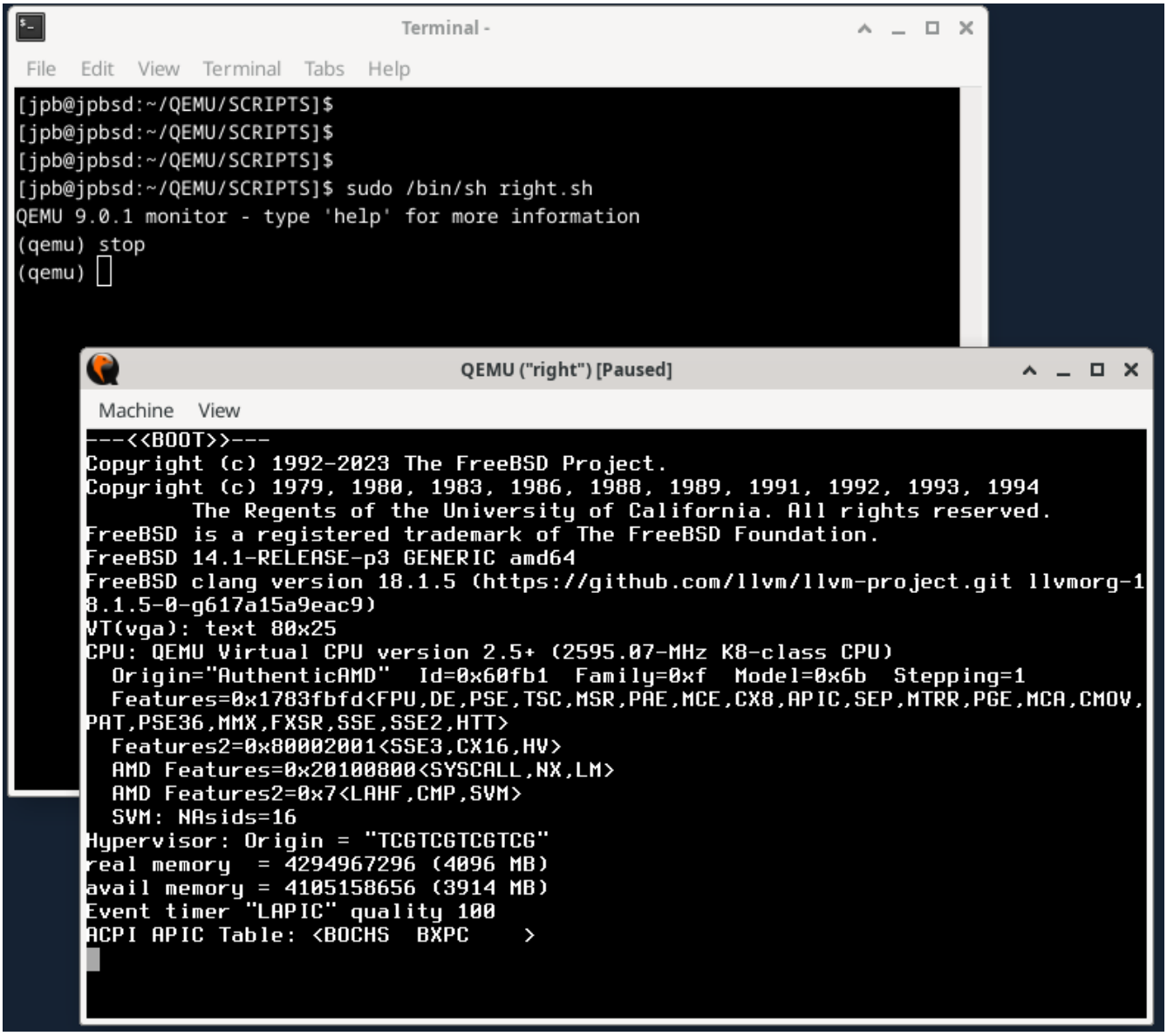


Рисунок 70. Приглашение монитора QEMU и команда "stop"

На изображении также показано, что команда `stop` замораживает систему во время последовательности загрузки FreeBSD. Система останется замороженной до тех пор, пока в мониторе не будет введена команда `cont`.

Добавление нового диска в виртуальную машину

Чтобы добавить новый диск к работающей виртуальной машине, его необходимо подготовить, как описано выше:

```
% cd ~/QEMU/VM
% qemu-img create -f raw  new10G.img  10G
```

Рисунок 12 показывает последовательность команд в мониторе, необходимую для добавления нового диска в виртуальную машину. После добавления устройства с помощью команды `device_add` в мониторе оно появляется на консоли системы FreeBSD, как показано в нижней части рисунка. Диск можно настроить по необходимости.

Обратите внимание, что новый диск должен быть добавлен в стартовый скрипт, если он будет использоваться после перезагрузки виртуальной машины.

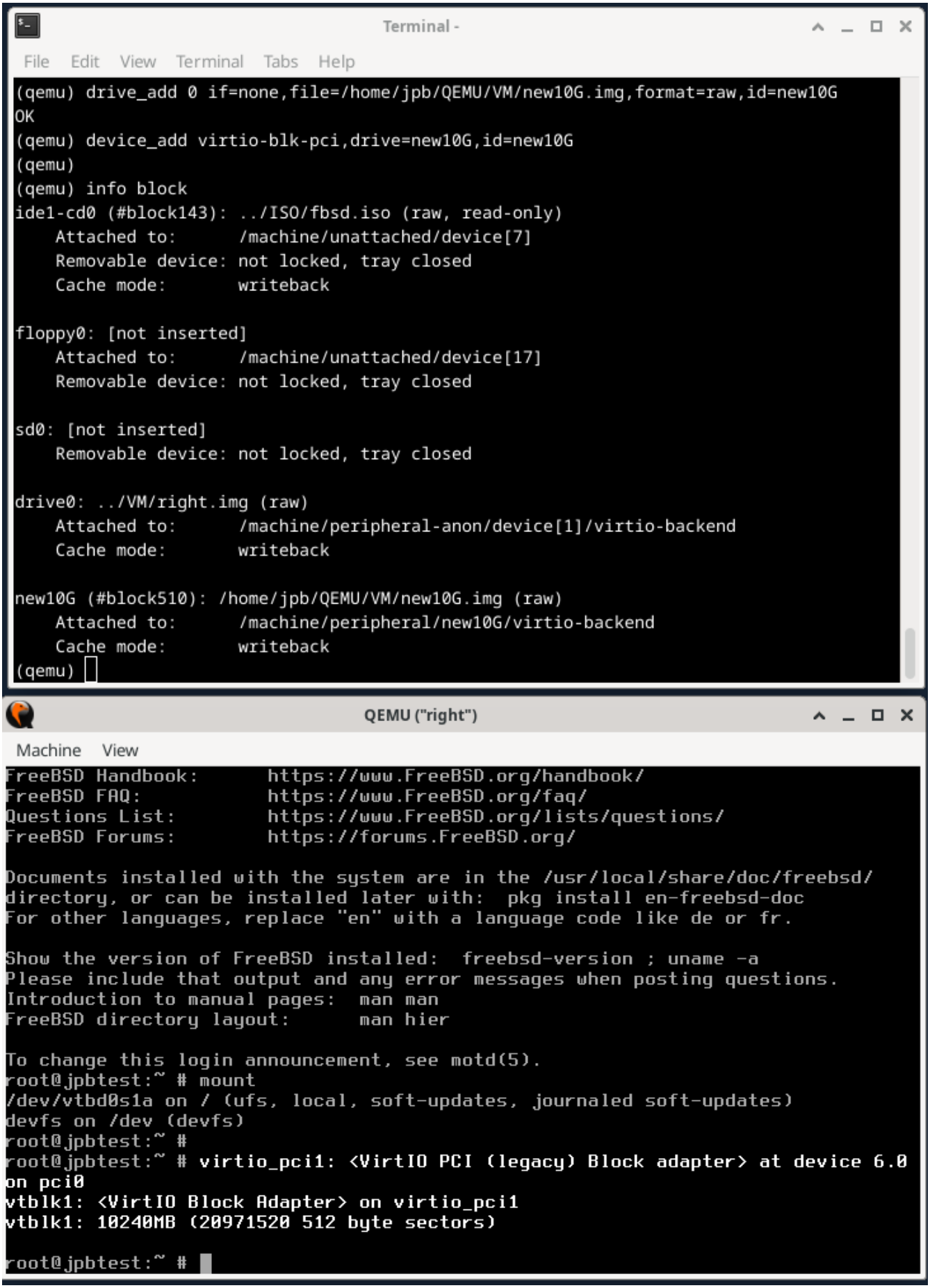


Рисунок 71. Команды монитора QEMU для добавления нового диска

Использование монитора QEMU для управления снимками

Документация QEMU описывает несколько схожих концепций, используя термин **снимок (snapshot)**. Существует опция `-snapshot` в командной строке, которая означает использование диска или его части для хранения копии устройства. Также есть команды монитора `snapshot_blkdev` и `snapshot_blkdev_internal`, описывающие сам процесс копирования блочного устройства. Наконец, команды монитора `savevm`, `loadvm` и `delvm` относятся к созданию, сохранению, загрузке или удалению копии всей виртуальной машины. Вместе с последними, команда монитора `info snapshots` выводит детали недавних снимков состояния.

Этот раздел посвящён созданию, сохранению и загрузке полного образа ВМ и использует термин **снимок** для этой цели.

Для начала заново создайте виртуальную машину "left", на этот раз используя формат `qcow2`.

```
% cd ~/QEMU/VM
% rm left.img
% qemu-img create -f qcow2 left.qcow2 16G  # Clean file for a new FreeBSD
installation.
% cd ../SCRIPTS
# /bin/sh left.sh                     # See the below program listing.
```

После завершения установки перезагрузите систему, на этот раз с использованием опции `-monitor stdio`, чтобы обеспечить доступ к монитору.

```
# left VM script.
/usr/local/bin/qemu-system-x86_64  -monitor stdio \
  -cpu qemu64 \
  -vga std \
  -m 4096 \
  -smp 4   \
  -cdrom ../ISO/fbsd.iso \
  -boot order=cd,menu=on \
  -blockdev driver=file,aio=threads,node-name=imgleft,filename=../VM/left.qcow2 \
  -blockdev driver=qcow2,node-name=drive0,file=imgleft \
  -device virtio-blk-pci,drive=drive0,bootindex=1  \
  -netdev tap,id=nd0,ifname=tap0,script=no,downscript=no,br=bridge0 \
  -device e1000,netdev=nd0,mac=02:20:6c:65:66:74 \
  -name \"left\"
```

Для демонстрации работы снимков можно использовать следующую процедуру:

1. Установите FreeBSD с нуля
2. Подготовьте окружение и создайте снимок состояния с помощью команды монитора `savevm`
3. Установите несколько пакетов

4. Выключите систему
5. Перезапустите экземпляр QEMU без оболочки и используйте команду монитора `loadvm` для восстановления ВМ
6. Обратите внимание, что восстановленная ВМ не имеет установленных пакетов

На этапе «Подготовка окружения» в отдельной виртуальной консоли (ttyv1) запускается сеанс редактирования в vi(1), имитирующий активность пользователя. При желании можно запустить дополнительные программы. Снимок должен учитывать состояние всех приложений, работающих на момент его создания.

Рисунок 13 показывает новую установленную систему FreeBSD без пакетов, а также отдельно сеанс редактирования на ttyv1. Редактор vi(1) в данный момент находится в режиме `insert`, и пользователь набирает слово "broadcast".

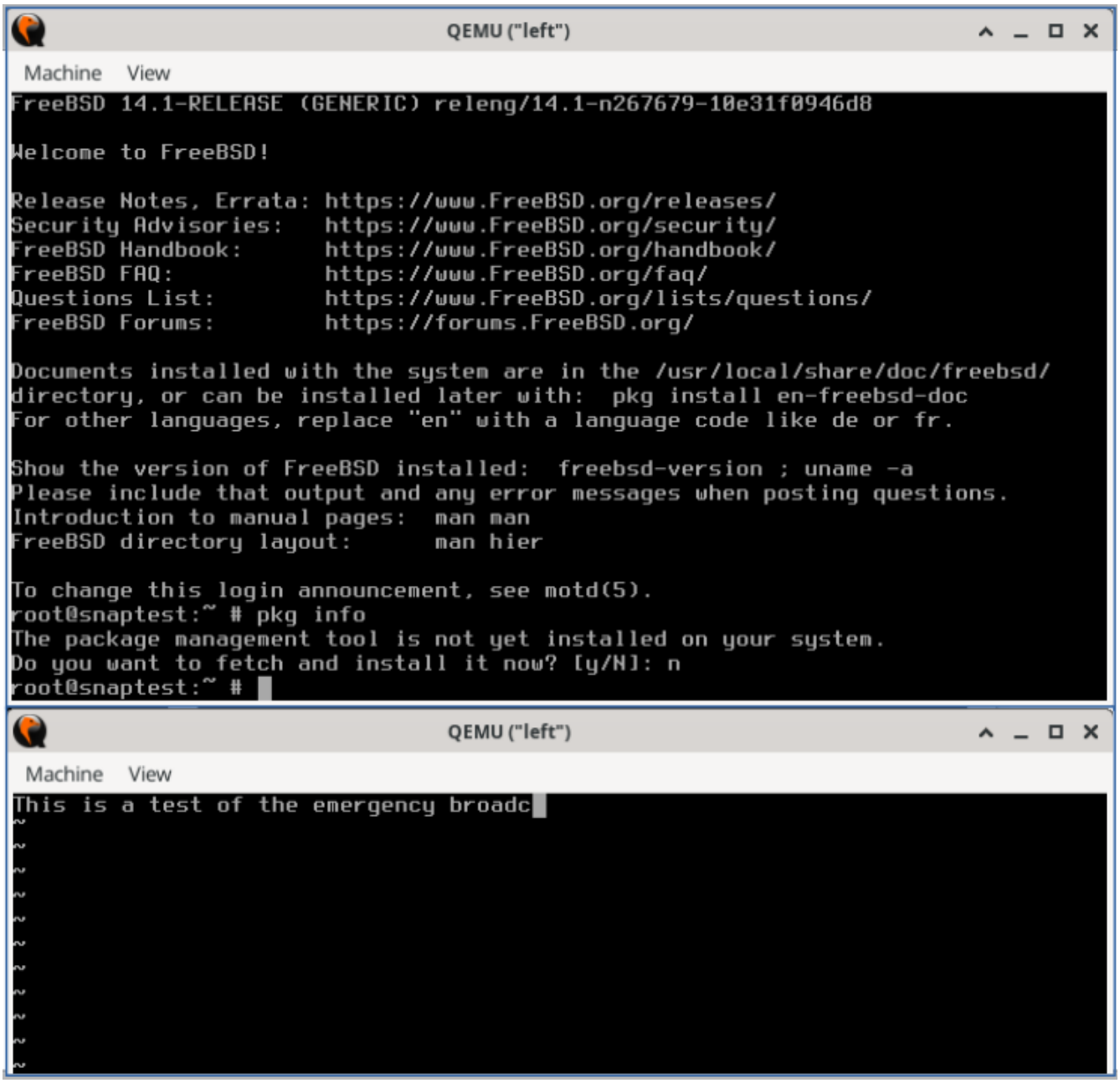


Рисунок 72. Виртуальная машина QEMU перед первым снимком состояния

Чтобы создать снимок состояния, введите `savevm` в мониторе. Убедитесь, что указали метку

(например, `original_install`).

```
QEMU 9.0.1 monitor - type 'help' for more information
(qemu)
(qemu) savevm original_install
```

Далее, в главном окне консоли, установите пакет, например zip(1), который не имеет зависимостей. После завершения установки вернитесь в монитор и создайте ещё один снимок состояния (`snap1_pkg+zip`).

Рисунок 14 показывает результаты выполненных выше команд и вывод команды `info shapshots`.

```
[jpb@jpbsd:~/QEMU/SCRIPTS]$ sudo /bin/sh left.sh
QEMU 9.0.1 monitor - type 'help' for more information
(qemu) savevm original_install
(qemu)
(qemu) savevm snap1_pkg+zip
(qemu)
(qemu) info snapshots
List of snapshots present on all disks:
ID      TAG              VM_SIZE                DATE        VM_CLOCK     ICOUNT
--      original_install  2.4 GiB 2024-09-03 21:27:21  0000:39:06.533         --
--      snap1_pkg+zip    2.79 GiB 2024-09-03 21:33:27  0000:44:59.614         --
(qemu)
(qemu)
(qemu)
```

Рисунок 73. QEMU — Использование команд монитора для создания снимков состояния

Перезагрузите систему, и до запуска FreeBSD переключитесь на монитор и введите `stop`. Виртуальная машина остановится.

Введите `loadvm` с тегом, который вы использовали ранее (здесь `original_install`).

```
QEMU 9.0.1 monitor - type 'help' for more information
(qemu) stop
(qemu) loadvm original_install
(qemu) cont
```

Сразу же экран виртуальной машины переключится на тот момент, когда была введена команда `savevm`, как указано выше. Обратите внимание, что виртуальная машина всё ещё остановлена.

Введите `cont`, чтобы запустить ВМ, переключитесь на сеанс редактирования на `ttyv1` и наберите одну букву на клавиатуре. Редактор, всё ещё находящийся в режиме вставки, должен отреагировать соответствующим образом. Любые другие программы, работавшие в момент создания снимка, не должны быть затронуты.

Приведенные выше шаги показывают, как можно создать снимок состояния, изменить

систему, а затем «откатить» изменения, восстановив предыдущий снимок.

По умолчанию QEMU хранит данные снимков в том же файле, что и образ. Просмотреть список снимков можно с помощью qemu-img(1), как показано ниже в Рисунок 15.

```
[jpb@jpbsd:~/QEMU/VM]$ qemu-img info left.qcow2
image: left.qcow2
file format: qcow2
virtual size: 16 GiB (17179869184 bytes)
disk size: 3.76 GiB
cluster_size: 65536
Snapshot list:
ID      TAG               VM_SIZE                DATE        VM_CLOCK     ICOUNT
1       original_install  2.4 GiB 2024-09-03 21:27:21  0000:39:06.533         --
2       snap1_pkg+zip    2.79 GiB 2024-09-03 21:33:27  0000:44:59.614         --
Format specific information:
    compat: 1.1
    compression type: zlib
    lazy refcounts: false
    refcount bits: 16
    corrupt: false
    extended l2: false
Child node '/file':
    filename: left.qcow2
    protocol type: file
    file length: 7.2 GiB (7735148544 bytes)
    disk size: 3.76 GiB
[jpb@jpbsd:~/QEMU/VM]$
```

Рисунок 74. QEMU Использование qemu-img(1) для проверки снимков

25.6.7. Использование USB-устройств в QEMU

QEMU поддерживает создание виртуальных USB-устройств, которые используют файл образа. Это виртуальные USB-устройства, которые можно разбивать на разделы, форматировать, монтировать и использовать, как реальное USB-устройство.

```
/usr/local/bin/qemu-system-x86_64  -monitor stdio \
  -cpu qemu64 \
  -vga cirrus \
  -m 4096  -smp 4   \
  -cdrom ../ISO/fbsd.iso \
  -boot order=cd,menu=on \
  -drive if=none,id=usbstick,format=raw,file=../VM/foo.img \
  -usb \
  -device usb-ehci,id=ehci \
  -device usb-storage,bus=ehci.0,drive=usbstick \
  -device usb-mouse \
  -blockdev driver=file,node-name=img1,filename=../VM/right.qcow2 \
  -blockdev driver=qcow2,node-name=drive0,file=img1 \
  -device virtio-blk-pci,drive=drive0,bootindex=1  \
  -netdev tap,id=nd0,ifname=tap1,script=no,downscript=no,br=bridge0 \
  -device e1000,netdev=nd0,mac=02:72:69:67:68:74 \
```

```
  -name \"right\"
```

Эта конфигурация включает спецификацию `-drive` с `id=usbstick`, формат raw и файл образ (который должен быть создан с помощью qemu-img(1)). Следующая строка содержит спецификацию `-device usb-ehci` для контроллера USB EHCI с `id=ehci`. Наконец, спецификация `-device usb-storage` связывает указанный накопитель с шиной USB EHCI.

При загрузке системы FreeBSD распознает USB-концентратор, добавит подключённое USB-устройство и назначит его на `da0`, как показано на рисунке 16.

Рисунок 75. QEMU Созданные USB-концентратор и устройство хранения данных

Устройство готово к разделению с помощью gpart(8) и форматированию с помощью newfs(8). Поскольку USB-устройство использует файл, созданный qemu-img(1), записанные на него данные сохранятся после перезагрузки.

25.6.8. Использование USB-устройств хоста через проброс (passthrough)

Поддержка проброса USB в QEMU указана как экспериментальная в версии 9.0.1 (лето 2024 года). Тем не менее, следующие шаги показывают, как USB-накопитель, подключённый к хосту, может быть использован в гостевой ВМ.

Для получения дополнительной информации и примеров см.:

- https://www.qemu.org/docs/master/system/devices/usb.html

Верхняя часть рисунка 17 показывает команды монитора QEMU:

- `info usbhost` отображает информацию обо всех USB-устройствах в хост-системе. Найдите нужное USB-устройство в хост-системе и запишите два шестнадцатеричных значения в соответствующей строке. (В примере ниже хост-устройство — это Memorex Mini с vendorid 0718 и productid 0619.) Используйте эти два значения, показанные командой `info usbhost`, на этапе `device_add` ниже.
- `device_add` добавляет USB-устройство в гостевую ВМ.

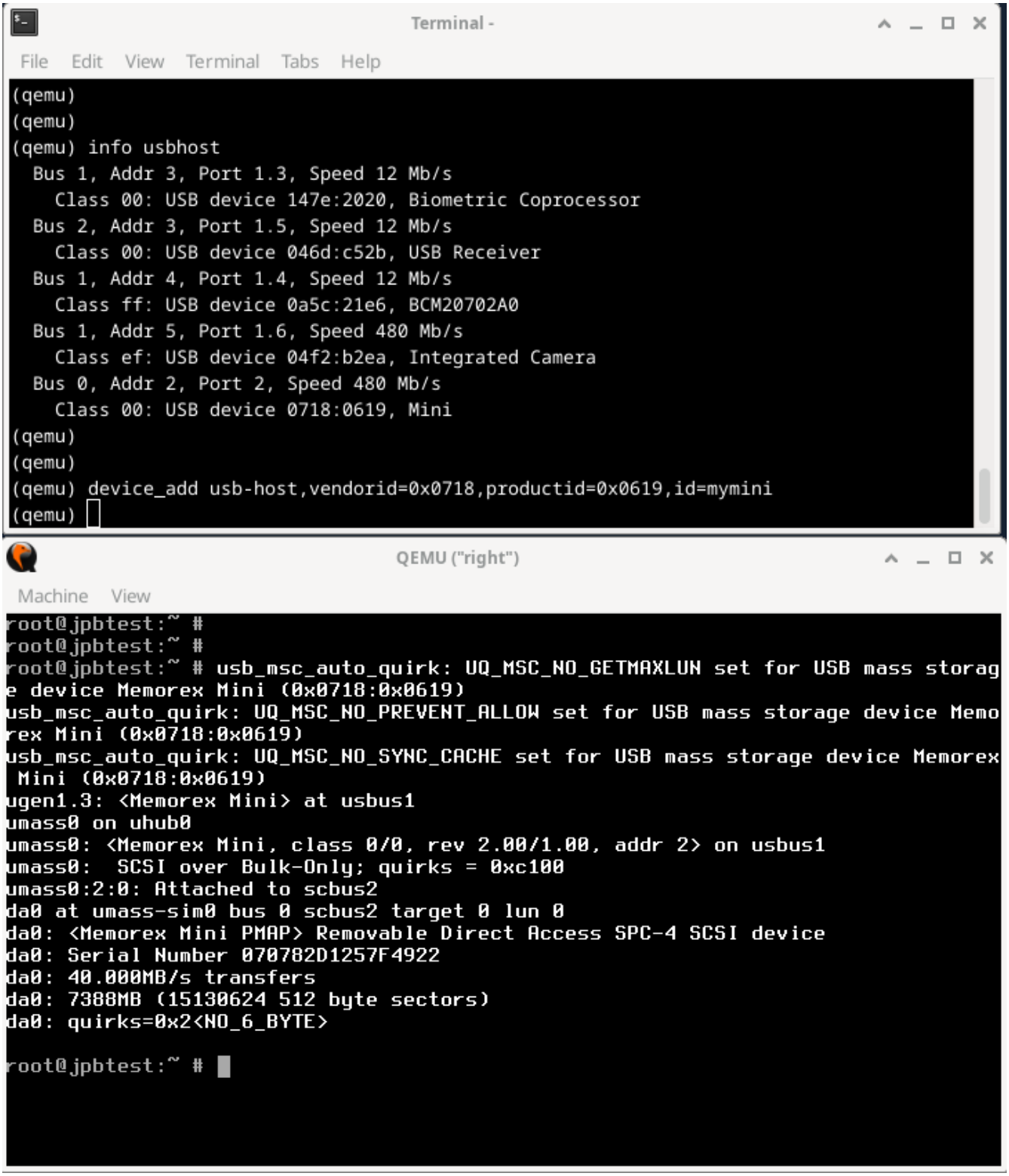


Рисунок 76. Команды монитора QEMU для доступа к USB-устройству на хосте

Как и ранее, после завершения `device_add` ядро FreeBSD распознает новое USB-устройство, как показано в нижней части экрана.

Использование нового устройства показано на рисунке 18.

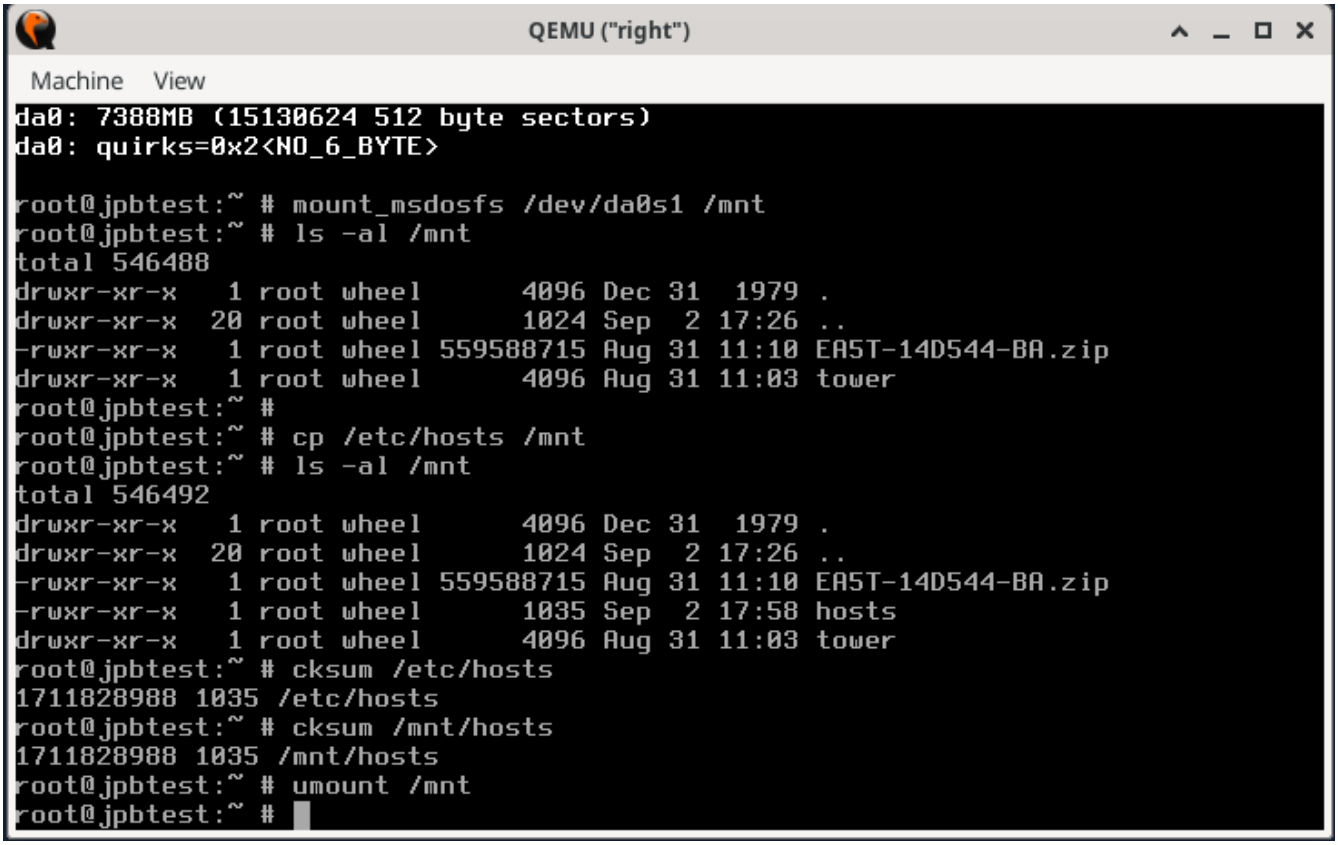


Рисунок 77. Использование USB-устройства хоста через проброс (passthrough)

Если USB-устройство отформатировано как файловая система FAT16 или FAT32, его можно подключить как файловую систему MS-DOS™ с помощью mount_msdosfs(8), как показано в примере. Файл `/etc/hosts` копируется на только что подключённый диск, и для проверки целостности файла на USB-устройстве вычисляются контрольные суммы. Затем устройство отключается с помощью umount(8).

Если USB-устройство отформатировано в NTFS, необходимо установить пакет `fusefs-ntfs` и использовать ntfs-3g(8) для доступа к устройству:

```
# pkg install fusefs-ntfs
# kldload fusefs
# gpart show da1
# ntfs-3g /dev/da1s1 /mnt

Access the drive as needed.  When finished:

# umount /mnt
```

Замените приведённые выше идентификаторы устройств в соответствии с установленным оборудованием. Дополнительную информацию о работе с файловыми системами NTFS см. в ntfs-3g(8).

25.6.9. Краткое описание QEMU на FreeBSD

Как упоминалось выше, QEMU работает с несколькими различными гипервизорными ускорителями.

Список поддерживаемых QEMU ускорителей виртуализации включает:

- `KVM` в Linux с поддержкой 64-битных архитектур Arm, MIPS, PPC, RISC-V, s390x и x86
- `Xen` на Linux в качестве dom0 с поддержкой Arm и x86
- `Hypervisor Framework (hvf)` в macOS с поддержкой x86 и Arm (только 64-битные)
- `Windows Hypervisor Platform (whpx)` в Windows с поддержкой x86
- `Диспетчер виртуальных машин NetBSD (nvmm)` на NetBSD с поддержкой x86
- `Tiny Code Generator (tcg)` на Linux и других POSIX-совместимых системах, Windows, macOS с поддержкой Arm, x86, Loongarch64, MIPS, PPC, s390x и Sparc64.

Все примеры в этом разделе использовали ускоритель `Tiny Code Generator (tcg)`, так как это единственный поддерживаемый ускоритель в FreeBSD на данный момент.

25.7. FreeBSD в качестве хоста с bhyve

Гипервизор bhyve с лицензией BSD стал частью базовой системы начиная с FreeBSD 10.0-RELEASE. Этот гипервизор поддерживает несколько гостевых систем, включая FreeBSD, OpenBSD, многие дистрибутивы Linux® и Microsoft Windows®. По умолчанию bhyve предоставляет доступ к последовательной консоли и не эмулирует графическую консоль. Функции оффлоадинга виртуализации современных процессоров используются для избежания устаревших методов трансляции инструкций и ручного управления отображением памяти.

Дизайн bhyve требует

- процессор Intel®, поддерживающий Intel Extended Page Tables (EPT),
- или процессор AMD®, поддерживающий AMD Rapid Virtualization Indexing (RVI) или Nested Page Tables (NPT),
- или процессор ARM® aarch64.

На ARM поддерживается только чистая виртуализация ARMv8.0, расширения Virtualization Host Extensions в настоящее время не используются. Для запуска гостевых систем Linux® или FreeBSD с более чем одним vCPU требуется поддержка неограниченного режима VMX (UG).

Самый простой способ проверить, поддерживает ли процессор Intel или AMD технологию bhyve, — выполнить команду `dmesg` или посмотреть в файле /var/run/dmesg.boot флаг `POPCNT` в строке `Features2` для процессоров AMD® или флаги `EPT` и `UG` в строке `VT-x` для процессоров Intel®.

25.7.1. Подготовка хоста

Первым шагом к созданию виртуальной машины в bhyve является настройка хостовой системы. Сначала загрузите модуль ядра bhyve:

```
# kldload vmm
```

Существует несколько способов подключения гостевой виртуальной машины к сети хоста; один из простых способов — создать интерфейс tap, к которому подключится сетевое устройство виртуальной машины. Чтобы сетевое устройство могло участвовать в сети, также необходимо создать мостовой интерфейс, включающий интерфейс tap и физический интерфейс в качестве членов. В данном примере физический интерфейс — это *igb0*:

```
# ifconfig tap0 create
# sysctl net.link.tap.up_on_open=1
net.link.tap.up_on_open: 0 -> 1
# ifconfig bridge0 create
# ifconfig bridge0 addm igb0 addm tap0
# ifconfig bridge0 up
```

25.7.2. Создание гостевой системы FreeBSD

Создайте файл, который будет использоваться как виртуальный диск для гостевой машины. Укажите размер и имя виртуального диска:

```
# truncate -s 16G guest.img
```

Скачать образ установки FreeBSD для установки:

```
# fetch https://download.freebsd.org/releases/ISO-IMAGES/14.0/FreeBSD-14.0-RELEASE-
amd64-bootonly.iso
FreeBSD-14.0-RELEASE-amd64-bootonly.iso                426 MB   16 MBps    22s
```

FreeBSD включает пример скрипта `vmrun.sh` для запуска виртуальной машины в bhyve. Он запускает виртуальную машину и выполняет её в цикле, поэтому она автоматически перезапустится в случае сбоя. `vmrun.sh` принимает несколько опций для управления конфигурацией машины, включая:

- `-c` управляет количеством виртуальных процессоров,
- `-m` ограничивает объём памяти, доступной гостевой системе,
- `-t` указывает, какой tap-устройство использовать,
- `-d` указывает, какой образ диска использовать,
- `-i` указывает bhyve загружаться с образа CD вместо диска, и
- `-I` определяет, какой образ CD использовать.

Последний параметр — это имя виртуальной машины, которое используется для отслеживания работающих машин. Следующая команда выводит список всех доступных аргументов программы:

```
# sh /usr/share/examples/bhyve/vmrun.sh -h
```

Этот пример запускает виртуальную машину в режиме установки:

```
# sh /usr/share/examples/bhyve/vmrun.sh -c 1 -m 1024M -t tap0 -d guest.img \
     -i -I FreeBSD-14.0-RELEASE-amd64-bootonly.iso guestname
```

Виртуальная машина загрузится и запустит установщик. После установки системы в виртуальной машине, когда система предложит перейти в оболочку в конце установки, выберите **[Да]**.

Перезагрузите виртуальную машину. Хотя перезагрузка виртуальной машины приводит к выходу из bhyve, скрипт vmrun.sh запускает `bhyve` в цикле и автоматически перезапустит его. Когда это произойдет, выберите опцию перезагрузки в меню загрузчика, чтобы выйти из цикла. Теперь гостевую систему можно запустить с виртуального диска:

```
# sh /usr/share/examples/bhyve/vmrun.sh -c 4 -m 1024M -t tap0 -d guest.img guestname
```

25.7.3. Создание гостевой системы Linux®

Гостевые системы Linux можно загружать как любую другую обычную виртуальную машину на основе UEFI, либо, в качестве альтернативы, можно использовать порт sysutils/grub2-bhyve.

Для этого сначала убедитесь, что порт установлен, затем создайте файл, который будет использоваться как виртуальный диск для гостевой машины:

```
# truncate -s 16G linux.img
```

Запуск виртуальной машины Linux с `grub2-bhyve` — это двухэтапный процесс.

1. Сначала необходимо загрузить ядро, затем можно запустить гостевую систему.
2. Ядро Linux® загружается с помощью пакета sysutils/grub2-bhyve.

Создайте файл device.map, который grub будет использовать для сопоставления виртуальных устройств с файлами в хостовой системе:

```
(hd0) ./linux.img
(cd0) ./somelinux.iso
```

Используйте sysutils/grub2-bhyve для загрузки ядра Linux® из образа ISO:

```
# grub-bhyve -m device.map -r cd0 -M 1024M linuxguest
```

Это запустит grub. Если на установочном CD содержится файл grub.cfg, будет отображено меню. Если нет, файлы `vmlinuz` и `initrd` необходимо найти и загрузить вручную:

```
grub> ls
(hd0) (cd0) (cd0,msdos1) (host)
grub> ls (cd0)/isolinux
boot.cat boot.msg grub.conf initrd.img isolinux.bin isolinux.cfg memtest
splash.jpg TRANS.TBL vesamenu.c32 vmlinuz
grub> linux (cd0)/isolinux/vmlinuz
grub> initrd (cd0)/isolinux/initrd.img
grub> boot
```

Теперь, когда ядро Linux® загружено, можно запустить гостевую систему:

```
# bhyve -A -H -P -s 0:0,hostbridge -s 1:0,lpc -s 2:0,virtio-net,tap0 \
    -s 3:0,virtio-blk,./linux.img -s 4:0,ahci-cd,./somelinux.iso \
    -l com1,stdio -c 4 -m 1024M linuxguest
```

Система загрузится и запустит установщик. После установки системы в виртуальной машине перезагрузите виртуальную машину. Это приведёт к завершению работы bhyve. Экземпляр виртуальной машины необходимо уничтожить перед тем, как его можно будет запустить снова:

```
# bhyvectl --destroy --vm=linuxguest
```

Теперь гостевую систему можно запустить напрямую с виртуального диска. Загрузите ядро:

```
# grub-bhyve -m device.map -r hd0,msdos1 -M 1024M linuxguest
grub> ls
(hd0) (hd0,msdos2) (hd0,msdos1) (cd0) (cd0,msdos1) (host)
(lvm/VolGroup-lv_swap) (lvm/VolGroup-lv_root)
grub> ls (hd0,msdos1)/
lost+found/ grub/ efi/ System.map-2.6.32-431.el6.x86_64 config-2.6.32-431.el6.x
86_64 symvers-2.6.32-431.el6.x86_64.gz vmlinuz-2.6.32-431.el6.x86_64
initramfs-2.6.32-431.el6.x86_64.img
grub> linux (hd0,msdos1)/vmlinuz-2.6.32-431.el6.x86_64 root=/dev/mapper/VolGroup-
lv_root
grub> initrd (hd0,msdos1)/initramfs-2.6.32-431.el6.x86_64.img
grub> boot
```

Запустите виртуальную машину:

```
# bhyve -A -H -P -s 0:0,hostbridge -s 1:0,lpc -s 2:0,virtio-net,tap0 \
    -s 3:0,virtio-blk,./linux.img -l com1,stdio -c 4 -m 1024M linuxguest
```

Linux® теперь загрузится в виртуальной машине и в конечном итоге покажет приглашение для входа. Войдите в систему и используйте виртуальную машину. Когда вы закончите, перезагрузите виртуальную машину для выхода из bhyve. Уничтожьте экземпляр виртуальной машины:

```
# bhyvectl --destroy --vm=linuxguest
```

25.7.4. Загрузка виртуальных машин bhyve с прошивкой UEFI

В дополнение к `bhyveload` и `grub-bhyve`, гипервизор bhyve также может загружать виртуальные машины с использованием прошивки UEFI. Этот вариант может поддерживать гостевые операционные системы, которые не поддерживаются другими загрузчиками.

Чтобы использовать поддержку UEFI в bhyve, сначала получите образы прошивки UEFI. Это можно сделать, установив порт sysutils/bhyve-firmware или пакет.

Имея микропрограмму, добавьте флаги `-l bootrom,/путь/к/микропрограмме` в командную строку bhyve. Фактическая команда bhyve может выглядеть так:

```
# bhyve -AHP -s 0:0,hostbridge -s 1:0,lpc \
    -s 2:0,virtio-net,tap1 -s 3:0,virtio-blk,./disk.img \
    -s 4:0,ahci-cd,./install.iso -c 4 -m 1024M \
    -l bootrom,/usr/local/share/uefi-firmware/BHYVE_UEFI.fd \
    guest
```

Чтобы разрешить гостевой системе сохранять переменные UEFI, вы можете использовать файл переменных, указанный с флагом `-l`. Обратите внимание, что bhyve будет записывать изменения, внесённые гостевой системой, в указанный файл переменных. Поэтому убедитесь, что вы предварительно создали отдельную копию шаблонного файла переменных для каждой гостевой системы:

```
# cp /usr/local/share/uefi-firmware/BHYVE_UEFI_VARS.fd /path/to/vm-
image/BHYVE_UEFI_VARS.fd
```

Затем добавьте этот файл переменных в аргументы bhyve:

```
# bhyve -AHP -s 0:0,hostbridge -s 1:0,lpc \
    -s 2:0,virtio-net,tap1 -s 3:0,virtio-blk,./disk.img \
    -s 4:0,ahci-cd,./install.iso -c 4 -m 1024M \
    -l bootrom,/usr/local/share/uefi-firmware/BHYVE_UEFI.fd,/path/to/vm-
image/BHYVE_UEFI_VARS.fd \
    guest
```

Некоторые дистрибутивы Linux требуют использования переменных UEFI

> для хранения пути к их загрузочному файлу UEFI (например, используют `linux64.efi` или `grubx64.efi` вместо `bootx64.efi`). Поэтому рекомендуется использовать файл переменных для виртуальных машин Linux, чтобы избежать необходимости вручную изменять файлы загрузочного раздела.

Для просмотра или изменения содержимого файла переменных используйте efivar(8) с хоста.

sysutils/bhyve-firmware также содержит прошивку с поддержкой CSM для загрузки гостевых систем без поддержки UEFI в режиме устаревшего BIOS:

```
# bhyve -AHP -s 0:0,hostbridge -s 1:0,lpc \
    -s 2:0,virtio-net,tap1 -s 3:0,virtio-blk,./disk.img \
    -s 4:0,ahci-cd,./install.iso -c 4 -m 1024M \
    -l bootrom,/usr/local/share/uefi-firmware/BHYVE_UEFI_CSM.fd \
    guest
```

25.7.5. Графический UEFI фреймбуфер для гостевых систем bhyve

Поддержка микропрограммы UEFI особенно полезна для преимущественно графических гостевых операционных систем, таких как Microsoft Windows®.

Поддержка фреймбуфера UEFI-GOP также может быть включена с помощью флагов `-s 29,fbuf,tcp=0.0.0.0:5900`. Разрешение фреймбуфера можно настроить с помощью параметров `w=800` и `h=600`, а также можно указать bhyve ожидать подключения VNC перед загрузкой гостевой системы, добавив `wait`. Доступ к фреймбуферу возможен с хоста или по сети через протокол VNC. Дополнительно можно добавить `-s 30,xhci,tablet` для точной синхронизации курсора мыши с хостом.

Результирующая команда bhyve будет выглядеть так:

```
# bhyve -AHP -s 0:0,hostbridge -s 31:0,lpc \
    -s 2:0,virtio-net,tap1 -s 3:0,virtio-blk,./disk.img \
    -s 4:0,ahci-cd,./install.iso -c 4 -m 1024M \
    -s 29,fbuf,tcp=0.0.0.0:5900,w=800,h=600,wait \
    -s 30,xhci,tablet \
    -l bootrom,/usr/local/share/uefi-firmware/BHYVE_UEFI.fd \
    guest
```

Заметим, что в режиме эмуляции BIOS обновление кадрового буфера прекращается после передачи управления от микропрограммы гостевой операционной системе.

25.7.6. Создание гостевой системы Microsoft Windows®

Настройка гостевой системы для Windows версий 10 или более ранних может быть выполнена непосредственно с оригинального установочного носителя и является относительно простым процессом. Помимо минимальных требований к ресурсам, для

работы Windows в качестве гостевой системы требуется

- привязка памяти виртуальной машины (флаг `-w`) и
- загрузка с загрузочной памяти (bootrom) UEFI.

Пример загрузки гостевой виртуальной машины с установочным ISO-образом Windows:

```
bhyve \
      -c 2 \
      -s 0,hostbridge \
      -s 3,nvme,windows2016.img \
      -s 4,ahci-cd,install.iso \
      -s 10,virtio-net,tap0 \
      -s 31,lpc \
      -s 30,xhci,tablet \
      -l bootrom,/usr/local/share/uefi-firmware/BHYVE_UEFI.fd \
      -m 8G -H -w \
      windows2016
```

Только один или два виртуальных процессора (VCPU) следует использовать во время установки, но их количество можно увеличить после установки Windows.

Для использования определённого сетевого интерфейса `virtio-net` необходимо установить драйверы VirtIO. Альтернативой является переключение на эмуляцию E1000 (Intel E82545) путем замены `virtio-net` на `e1000` в приведённой выше командной строке. Однако это повлияет на производительность.

Создание гостевой системы Windows 11

Начиная с Windows 11, Microsoft ввела требование к оборудованию в виде модуля TPM 2. bhyve поддерживает передачу аппаратного TPM гостевой системе. Установочный носитель можно модифицировать для отключения соответствующих проверок оборудования. Подробное описание этого процесса доступно по ссылке FreeBSD Wiki.

Изменение установочных носителей Windows и запуск гостевых систем Windows без модуля TPM не поддерживаются производителем. Учитывайте свои задачи и сценарии использования перед применением подобных методов.

25.7.7. Использование ZFS с гостевыми ВМ в bhyve

Если на хост-машине доступен ZFS, использование томов ZFS вместо файлов образов дисков может обеспечить значительное повышение производительности для гостевых ВМ. Том ZFS можно создать следующим образом:

```
# zfs create -V16G -o volmode=dev zroot/linuxdisk0
```

При запуске ВМ укажите том ZFS в качестве диска:

```
# bhyve -A -H -P -s 0:0,hostbridge -s 1:0,lpc -s 2:0,virtio-net,tap0 \
    -s3:0,virtio-blk,/dev/zvol/zroot/linuxdisk0 \
    -l com1,stdio -c 4 -m 1024M linuxguest
```

Если вы используете ZFS как на хосте, так и внутри гостевой системы, учитывайте конкуренцию за память из-за кэширования содержимого виртуальной машины обеими системами. Чтобы уменьшить эту нагрузку, рассмотрите возможность настройки файловых систем ZFS на хосте для использования кэширования только метаданных. Для этого примените следующие параметры к файловым системам ZFS на хосте, заменив `<name>` на имя конкретного zvol-набора данных виртуальной машины.

```
# zfs set primarycache=metadata <name>
```

25.7.8. Создание снимка виртуальной машины

Современные гипервизоры позволяют своим пользователям создавать "снимки" состояния; такой снимок включает в себя диск гостевой системы, содержимое процессора и памяти. Снимок обычно можно сделать независимо от того, работает гостевая система или выключена. Затем можно сбросить и вернуть виртуальную машину в точное состояние на момент создания снимка.

Снимки ZFS

Использование томов ZFS в качестве основного хранилища для виртуальной машины позволяет создавать снимки диска гостевой системы. Например:

```
zfs snapshot zroot/path/to/zvol@snapshot_name
```

Хотя можно создать снимок ZFS тома таким образом, пока гостевая система работает, следует учитывать, что содержимое виртуального диска может находиться в несогласованном состоянии, пока гость активен. Поэтому рекомендуется сначала завершить работу или приостановить гостевую систему перед выполнением этой команды. Функция приостановки гостевой системы не поддерживается по умолчанию и должна быть сначала включена (см. Снимки памяти и процессора)

Откат ZFS zvol к снимку во время использования виртуальной машиной может повредить содержимое файловой системы и вызвать сбой гостевой ОС. Все несохранённые данные в гостевой системе будут потеряны, а изменения, сделанные после последнего снимка, могут быть уничтожены.

Повторный откат может потребоваться после выключения виртуальной машины, чтобы вернуть файловую систему в работоспособное состояние. Это, в свою очередь, окончательно уничтожит все изменения, сделанные

после создания снимка.

Снимки памяти и процессора (экспериментальная функция)

Начиная с FreeBSD 13, bhyve имеет экспериментальную функцию "snapshot" для сохранения состояния памяти и процессора гостевой системы в файл с последующей остановкой виртуальной машины. Гостевая система может быть возобновлена из содержимого файла снимка позже.

Однако эта функция не включена по умолчанию и требует пересборки системы из исходного кода. Подробное описание процесса компиляции ядра с пользовательскими настройками приведено в Сборка из исходного кода.

Функциональность не готова для производственного использования и ограничена работой с определёнными конфигурациями виртуальных машин. Существует несколько ограничений:

- `nvme` и `virtio-blk` бэкенды хранения пока не работают
- Снимки поддерживаются только в случае, если гостевая система использует один тип каждого устройства. То есть, если подключено более одного диска `ahci-hd`, создание снимка завершится ошибкой
- дополнительно, функция может быть достаточно стабильной на процессорах Intel, но, скорее всего, не будет работать на процессорах AMD.

Убедитесь, что каталог /usr/src актуален, прежде чем выполнять следующие шаги. Подробная процедура обновления описана в разделе Обновление исходного кода.

Добавьте следующее в /etc/src.conf:

```
WITH_BHYVE_SNAPHOT=yes
BHYVE_SNAPSHOT=1
MK_BHYVE_SNAPSHOT=yes
```

Если система была частично или полностью пересобрана, рекомендуется выполнить

```
# cd /usr/src
# make cleanworld
```

прежде чем продолжить.

Затем выполните шаги, описанные в подразделе Быстрый старт раздела Обновление FreeBSD из исходного кода , чтобы собрать и установить систему и ядро.

Чтобы проверить успешную активацию функции снимков, введите

```
# bhyvectl --usage
```

и проверить, есть ли в выводе флаг `--suspend`. Если флаг отсутствует, значит функция не активировалась корректно.

Затем вы можете создать снимок состояния и приостановить работающую виртуальную машину по вашему выбору:

```
# bhyvectl --vm=vmname --suspend=/path/to/snapshot/filename
```

Укажите абсолютный путь и имя файла для `--suspend`. В противном случае bhyve запишет данные снимка в тот каталог, из которого был запущен bhyve.

Убедитесь, что записываете данные снимка в защищённый каталог. Сформированный вывод содержит полный дамп памяти гостевой системы и, следовательно, может включать конфиденциальные данные (например, пароли)!

Это создаёт три файла:

- моментальный снимок памяти — назван аналогично параметру `--suspend`
- файл ядра — имя, аналогичное входному параметру `--suspend`, с суффиксом .kern
- метаданные — содержит метаданные о состоянии системы, с именем, оканчивающимся на суффикс .meta

Для восстановления гостевой системы из снимка используйте флаг `-r` с `bhyve`:

```
# bhyve -r /path/to/snapshot/filename
```

Восстановление снимка гостевой системы на архитектуре процессора, отличной от исходной, невозможно. Как правило, попытка восстановления на системе, не идентичной той, на которой был создан снимок, скорее всего, завершится неудачей.

25.7.9. bhyve в клетке

Для повышения безопасности и изоляции виртуальных машин от основной операционной системы можно запускать bhyve в клетке. См. Клетки для подробного описания клеток и их преимуществ в плане безопасности.

Создание клетки для bhyve

Сначала создайте окружение клетки. Если используется файловая система UFS, просто

выполните:

```
# mkdir -p /jails/bhyve
```

Если используется файловая система ZFS, выполните следующие команды:

```
# zfs create zroot/jails
# zfs create zroot/jails/bhyve
```

Затем создайте ZFS zvol для виртуальной машины `bhyvevm0`:

```
# zfs create zroot/vms
# zfs create -V 20G zroot/vms/bhyvevm0
```

Если ZFS не используется, для создания файла образа диска непосредственно в структуре каталогов клетки применяйте следующие команды:

```
# mkdir /jails/bhyve/vms
# truncate -s 20G /jails/bhyve/vms/bhyvevm0
```

Загрузите образ FreeBSD, предпочтительно версии, равной или более старой, чем на хосте, и извлеките его в каталог клетки:

```
# cd /jails
# fetch -o base.txz http://ftp.freebsd.org/pub/FreeBSD/releases/amd64/13.2-
RELEASE/base.txz
# tar -C /jails/bhyve -xvf base.txz
```

Запуск более высокой версии FreeBSD в клетке, чем на хосте, не поддерживается (например, запуск 14.0-RELEASE в jail на хосте с 13.2-RELEASE).

Далее добавьте набор правил devfs в /etc/devfs.rules:

```
[devfsrules_jail_bhyve=100]
add include $devfsrules_hide_all
add include $devfsrules_unhide_login
add path 'urandom' unhide
add path 'random' unhide
add path 'crypto' unhide
add path 'shm' unhide
add path 'zero' unhide
add path 'null' unhide
add path 'mem' unhide
```

```
add path 'vmm' unhide
add path 'vmm/*' unhide
add path 'vmm.io' unhide
add path 'vmm.io/*' unhide
add path 'nmdmbhyve*' unhide
add path 'zvol' unhide
add path 'zvol/zroot' unhide
add path 'zvol/zroot/vms' unhide
add path 'zvol/zroot/vms/bhyvevm0' unhide
add path 'zvol/zroot/vms/bhyvevm1' unhide
add path 'tap10*' unhide
```

Если в вашем файле /etc/devfs.rules уже есть другое правило devfs с числовым идентификатором 100, замените указанный в примере идентификатор на другой, ещё не использованный.

Если не используется файловая система ZFS, пропустите связанные с zvol правила в /etc/devfs.rules:

```
add path 'zvol' unhide
add path 'zvol/zroot' unhide
add path 'zvol/zroot/vms' unhide
add path 'zvol/zroot/vms/bhyvevm0' unhide
add path 'zvol/zroot/vms/bhyvevm1' unhide
```

Эти правила приведут к тому, что bhyve

- создаст виртуальную машину с дисковыми томами `bhyvevm0` и `bhyvevm1`,
- будет использовать сетевые интерфейсы tap с префиксом имени `tap10`. Это означает, что допустимыми именами интерфейсов будут `tap10`, `tap100`, `tap101`, … `tap109`, `tap1000` и так далее.

  Ограничение доступа к подмножеству возможных имён интерфейсов tap предотвратит доступ клетки (и, следовательно, bhyve) к интерфейсам tap хоста и других клеток.

- используйте устройства nmdm с префиксом "bhyve", например /dev/nmdmbhyve0.

Эти правила можно расширять и изменять, используя различные имена гостевых систем и интерфейсов по необходимости.

Если вы планируете использовать bhyve как на хосте, так и в одной или нескольких клеток, помните, что имена интерфейсов tap и nmdm будут работать в общей среде. Например, /dev/nmdmbhyve0 можно использовать либо для bhyve на хосте, либо в клетке.

Перезапустите devfs для применения изменений:

```
# service devfs restart
```

Затем добавьте определение для вашей новой клетке в /etc/jail.conf или /etc/jail.conf.d. Замените номер интерфейса $if и IP-адрес на свои значения.

Пример 33. Использование NAT или маршрутизируемого трафика с межсетевым экраном

```
bhyve {
        $if = 0;
        exec.prestart = "/sbin/ifconfig epair${if} create up";
        exec.prestart += "/sbin/ifconfig epair${if}a up";
        exec.prestart += "/sbin/ifconfig epair${if}a name ${name}0";
        exec.prestart += "/sbin/ifconfig epair${if}b name jail${if}";
        exec.prestart += "/sbin/ifconfig ${name}0 inet 192.168.168.1/27";
        exec.prestart += "/sbin/sysctl net.inet.ip.forwarding=1";

        exec.clean;

        host.hostname = "the-hostname-here";
        vnet;
        vnet.interface = "jail${if}";
        path = "/jails/${name}";
        persist;
        securelevel = 3;
        devfs_ruleset = 100;
        mount.devfs;

        allow.vmm;

        exec.start += "/bin/sh /etc/rc";
        exec.stop = "/bin/sh /etc/rc.shutdown";

        exec.poststop += "/sbin/ifconfig ${name}0 destroy";
}
```

Этот пример предполагает использование межсетевого экрана, такого как `pf` или `ipfw`, для трансляции сетевых адресов (NAT) трафика вашей jail. Подробнее о доступных вариантах реализации этого смотрите в главе Межсетевые экраны.

Пример 34. Использование мостового сетевого подключения

```
bhyve {
        $if = 0;
        exec.prestart = "/sbin/ifconfig epair${if} create up";
        exec.prestart += "/sbin/ifconfig epair${if}a up";
        exec.prestart += "/sbin/ifconfig epair${if}a name ${name}0";
        exec.prestart += "/sbin/ifconfig epair${if}b name jail${if}";
```

```
        exec.prestart += "/sbin/ifconfig bridge0 addm ${name}0";
        exec.prestart += "/sbin/sysctl net.inet.ip.forwarding=1";

        exec.clean;

        host.hostname = "the-hostname-here";
        vnet;
        vnet.interface = "jail${if}";
        path = "/jails/${name}";
        persist;
        securelevel = 3;
        devfs_ruleset = 100;
        mount.devfs;

        allow.vmm;

        exec.start += "/bin/sh /etc/rc";
        exec.stop = "/bin/sh /etc/rc.shutdown";

        exec.poststop += "/sbin/ifconfig ${name}0 destroy";
}
```

Если вы ранее заменили идентификатор набора правил devfs 100 в /etc/devfs.rules на свой уникальный номер, не забудьте также заменить числовой идентификатор в вашем jails.conf.

Настройка клетки

Для первого запуска клетки и выполнения дополнительных настроек введите:

```
# cp /etc/resolv.conf /jails/bhyve/etc
# service jail onestart bhyve
# jexec bhyve
# sysrc ifconfig_jail0="inet 192.168.168.2/27"
# sysrc defaultrouter="192.168.168.1"
# sysrc sendmail_enable=NONE
# sysrc cloned_interfaces="tap100"
# exit
```

Перезапустите и включите клетку:

```
# sysrc jail_enable=YES
# service jail restart bhyve
```

После этого вы можете создать виртуальную машину внутри клетки. Для гостевой системы FreeBSD сначала загрузите установочный ISO-образ:

```
# jexec bhyve
# cd /vms
# fetch -o freebsd.iso https://download.freebsd.org/releases/ISO-IMAGES/14.0/FreeBSD-
14.0-RELEASE-amd64-bootonly.iso
```

Создание виртуальной машины внутри клетки

Чтобы создать виртуальную машину, сначала инициализируйте её с помощью `bhyvectl`:

```
# jexec bhyve
# bhyvectl --create --vm=bhyvevm0
```

Создание гостевой системы с помощью `bhyvectl` может потребоваться при запуске виртуальной машины из клетки. Пропуск этого шага может привести к следующему сообщению об ошибке при запуске `bhyve`:

```
vm_open: vm-name could not be opened. No such file or directory
```

Наконец, используйте предпочитаемый способ запуска гостевой системы.

Пример 35. Запуск с `vmrun.sh` и ZFS

Используя `vmrun.sh` на файловых системах ZFS:

```
# jexec bhyve
# sh /usr/share/examples/bhyve/vmrun.sh -c 1 -m 1024M \
     -t tap100 -d /dev/zvol/zroot/vms/bhyvevm0 -i -I /vms/FreeBSD-14.0-RELEASE-
amd64-bootonly.iso bhyvevm0
```

Пример 36. Запуск с `vmrun.sh` и UFS

Используя `vmrun.sh` на файловой системе UFS:

```
# jexec bhyve
# sh /usr/share/examples/bhyve/vmrun.sh -c 1 -m 1024M \
     -t tap100 -d /vms/bhyvevm0 -i -I /vms/FreeBSD-14.0-RELEASE-amd64-bootonly.iso
bhyvevm0
```

Пример 37. Запуск bhyve для гостевой системы UEFI с ZFS

Если же вы хотите использовать гостевую систему с UEFI, не забудьте сначала установить необходимый пакет с микропрограммой sysutils/bhyve-firmware в клетку:

```
# pkg -j bhyve install bhyve-firmware
```

Затем используйте bhyve напрямую:

```
# bhyve -A -c 4 -D -H -m 2G \
        -s 0,hostbridge \
        -s 1,lpc \
        -s 2,virtio-net,tap100 \
        -s 3,virtio-blk,/dev/zvol/zroot/vms/bhyvevm0 \
    -s 4,ahci-cd,/vms/FreeBSD-14.0-RELEASE-amd64-bootonly.iso \
        -s 31,fbuf,tcp=127.0.0.1:5900,w=1024,h=800,tablet \
        -l bootrom,/usr/local/share/uefi-firmware/BHYVE_UEFI.fd \
        -l com1,/dev/nmdbbhyve0A \
        bhyvevm0
```

Это позволит вам подключиться к вашей виртуальной машине bhyvevm0 через VNC, а также через последовательную консоль по адресу /dev/nmdbbhyve0B.

25.7.10. Консоли виртуальной машины

Преимуществом является обёртывание консоли bhyve в инструмент управления сеансами, например sysutils/tmux или sysutils/screen, чтобы иметь возможность отключаться и подключаться к консоли. Также можно сделать консоль bhyve нуль-модемным устройством, доступным через cu. Для этого загрузите модуль ядра nmdm и замените `-l com1,stdio` на `-l com1,/dev/nmdm0A`. Устройства /dev/nmdm создаются автоматически по мере необходимости, где каждое представляет собой пару, соответствующую двум концам нуль-модемного кабеля (/dev/nmdm0A и /dev/nmdm0B). Подробнее см. в nmdm(4).

```
# kldload nmdm
# bhyve -A -H -P -s 0:0,hostbridge -s 1:0,lpc -s 2:0,virtio-net,tap0 -s 3:0,virtio-
blk,./linux.img \
    -l com1,/dev/nmdm0A -c 4 -m 1024M linuxguest
# cu -l /dev/nmdm0B
Connected

Ubuntu 13.10 handbook ttyS0

handbook login:
```

Чтобы отключиться от консоли, введите перевод строки (т.е. нажмите RETURN), затем тильду (~) и точку (.). Учтите, что разрывается только соединение, а сеанс входа в систему остаётся активным. Таким образом, другой пользователь, подключившись к той же консоли, может воспользоваться активными сеансами без необходимости аутентификации. По соображениям безопасности рекомендуется выйти из системы перед отключением.

Число в пути устройства nmdm должно быть уникальным для каждой виртуальной

машины и не должно использоваться другими процессами до запуска bhyve. Это число можно выбирать произвольно, оно не обязательно должно быть взято из последовательного ряда чисел. Пара узлов устройств (например, /dev/nmdm0a и /dev/nmdm0b) создаётся динамически при подключении консоли bhyve и уничтожается при её завершении. Учитывайте это при создании скриптов для запуска виртуальных машин: необходимо убедиться, что всем виртуальным машинам назначены уникальные устройства nmdm.

25.7.11. Управление виртуальными машинами

Для каждой виртуальной машины создаётся узел устройства в /dev/vmm. Это позволяет администратору легко просматривать список работающих виртуальных машин:

```
# ls -al /dev/vmm
total 1
dr-xr-xr-x   2 root  wheel    512 Mar 17 12:19 ./
dr-xr-xr-x  14 root  wheel    512 Mar 17 06:38 ../
crw-------   1 root  wheel  0x1a2 Mar 17 12:20 guestname
crw-------   1 root  wheel  0x19f Mar 17 12:19 linuxguest
crw-------   1 root  wheel  0x1a1 Mar 17 12:19 otherguest
```

Указанную виртуальную машину можно уничтожить с помощью `bhyvectl`:

```
# bhyvectl --destroy --vm=guestname
```

Уничтожение виртуальной машины таким способом означает её немедленное завершение. Все несохранённые данные будут потеряны, открытые файлы и файловые системы могут быть повреждены. Для корректного завершения работы виртуальной машины отправьте сигнал `TERM` её процессу bhyve. Это инициирует событие ACPI завершения работы для гостевой системы:

```
# ps ax | grep bhyve
17424  -  SC      56:48.27 bhyve: guestvm (bhyve)
# kill 17424
```

25.7.12. Инструменты и утилиты

В портах доступно множество утилит и приложений, которые помогают упростить настройку и управление виртуальными машинами bhyve:

Таблица 36. Менеджеры bhyve

Имя	Лицензия	Пакет	Documentation
vm-bhyve	BSD-2	sysutils/vm-bhyve	Документация
CBSD	BSD-2	sysutils/cbsd	Документация

Имя	Лицензия	Пакет	Documentation
Virt-Manager	LGPL-3	deskutils/virt-manager	Документация
Bhyve RC Script	Неизвестно	sysutils/bhyve-rc	Документация
bmd	BSD-2	sysutils/bmd	Документация
vmstated	BSD-2	sysutils/vmstated	Документация

25.7.13. Постоянная конфигурация

Для настройки системы на автоматический запуск гостевых систем bhyve при загрузке необходимо внести изменения в некоторые конфигурационные файлы.

1. /etc/sysctl.conf

   При использовании интерфейсов tap в качестве сетевого бэкенда необходимо либо вручную переводить каждый используемый интерфейс tap в состояние UP, либо просто установить следующий параметр sysctl:

   ```
   net.link.tap.up_on_open=1
   ```

2. /etc/rc.conf

   Чтобы подключить устройство tap вашей виртуальной машины к сети через bridge, необходимо сохранить настройки устройства в /etc/rc.conf. Дополнительно можно загрузить необходимые модули ядра `vmm` для bhyve и `nmdm` для устройств nmdm через переменную конфигурации `kld_list`. При настройке `ifconfig_bridge0` убедитесь, что заменили `<ipaddr>/<netmask>` на реальный IP-адрес вашего физического интерфейса (igb0 в данном примере) и удалите IP-настройки с вашего физического устройства.

   ```
   # sysrc cloned_interfaces+="bridge0 tap0"
   # sysrc ifconfig_bridge0="inet <ipaddr>/<netmask> addm igb0 addm tap0"
   # sysrc kld_list+="nmdm vmm"
   # sysrc ifconfig_igb0="up"
   ```

Пример 38. Установка IP для устройства bridge

Для хоста с интерфейсом *igb0*, подключённым к сети с IP `10.10.10.1` и маской подсети `255.255.255.0`, используйте следующие команды:

```
# sysrc ifconfig_igb0="up"
# sysrc ifconfig_bridge0="inet 10.10.10.1/24 addm igb0 addm tap0"
# sysrc kld_list+="nmdm vmm"
# sysrc cloned_interfaces+="bridge0 tap0"
```

Изменение конфигурации IP-адреса системы может заблокировать ваш доступ, если вы выполняете эти команды, подключившись удалённо (например, через SSH)! Примите меры предосторожности, чтобы сохранить доступ к системе, или вносите эти изменения, работая в локальной терминальной сессии.

25.8. FreeBSD в качестве хоста Xen™

Xen — это гипервизор класса 1 с лицензией GPLv2 для архитектур Intel® и ARM®. Начиная с FreeBSD 8.0, система включает поддержку непривилегированных доменов (виртуальных машин) DomU и Amazon EC2 для i386™ и AMD® 64-битных процессоров, а также поддержку управляющего домена (хоста) Dom0 в FreeBSD 11.0. Поддержка паравиртуализованных (PV) доменов была удалена в FreeBSD 11 в пользу аппаратно-виртуализованных (HVM) доменов, которые обеспечивают лучшую производительность.

Xen™ — это гипервизор первого типа (bare-metal), что означает, что он загружается первым после BIOS. Затем запускается специальная привилегированная гостевая система, называемая Domain-0 (сокращенно `Dom0`). Dom0 использует свои особые привилегии для прямого доступа к физическому оборудованию, что делает его высокопроизводительным решением. Она может напрямую обращаться к контроллерам дисков и сетевым адаптерам. Инструменты управления Xen™ для управления гипервизором Xen™ также используются Dom0 для создания, перечисления и удаления виртуальных машин. Dom0 предоставляет виртуальные диски и сетевые ресурсы для непривилегированных доменов, часто называемых `DomU`. Xen™ Dom0 можно сравнить с сервисной консолью других гипервизоров, в то время как DomU — это среда, в которой запускаются отдельные гостевые виртуальные машины.

Xen™ может переносить виртуальные машины между разными серверами Xen™. Если два хоста Xen используют общее хранилище данных, миграцию можно выполнить без предварительного выключения виртуальной машины. Вместо этого миграция выполняется в реальном времени, пока DomU работает, и нет необходимости перезапускать его или планировать простой. Это полезно в сценариях обслуживания или обновления, чтобы гарантировать непрерывность предоставления услуг DomU. Множество других возможностей Xen™ перечислены на странице обзора Xen Wiki. Обратите внимание, что не все функции пока поддерживаются в FreeBSD.

25.8.1. Требования к оборудованию для Xen™ Dom0

Для запуска гипервизора Xen™ на хосте требуется определённая аппаратная функциональность. Для работы FreeBSD в качестве хоста Xen (Dom0) необходимы поддержка Intel Extended Page Tables (EPT) или AMD Nested Page Tables (NPT), а также поддержка Input/Output Memory Management Unit (IOMMU) в процессоре хоста.

Для запуска FreeBSD 13 в качестве Xen™ Dom0 система должна быть загружена в режиме legacy boot (BIOS). FreeBSD 14 и новее поддерживают загрузку в качестве Xen™ Dom0 как в режиме BIOS, так и в режиме UEFI.

25.8.2. Настройка управляющего домена Xen™ Dom0

Пользователям следует установить пакеты emulators/xen-kernel и sysutils/xen-tools, основанные на Xen™ 4.18.

После установки пакетов Xen необходимо отредактировать конфигурационные файлы, чтобы подготовить хост для интеграции с Dom0. В файл /etc/sysctl.conf следует добавить запись, отключающую ограничение на количество страниц памяти, которые могут быть закреплены. В противном случае виртуальные машины DomU с повышенными требованиями к памяти не смогут запуститься.

```
# echo 'vm.max_wired=-1' >> /etc/sysctl.conf
```

Ещё одна настройка, связанная с памятью, включает изменение файла /etc/login.conf, где необходимо установить параметр `memorylocked` в значение `unlimited`. В противном случае создание доменов DomU может завершиться ошибкой `Cannot allocate memory`. После внесения изменений в /etc/login.conf выполните команду `cap_mkdb` для обновления базы данных capability. Подробности см. в разделе Ограничения ресурсов.

```
# sed -i '' -e 's/memorylocked=64K/memorylocked=unlimited/' /etc/login.conf
# cap_mkdb /etc/login.conf
```

Добавьте запись для консоли Xen™ в /etc/ttys:

```
# echo 'xc0     "/usr/libexec/getty Pc"         xterm   onifconsole  secure' >>
/etc/ttys
```

Выбор ядра Xen™ в /boot/loader.conf активирует Dom0. Xen™ также требует ресурсы, такие как процессор и память, от основной машины для себя и других доменов DomU. Количество процессоров и памяти зависит от индивидуальных требований и возможностей оборудования. В этом примере для Dom0 доступно 8 ГБ памяти и 4 виртуальных процессора. Также активирована последовательная консоль и определены параметры журналирования.

Следующая команда используется для пакетов Xen 4.7:

```
# echo 'hw.pci.mcfg=0' >> /boot/loader.conf
# echo 'if_tap_load="YES"' >> /boot/loader.conf
# echo 'xen_kernel="/boot/xen"' >> /boot/loader.conf
# echo 'xen_cmdline="dom0_mem=8192M dom0_max_vcpus=4 dom0pvh=1 console=com1,vga
com1=115200,8n1 guest_loglvl=all loglvl=all"' >> /boot/loader.conf
```

Для версий Xen 4.11 и выше вместо этого следует использовать следующую команду:

```
# echo 'if_tap_load="YES"' >> /boot/loader.conf
# echo 'xen_kernel="/boot/xen"' >> /boot/loader.conf
```

```
# echo 'xen_cmdline="dom0_mem=8192M dom0_max_vcpus=4 dom0=pvh console=com1,vga
com1=115200,8n1 guest_loglvl=all loglvl=all"' >> /boot/loader.conf
```

Файлы журналов, создаваемые Xen™ для DomU ВМ, хранятся в /var/log/xen. В случае возникновения проблем обязательно проверьте содержимое этого каталога.

Активируйте сервис xencommons при загрузке системы:

```
# sysrc xencommons_enable=yes
```

Этих настроек достаточно для запуска системы с поддержкой Dom0. Однако в ней отсутствует сетевая функциональность для машин DomU. Чтобы это исправить, определите мостовой интерфейс с основной сетевой картой системы, который DomU-виртуальные машины смогут использовать для подключения к сети. Замените *em0* на имя сетевого интерфейса хоста.

```
# sysrc cloned_interfaces="bridge0"
# sysrc ifconfig_bridge0="addm em0 SYNCDHCP"
# sysrc ifconfig_em0="up"
```

Перезагрузите хост для загрузки ядра Xen™ и запуска Dom0.

```
# shutdown -r now
```

После успешной загрузки ядра Xen™ и повторного входа в систему, инструмент управления Xen™ `xl` используется для отображения информации о доменах.

```
# xl list
Name                                        ID   Mem VCPUs      State   Time(s)
Domain-0                                     0  8192     4     r-----     962.0
```

Вывод подтверждает, что Dom0 (называемый `Domain-0`) имеет идентификатор `0` и работает. Также у него есть память и виртуальные CPU, которые были определены ранее в /boot/loader.conf. Дополнительную информацию можно найти в документации Xen™. Теперь можно создавать гостевые ВМ DomU.

25.8.3. Конфигурация гостевой виртуальной машины Xen™ DomU

Непривилегированные домены состоят из конфигурационного файла и виртуальных или физических жёстких дисков. Виртуальное дисковое хранилище для DomU может быть файлами, созданными с помощью truncate(1), или томами ZFS, как описано в “Создание и уничтожение томов”. В этом примере используется том объёмом 20 ГБ. Виртуальная машина создаётся с томом ZFS, образом ISO FreeBSD, 1 ГБ оперативной памяти и двумя

виртуальными процессорами. Файл установки ISO извлекается с помощью fetch(1) и сохраняется локально в файле с именем freebsd.iso.

```
# fetch https://download.freebsd.org/releases/ISO-IMAGES/14.0/FreeBSD-14.0-RELEASE-
amd64-bootonly.iso -o freebsd.iso
```

Создаётся том ZFS объёмом 20 ГБ с именем xendisk0, который будет использоваться в качестве дискового пространства для виртуальной машины.

```
# zfs create -V20G -o volmode=dev zroot/xendisk0
```

Новый гостевая ВМ DomU определяется в файле. Некоторые конкретные определения, такие как имя, раскладка клавиатуры и детали подключения VNC, также задаются. Следующий файл freebsd.cfg содержит минимальную конфигурацию DomU для этого примера:

```
# cat freebsd.cfg
builder = "hvm" ①
name = "freebsd" ②
memory = 1024 ③
vcpus = 2 ④
vif = [ 'mac=00:16:3E:74:34:32,bridge=bridge0' ] ⑤
disk = [
'/dev/zvol/tank/xendisk0,raw,hda,rw', ⑥
'/root/freebsd.iso,raw,hdc:cdrom,r' ⑦
  ]
vnc = 1 ⑧
vnclisten = "0.0.0.0"
serial = "pty"
usbdevice = "tablet"
```

Эти строки объясняются более подробно:

① Это определяет, какой тип виртуализации использовать. `hvm` означает аппаратную виртуализацию или аппаратную виртуальную машину. Гостевые операционные системы могут работать без изменений на процессорах с расширениями виртуализации, обеспечивая производительность почти такую же, как и на физическом оборудовании. `generic` — значение по умолчанию, которое создаёт PV-домен.

② Имя этой виртуальной машины для отличия от других, работающих на том же Dom0. Обязательно.

③ Количество оперативной памяти в мегабайтах, которое будет доступно виртуальной машине. Этот объём вычитается из общего объёма доступной памяти гипервизора, а не из памяти Dom0.

④ Количество виртуальных CPU, доступных гостевой ВМ. Для наилучшей производительности не следует создавать гостевые системы с количеством виртуальных CPU, превышающим число физических CPU на хосте.

⑤ Виртуальный сетевой адаптер. Это мост, подключённый к сетевому интерфейсу хоста. Параметр `mac` — это MAC-адрес, установленный на виртуальном сетевом интерфейсе. Этот параметр необязателен: если MAC-адрес не указан, Xen™ сгенерирует случайный.

⑥ Полный путь к диску, файлу или тому ZFS для дискового хранилища этой ВМ. Параметры и определения нескольких дисков разделяются запятыми.

⑦ Определяет загрузочный носитель, с которого устанавливается исходная операционная система. В данном примере это загруженный ранее образ ISO. О других типах устройств и настраиваемых параметрах см. документацию Xen™.

⑧ Параметры, управляющие подключением VNC к последовательной консоли DomU. В порядке перечисления: поддержка VNC, определение IP-адреса для прослушивания, файл устройства последовательной консоли и метод ввода для точного позиционирования мыши и других методов ввода. `keymap` определяет, какую раскладку клавиатуры использовать, по умолчанию установлено значение `english`.

После создания файла со всеми необходимыми параметрами DomU создаётся путём передачи его в `xl create` в качестве аргумента.

```
# xl create freebsd.cfg
```

Каждый раз при перезапуске Dom0 конфигурационный файл необходимо снова передавать в `xl create`, чтобы воссоздать DomU. По умолчанию после перезагрузки создаётся только Dom0, но не отдельные виртуальные машины. Виртуальные машины могут продолжить работу с момента остановки, так как операционная система сохраняется на виртуальном диске. Конфигурация виртуальной машины может со временем меняться (например, при добавлении памяти). Конфигурационные файлы виртуальных машин необходимо надлежащим образом резервировать и хранить в доступном месте, чтобы иметь возможность воссоздать гостевую виртуальную машину при необходимости.

Вывод команды `xl list` подтверждает, что DomU был создан.

```
# xl list
Name                                        ID   Mem VCPUs      State   Time(s)
Domain-0                                     0  8192     4     r-----  1653.4
freebsd                                      1  1024     1     -b----   663.9
```

Для начала установки базовой операционной системы запустите клиент VNC, указав основной сетевой адрес хоста или IP-адрес, определённый в строке `vnclisten` файла freebsd.cfg. После установки операционной системы завершите работу DomU и отключите VNC-клиент. Отредактируйте файл freebsd.cfg, удалив строку с определением `cdrom` или закомментировав её, поставив символ `#` в начале строки. Чтобы загрузить новую конфигурацию, необходимо удалить старый DomU командой `xl destroy`, передав в качестве параметра имя или идентификатор. После этого воссоздайте его, используя изменённый файл **freebsd.cfg**.

```
# xl destroy freebsd
# xl create freebsd.cfg
```

Затем к машине можно снова получить доступ с помощью VNC-клиента. На этот раз она загрузится с виртуального диска, на который была установлена операционная система, и её можно будет использовать как виртуальную машину.

25.8.4. Устранение неполадок

Этот раздел содержит основную информацию, которая поможет в устранении проблем, возникающих при использовании FreeBSD в качестве хоста или гостевой системы Xen™.

Устранение неполадок при загрузке хоста

Обратите внимание, что следующие советы по устранению неполадок предназначены для Xen™ 4.11 или новее. Если вы всё ещё используете Xen™ 4.7 и сталкиваетесь с проблемами, рассмотрите возможность перехода на более новую версию Xen™.

Для устранения проблем с загрузкой хоста вам, скорее всего, понадобится последовательный кабель или отладочный USB-кабель. Подробный вывод загрузки Xen™ можно получить, добавив параметры к опции `xen_cmdline` в файле loader.conf. Несколько соответствующих отладочных параметров:

- `iommu=debug`: может использоваться для вывода дополнительной диагностической информации о iommu.
- `dom0=verbose`: может использоваться для вывода дополнительной диагностической информации о процессе сборки dom0.
- `sync_console`: флаг для принудительного синхронного вывода на консоль. Полезен при отладке, чтобы избежать потери сообщений из-за ограничения скорости. Никогда не используйте эту опцию в рабочих средах, так как она может позволить злоумышленникам выполнять DoS-атаки на Xen™ через консоль.

FreeBSD также следует загружать в подробном режиме для выявления возможных проблем. Чтобы активировать подробную загрузку, выполните следующую команду:

```
# echo 'boot_verbose="YES"' >> /boot/loader.conf
```

Если ни один из этих вариантов не помог решить проблему, отправьте журнал загрузки по последовательному соединению по адресам freebsd-xen@FreeBSD.org и xen-devel@lists.xenproject.org для дальнейшего анализа.

Устранение неполадок при создании гостевой системы

Проблемы также могут возникать при создании гостевых систем. В следующем разделе представлены рекомендации для диагностики подобных проблем.

Наиболее распространённой причиной сбоев при создании гостевой системы является

команда xl, которая выдаёт ошибку и завершается с кодом возврата, отличным от 0. Если предоставленного сообщения об ошибке недостаточно для определения проблемы, можно получить более подробный вывод от xl, многократно используя опцию v.

```
# xl -vvv create freebsd.cfg
Parsing config from freebsd.cfg
libxl: debug: libxl_create.c:1693:do_domain_create: Domain 0:ao 0x800d750a0: create:
how=0x0 callback=0x0 poller=0x800d6f0f0
libxl: debug: libxl_device.c:397:libxl__device_disk_set_backend: Disk vdev=xvda
spec.backend=unknown
libxl: debug: libxl_device.c:432:libxl__device_disk_set_backend: Disk vdev=xvda, using
backend phy
libxl: debug: libxl_create.c:1018:initiate_domain_create: Domain 1:running bootloader
libxl: debug: libxl_bootloader.c:328:libxl__bootloader_run: Domain 1:not a PV/PVH
domain, skipping bootloader
libxl: debug: libxl_event.c:689:libxl__ev_xswatch_deregister: watch w=0x800d96b98:
deregister unregistered
domainbuilder: detail: xc_dom_allocate: cmdline="", features=""
domainbuilder: detail: xc_dom_kernel_file: filename
="/usr/local/lib/xen/boot/hvmloader"
domainbuilder: detail: xc_dom_malloc_filemap    : 326 kB
libxl: debug: libxl_dom.c:988:libxl__load_hvm_firmware_module: Loading BIOS:
/usr/local/share/seabios/bios.bin
...
```

Если подробный вывод не помог диагностировать проблему, также существуют журналы QEMU и инструментов Xen™ в /var/log/xen. Обратите внимание, что имя домена добавляется к имени журнала, поэтому если домен называется freebsd, вы найдете файлы /var/log/xen/xl-freebsd.log и, вероятно, /var/log/xen/qemu-dm-freebsd.log. Оба файла журналов могут содержать полезную информацию для отладки. Если ничто из этого не помогло решить проблему, пожалуйста, отправьте описание возникшей проблемы и как можно больше информации на адреса freebsd-xen@FreeBSD.org и xen-devel@lists.xenproject.org, чтобы получить помощь.

Глава 26. Локализация - использование и настройка i18n/L10n

26.1. Обзор

FreeBSD — это распределённый проект с пользователями и участниками по всему миру. Поэтому FreeBSD поддерживает локализацию на множество языков, позволяя пользователям просматривать, вводить и обрабатывать данные на языках, отличных от английского. Доступны многие основные языки, включая, но не ограничиваясь: китайский, немецкий, японский, корейский, французский, русский и вьетнамский.

Термин "internationalization" был сокращён до i18n, где число 18 обозначает количество букв между первой и последней буквами слова "internationalization". L10n использует ту же схему сокращения, но от слова "localization". Методы, протоколы и приложения i18n/L10n позволяют пользователям работать на предпочитаемых ими языках.

Эта глава рассказывает о возможностях интернационализации и локализации в FreeBSD.

Прочтите эту главу, чтобы узнать:

- Как формируются названия локалей.
- Как установить локаль для входной оболочки.
- Как настроить консоль для неанглийских языков.
- Как настроить Xorg для разных языков.
- Как найти приложения, соответствующие стандарту i18n.
- Где найти дополнительную информацию по настройке конкретных языков.

Прежде чем читать эту главу, необходимо:

- Знать, как устанавливать дополнительные сторонние приложения.

26.2. Использование локализации

Настройки локализации основаны на трёх компонентах: коде языка, коде страны и кодировке. Названия локалей формируются из этих частей следующим образом:

```
LanguageCode_CountryCode.Encoding
```

ЯзыковойКод и *КодСтраны* используются для определения страны и конкретного варианта языка. Распространённые языковые коды и коды стран содержит несколько примеров *ЯзыковойКод_КодСтраны*:

Таблица 37. Часто используемые коды языков и стран

КодЯзыка_КодСтраны	Описание
en_US	Английский, Соединённые Штаты
ru_RU	Русский, Россия
zh_TW	Традиционный китайский, Тайвань

Полный список доступных локалей можно получить, набрав:

```
% locale -a | more
```

Для определения текущих настроек локали:

```
% locale
```

Языковые наборы символов, такие как ISO8859-1, ISO8859-15, KOI8-R и CP437, описаны в multibyte(3). Актуальный список наборов символов доступен в Реестре IANA.

Некоторые языки, такие как китайский или японский, не могут быть представлены с использованием символов ASCII и требуют расширенной кодировки с использованием широких или многобайтовых символов. Примерами широких или многобайтовых кодировок являются EUC и Big5. Старые приложения могут ошибочно принимать эти кодировки за управляющие символы, в то время как новые приложения обычно распознают их правильно. В зависимости от реализации, пользователям может потребоваться скомпилировать приложение с поддержкой широких или многобайтовых символов или правильно его настроить.

FreeBSD использует совместимые с Xorg кодировки локалей.

Остальная часть этого раздела описывает различные методы настройки локали в системе FreeBSD. Следующий раздел будет посвящен вопросам поиска и компиляции приложений с поддержкой i18n.

26.2.1. Установка локали для входной оболочки

Настройки локали настраиваются либо в файле ~/.login_conf пользователя, либо в стартовом файле его оболочки: ~/.profile, ~/.bashrc или ~/.cshrc.

Должны быть установлены две переменные окружения:

- `LANG`, который устанавливает локаль
- `MM_CHARSET`, который устанавливает MIME-кодировку, используемую приложениями

В дополнение к настройкам оболочки пользователя, эти переменные также должны быть установлены для конкретной конфигурации приложений и конфигурации Xorg.

Доступны два метода для задания необходимых переменных: метод класса входа, который является рекомендуемым, и метод файла запуска. В следующих двух разделах показано, как

использовать оба метода.

Метод классов входа

Этот первый метод является рекомендуемым, так как он устанавливает необходимые переменные окружения для названия локали и MIME-кодировок символов для всех возможных командных оболочек. Данную настройку может выполнить как каждый пользователь в отдельности, так и суперпользователь для всех пользователей сразу.

Этот минимальный пример устанавливает обе переменные для кодировки Latin-1 в файле .login_conf домашнего каталога отдельного пользователя:

```
me:\
    :charset=ISO-8859-1:\
    :lang=de_DE.ISO8859-1:
```

Вот пример файла ~/.login_conf пользователя, который устанавливает переменные для традиционного китайского в кодировке BIG-5. Требуется больше переменных, так как некоторые приложения некорректно обрабатывают локали для китайского, японского и корейского языков:

```
#Users who do not wish to use monetary units or time formats
#of Taiwan can manually change each variable
me:\
    :lang=zh_TW.Big5:\

:setenv=LC_ALL=zh_TW.Big5,LC_COLLATE=zh_TW.Big5,LC_CTYPE=zh_TW.Big5,LC_MESSAGES=zh_TW.
Big5,LC_MONETARY=zh_TW.Big5,LC_NUMERIC=zh_TW.Big5,LC_TIME=zh_TW.Big5:\
    :charset=big5:\
    :xmodifiers="@im=gcin": #Set gcin as the XIM Input Server
```

Или суперпользователь может настроить локализацию для всех пользователей системы. Следующие переменные в /etc/login.conf используются для установки локали и набора символов MIME:

```
language_name|Account Type Description:\
    :charset=MIME_charset:\
    :lang=locale_name:\
    :tc=default:
```

Итак, предыдущий пример с Latin-1 будет выглядеть так:

```
german|German Users Accounts:\
    :charset=ISO-8859-1:\
    :lang=de_DE.ISO8859-1:\
    :tc=default:
```

Подробнее об этих переменных смотрите в login.conf(5). Обратите внимание, что там уже есть предопределённый класс *russian*.

При редактировании файла /etc/login.conf не забудьте выполнить следующую команду для обновления базы данных возможностей:

```
# cap_mkdb /etc/login.conf
```

Для конечного пользователя команда `cap_mkdb` должна быть выполнена для его файла ~/.login_conf, чтобы изменения вступили в силу.

Утилиты, изменяющие классы входа в систему

В дополнение к ручному редактированию /etc/login.conf доступно несколько утилит для установки локали для вновь создаваемых пользователей.

При использовании `vipw` для добавления новых пользователей укажите *language*, чтобы задать локаль:

```
user:password:1111:11:language:0:0:User Name:/home/user:/bin/sh
```

При использовании `adduser` для добавления новых пользователей язык по умолчанию может быть предварительно настроен для всех новых пользователей или указан для отдельного пользователя.

Если все новые пользователи используют один и тот же язык, укажите `defaultclass=language` в /etc/adduser.conf.

Чтобы переопределить этот параметр при создании пользователя, введите требуемую локаль в этом запросе:

```
Enter login class: default []:
```

или указать локаль для установки при вызове `adduser`:

```
# adduser -class language
```

Если `pw` используется для добавления новых пользователей, укажите локаль следующим образом:

```
# pw useradd user_name -L language
```

Для изменения класса входа существующего пользователя можно использовать `chpass`. Запустите её от имени суперпользователя и укажите имя пользователя для редактирования

в качестве аргумента.

```
# chpass user_name
```

Метод файла запуска оболочки

Этот второй метод не рекомендуется, так как каждая используемая оболочка требует ручной настройки, причём у каждой оболочки свой файл конфигурации и разный синтаксис. Например, чтобы установить немецкий язык для оболочки `sh`, можно добавить следующие строки в ~/.profile для настройки оболочки только этого пользователя. Эти же строки можно добавить в /etc/profile или /usr/share/skel/dot.profile, чтобы настроить эту оболочку для всех пользователей:

```
LANG=de_DE.ISO8859-1; export LANG
MM_CHARSET=ISO-8859-1; export MM_CHARSET
```

Однако имя файла конфигурации и используемый синтаксис отличаются для оболочки `csh`. Вот эквивалентные настройки для ~/.login, /etc/csh.login или /usr/share/skel/dot.login:

```
setenv LANG de_DE.ISO8859-1
setenv MM_CHARSET ISO-8859-1
```

Чтобы усложнить ситуацию, синтаксис, необходимый для настройки Xorg в ~/.xinitrc, также зависит от оболочки. Первый пример приведён для оболочки `sh`, а второй — для `csh`:

```
LANG=de_DE.ISO8859-1; export LANG
```

```
setenv LANG de_DE.ISO8859-1
```

26.2.2. Настройка консоли

Доступно несколько локализованных шрифтов для консоли. Чтобы просмотреть список доступных шрифтов, введите `ls /usr/share/syscons/fonts`. Для настройки шрифта консоли укажите *font_name* без суффикса .fnt в файле /etc/rc.conf:

```
font8x16=font_name
font8x14=font_name
font8x8=font_name
```

Раскладки клавиатуры (keymap) и карты экрана (screenmap) можно настроить добавлением следующих команд в /etc/rc.conf:

```
scrnmap=screenmap_name
keymap=keymap_name
keychange="fkey_number sequence"
```

Чтобы просмотреть список доступных карт экрана, введите `ls /usr/share/syscons/scrnmaps`. Не указывайте суффикс .scm при указании *screenmap_name*. Обычно требуется карта экрана с соответствующим сопоставленным шрифтом в качестве обходного решения для расширения 8-го бита до 9-го в матрице символов шрифта адаптера VGA, чтобы буквы были выведены из области псевдографики, если экранный шрифт использует столбец 8-го бита.

Чтобы просмотреть список доступных раскладок клавиатуры, введите `ls /usr/share/syscons/keymaps`. При указании *keymap_name* не включайте суффикс .kbd. Для проверки раскладок без перезагрузки используйте kbdmap(1).

Запись `keychange` обычно требуется для программирования функциональных клавиш в соответствии с выбранным типом терминала, так как последовательности функциональных клавиш нельзя определить в keymap.

Затем установите правильный тип консольного терминала в /etc/ttys для всех записей виртуальных терминалов. В Определённые типы терминалов для наборов символов приведены доступные типы терминалов:

Таблица 38. Определённые типы терминалов для наборов символов

Набор символов	Тип терминала
ISO8859-1 или ISO8859-15	`cons25l1`
ISO8859-2	`cons25l2`
ISO8859-7	`cons25l7`
KOI8-R	`cons25r`
KOI8-U	`cons25u`
CP437 (VGA default)	`cons25`
US-ASCII	`cons25w`

Для языков с широкими или многобайтовыми символами установите консоль для этого языка из коллекции портов FreeBSD. Доступные порты перечислены в Доступные консоли из коллекции портов. После установки обратитесь к pkg-message или man-страницам порта для получения инструкций по настройке и использованию.

Таблица 39. Доступная консоль из коллекции портов

Язык	Расположение порта
Традиционный китайский (BIG-5)	chinese/big5con
Китайский/Японский/Корейский	chinese/cce
Китайский/Японский/Корейский	chinese/zhcon
Японский	chinese/kon2

Язык	Расположение порта
Японский	japanese/kon2-14dot
Японский	japanese/kon2-16dot

Если moused включен в /etc/rc.conf, может потребоваться дополнительная настройка. По умолчанию курсор мыши драйвера syscons(4) занимает диапазон `0xd0-0xd3` в наборе символов. Если язык использует этот диапазон, измените диапазон курсора, добавив следующую строку в /etc/rc.conf:

```
mousechar_start=3
```

26.2.3. Настройка Xorg

В главе Система X Window описана установка и настройка Xorg. При настройке локализации в Xorg дополнительные шрифты и методы ввода доступны в коллекции портов FreeBSD. Настройки интернационализации для конкретных приложений, такие как шрифты и меню, можно настроить в ~/.Xresources, что должно позволить пользователям видеть выбранный язык в меню графических приложений.

Протокол X Input Method (XIM) является стандартом Xorg для ввода неанглийских символов. В Доступные методы ввода приведены приложения для ввода, доступные в коллекции портов FreeBSD. Также доступны дополнительные приложения Fcitx и Uim.

Таблица 40. Доступные методы ввода

Язык	Метод ввода
Китайский	chinese/gcin
Китайский	chinese/ibus-chewing
Китайский	chinese/ibus-pinyin
Китайский	chinese/oxim
Китайский	chinese/scim-fcitx
Китайский	chinese/scim-pinyin
Китайский	chinese/scim-tables
Японский	japanese/ibus-anthy
Японский	japanese/ibus-mozc
Японский	japanese/ibus-skk
Японский	japanese/im-ja
Японский	japanese/kinput2
Японский	japanese/scim-anthy
Японский	japanese/scim-canna
Японский	japanese/scim-honoka

Язык	Метод ввода
Японский	japanese/scim-honoka-plugin-romkan
Японский	japanese/scim-honoka-plugin-wnn
Японский	japanese/scim-prime
Японский	japanese/scim-skk
Японский	japanese/scim-tables
Японский	japanese/scim-tomoe
Японский	japanese/scim-uim
Японский	japanese/skkinput
Японский	japanese/skkinput3
Японский	japanese/uim-anthy
Корейский	korean/ibus-hangul
Корейский	korean/imhangul
Корейский	korean/nabi
Корейский	korean/scim-hangul
Корейский	korean/scim-tables
Вьетнамский	vietnamese/xvnkb
Вьетнамский	vietnamese/x-unikey

26.3. Поиск приложений с поддержкой i18n

Приложения i18n разрабатываются с использованием наборов i18n в библиотеках. Это позволяет разработчикам написать один простой файл и перевести отображаемые меню и тексты на каждый язык.

Коллекция портов FreeBSD (FreeBSD Ports Collection) включает множество приложений со встроенной поддержкой широких или многобайтовых символов для различных языков. Такие приложения содержат в своих названиях `i18n` для удобства идентификации. Однако они не всегда поддерживают нужный язык.

Некоторые приложения могут быть скомпилированы с указанием конкретной кодировки. Обычно это делается в Makefile порта или путем передачи значения в configure. Для получения дополнительной информации о том, как определить необходимое значение configure или параметры компиляции при сборке порта, обратитесь к документации по i18n в исходном коде соответствующего порта FreeBSD или к Makefile порта.

26.4. Настройка локали для определённых языков

Этот раздел содержит примеры настройки локализации системы FreeBSD для русского языка. Далее приведены дополнительные ресурсы для локализации других языков.

26.4.1. Русский язык (КОИ8-R кодировка)

В этом разделе показаны конкретные настройки, необходимые для локализации системы FreeBSD на русский язык. Подробное описание каждого типа настроек приведено в Использование локализации.

Чтобы установить эту локаль для регистрационной оболочки, добавьте следующие строки в файл ~/.login_conf каждого пользователя:

```
me:My Account:\
    :charset=KOI8-R:\
    :lang=ru_RU.KOI8-R:
```

Для настройки консоли добавьте следующие строки в /etc/rc.conf:

```
keymap="ru.utf-8"
scrnmap="utf-82cp866"
font8x16="cp866b-8x16"
font8x14="cp866-8x14"
font8x8="cp866-8x8"
mousechar_start=3
```

Для каждой записи `ttyv` в /etc/ttys используйте `cons25r` в качестве типа терминала.

Для настройки печати необходим специальный выходной фильтр для преобразования из KOI8-R в CP866, так как большинство принтеров с русскими символами используют аппаратную кодовую страницу CP866. FreeBSD включает стандартный фильтр для этой цели — /usr/libexec/lpr/ru/koi2alt. Чтобы использовать этот фильтр, добавьте следующую запись в /etc/printcap:

```
lp|Russian local line printer:\
    :sh:of=/usr/libexec/lpr/ru/koi2alt:\
    :lp=/dev/lpt0:sd=/var/spool/output/lpd:lf=/var/log/lpd-errs:
```

Обратитесь к printcap(5) для более подробного объяснения.

Для настройки поддержки русских имён файлов в монтируемых файловых системах MS-DOS® добавьте параметр `-L` и указание локали при добавлении записи в /etc/fstab:

```
/dev/ad0s2      /dos/c  msdos   rw,-Lru_RU.KOI8-R 0 0
```

Обратитесь к mount_msdosfs(8) для получения дополнительной информации.

Для настройки русских шрифтов в Xorg установите пакет x11-fonts/xorg-fonts-cyrillic. Затем проверьте раздел `"Files"` в /etc/X11/xorg.conf. Следующая строка должна быть добавлена *перед* всеми остальными записями `FontPath`:

```
FontPath   "/usr/local/lib/X11/fonts/cyrillic"
```

Дополнительные кириллические шрифты доступны в Коллекции портов.

Чтобы активировать русскую раскладку клавиатуры, добавьте следующее в раздел `"Keyboard"` файла /etc/xorg.conf:

```
Option "XkbLayout"   "us,ru"
Option "XkbOptions"  "grp:toggle"
```

Убедитесь, что `XkbDisable` закомментирован в этом файле.

Для `grp:toggle` используйте `Right Alt`, для `grp:ctrl_shift_toggle` — `Ctrl` + `Shift`. Для `grp:caps_toggle` используйте `CapsLock`. Прежняя функция `CapsLock` остаётся доступной только в режиме LAT при использовании `Shift` + `CapsLock`. `grp:caps_toggle` не работает в Xorg по неизвестной причине.

Если на клавиатуре есть клавиши «Windows®» и некоторые неалфавитные клавиши отображаются неправильно, добавьте следующую строку в файл /etc/xorg.conf:

```
Option "XkbVariant" ",winkeys"
```

Русская раскладка XKB может не работать с нелокализованными приложениями. Минимально локализованные приложения должны вызывать функцию `XtSetLanguageProc (NULL, NULL, NULL);` в начале программы.

См. http://koi8.pp.ru/xwin.html для получения дополнительных инструкций по локализации приложений Xorg. Более общую информацию о кодировке KOI8-R можно найти на http://koi8.pp.ru/.

26.4.2. Дополнительные ресурсы для конкретных языков

В этом разделе приведены дополнительные ресурсы для настройки других локалей.

Традиционный китайский для Тайваня

Проект FreeBSD-Taiwan предлагает руководство на китайском языке по FreeBSD по адресу http://netlab.cse.yzu.edu.tw/~statue/freebsd/zh-tut/.

Локализация на греческий язык

Полная статья о поддержке греческого языка в FreeBSD доступна здесь, только на греческом, как часть официальной документации FreeBSD на греческом языке.

Японская и корейская локализация

Для японского языка обратитесь к http://www.jp.FreeBSD.org/, а для корейского — к http://www.kr.FreeBSD.org/.

Неанглоязычная документация FreeBSD

Некоторые участники проекта FreeBSD перевели части документации FreeBSD на другие языки. Они доступны по ссылкам на веб-сайте FreeBSD или в /usr/share/doc.

Глава 27. Обновление и смена версии FreeBSD

27.1. Обзор

FreeBSD постоянно развивается между выпусками. Некоторые предпочитают использовать официальные выпущенные версии, в то время как другие следят за последними изменениями в разработке. Однако даже официальные выпуски часто обновляются с исправлениями безопасности и другими критическими исправлениями. Независимо от используемой версии, FreeBSD предоставляет все необходимые инструменты для поддержания системы в актуальном состоянии и позволяет легко обновляться между версиями. В этой главе описывается, как отслеживать разработку системы и основные инструменты для поддержания FreeBSD в актуальном состоянии.

Прочтите эту главу, чтобы узнать:

- Как поддерживать систему FreeBSD в актуальном состоянии с помощью freebsd-update или Git.
- Как сравнить состояние установленной системы с известной чистой копией.
- Как поддерживать установленную документацию в актуальном состоянии с помощью Git или портов документации.
- Разница между двумя ветвями разработки: FreeBSD-STABLE и FreeBSD-CURRENT.
- Как пересобрать и переустановить всю базовую систему.

Прежде чем читать эту главу, необходимо:

- Правильно настроить сетевое подключение (Сложные вопросы работы в сети).
- Знать, как устанавливать дополнительное стороннее программное обеспечение (Установка приложений: Пакеты и Порты).

В этой главе для получения и обновления исходных кодов FreeBSD используется `git`. При желании можно использовать порт или пакет devel/git.

27.2. Введение

27.2.1. Мажорные и минорные версии

Начиная с версии FreeBSD 6.0-RELEASE, выпуски FreeBSD строго придерживаются двухуровневой схемы нумерации. Термин «основная версия» (мажорная) используется для обозначения первой части номера версии, например «14» в «14.3-RELEASE», а термин «дополнительная версия» (минорная) — для второй части номера версии, например «3» в «14.3-RELEASE».

Кроме того, ветки релизов FreeBSD имеют уровень исправлений (патчей — patch level), который опускается, если равен нулю, и увеличивается с каждым выпуском пакета бюллетеней безопасности и уведомлений об исправлении ошибок (errata notices), например «14.3-RELEASE-p7» для седьмого пакета исправлений, опубликованного для FreeBSD 14.3.

Как следствие модели поддержки FreeBSD, номера версий FreeBSD возрастают монотонно в пределах одной основной версии, но не между разными основными версиями. Например, FreeBSD 14.2-RELEASE, 13.5-RELEASE, 14.3-RELEASE, 15.0-RELEASE, 14.4-RELEASE и 15.1-RELEASE были выпущены именно в таком порядке.

27.2.2. Обновление (апдейт) и смена версии (апгрейд)

Термин «обновление» (updating) мы используем для описания применения бюллетеней безопасности или исправлений ошибок (errata), которые могут увеличивать уровень исправлений (patch level), но не изменяют номер версии. Термин «апгрейд» (upgrading) используется для описания замены всей базовой системы на систему с другим номером версии.

27.2.3. Пути апгрейда

После установки FreeBSD можно сделать обновление или апгрейд с использованием бинарных патчей, пакетов или из исходных кодов. Во всех случаях FreeBSD поддерживает обновление или апгрейд только с хронологически более старой версии на более новую.

В частности, FreeBSD поддерживает обновление до более нового уровня исправлений той же версии либо апгрейд с данной версии до более новой дополнительной версии в пределах той же основной версии (например, с 14.3 до 14.4) или до более новой версии следующей основной версии (например, с 14.3 до 15.0 или 15.1). Переход на более старую версию в пределах той же основной версии (например, с 14.4 на 14.3), на более старую версию следующей основной версии (например, с 14.4, выпущенной в марте 2026 года, на 15.0, выпущенную тремя месяцами ранее, в декабре 2025 года) или пропуск основной версии (например, с 13.5 на 15.1) не рекомендуется и не поддерживается, даже если инструменты это допускают.

27.3. Обновление FreeBSD

Применение исправлений безопасности своевременно и обновление до более новой версии операционной системы являются важными аспектами текущего администрирования системы. FreeBSD включает утилиту `freebsd-update`, которую можно использовать для выполнения обеих задач.

Этот инструмент поддерживает бинарные обновления безопасности и исправления ошибок в FreeBSD без необходимости ручной компиляции и установки патчей или нового ядра. Бинарные обновления доступны для всех архитектур и выпусков, которые в настоящее время поддерживаются командой безопасности. Список поддерживаемых выпусков и их предполагаемые даты окончания поддержки приведены на странице https://www.FreeBSD.org/security/.

Эта утилита также поддерживает обновления операционной системы со сменой младших версий, а также переход на другую ветку выпусков. Перед обновлением до нового выпуска ознакомьтесь с его анонсом, так как он содержит важную информацию, относящуюся к выпуску. Анонсы выпусков доступны по адресу https://www.FreeBSD.org/releases/.

Если существует crontab(5), использующий возможности freebsd-update(8), его необходимо отключить перед обновлением операционной системы.

В этом разделе описывается файл конфигурации, используемый `freebsd-update`, демонстрируется применение патча безопасности, обновление со сменой младшей или старшей версии операционной системы, а также рассматриваются некоторые аспекты, которые следует учитывать при обновлении ОС.

27.3.1. Файл конфигурации

Файл конфигурации по умолчанию для `freebsd-update` работает без изменений. Некоторые пользователи могут захотеть настроить конфигурацию по умолчанию в /etc/freebsd-update.conf, чтобы лучше контролировать процесс. Комментарии в этом файле объясняют доступные опции, но следующие моменты могут потребовать дополнительного пояснения:

```
# Components of the base system which should be kept updated.
Components world kernel
```

Этот параметр определяет, какие части FreeBSD будут поддерживаться в актуальном состоянии. По умолчанию обновляется вся базовая система и ядро. Вместо этого можно указать отдельные компоненты, например `src/base` или `src/sys`. Однако лучшим вариантом будет оставить значение по умолчанию, так как изменение параметра для включения конкретных компонентов требует перечисления всех необходимых элементов. Со временем это может привести к катастрофическим последствиям, поскольку исходный код и исполняемые файлы могут перестать быть синхронизированными.

```
# Paths which start with anything matching an entry in an IgnorePaths
# statement will be ignored.
IgnorePaths /boot/kernel/linker.hints
```

Чтобы оставить указанные каталоги, такие как /bin или /sbin, без изменений в процессе обновления, добавьте их пути в эту инструкцию. Эта опция может быть использована для предотвращения перезаписи локальных изменений с помощью `freebsd-update`.

```
# Paths which start with anything matching an entry in an UpdateIfUnmodified
# statement will only be updated if the contents of the file have not been
# modified by the user (unless changes are merged; see below).
UpdateIfUnmodified /etc/ /var/ /root/ /.cshrc /.profile
```

Эта опция будет обновлять только неизменённые конфигурационные файлы в указанных каталогах. Любые изменения, внесённые пользователем, предотвратят автоматическое

обновление этих файлов. Существует другая опция, `KeepModifiedMetadata`, которая предписывает `freebsd-update` сохранять изменения во время слияния.

```
# When upgrading to a new FreeBSD release, files which match MergeChanges
# will have any local changes merged into the version from the new release.
MergeChanges /etc/ /var/named/etc/ /boot/device.hints
```

Список каталогов с файлами конфигурации, которые `freebsd-update` должен попытаться объединить. Процесс объединения файлов представляет собой серию патчей diff(1). Объединения могут быть приняты, открыты в редакторе или привести к прерыванию работы `freebsd-update`. В случае сомнений создайте резервную копию /etc и просто примите объединения.

```
# Directory in which to store downloaded updates and temporary
# files used by FreeBSD Update.
# WorkDir /var/db/freebsd-update
```

Этот каталог предназначен для всех патчей и временных файлов. В случае обновления версии в этом расположении должно быть доступно как минимум гигабайт дискового пространства.

```
# When upgrading between releases, should the list of Components be
# read strictly (StrictComponents yes) or merely as a list of components
# which *might* be installed of which FreeBSD Update should figure out
# which actually are installed and upgrade those (StrictComponents no)?
# StrictComponents no
```

Когда эта опция установлена в `yes`, `freebsd-update` будет считать список `Components` полным и не будет пытаться вносить изменения за его пределами. Фактически, `freebsd-update` попытается обновить каждый файл, принадлежащий списку `Components`.

Обратитесь к freebsd-update.conf(5) для получения дополнительной информации.

27.3.2. Применение обновлений безопасности

Процесс применения патчей безопасности FreeBSD был упрощен, что позволяет администратору поддерживать систему в полностью обновленном состоянии с помощью `freebsd-update`. Дополнительная информация о выпусках безопасности FreeBSD доступна в Рекомендации по безопасности FreeBSD.

Обновления безопасности FreeBSD могут быть загружены и установлены с помощью следующих команд. Первая команда проверит наличие доступных обновлений и, если они есть, выведет список файлов, которые будут изменены при их применении. Вторая команда применит обновления.

```
# freebsd-update fetch
```

```
# freebsd-update install
```

Если обновление включает какие-либо исправления для ядра, системе потребуется перезагрузка для загрузки исправленного ядра. Если исправление было применено к каким-либо выполняемым файлам, затронутые приложения следует перезапустить, чтобы использовалась исправленная версия бинарного файла.

Обычно пользователь должен быть готов перезагрузить систему. Чтобы проверить, требуется ли перезагрузка из-за обновления ядра, выполните команды `freebsd-version -k` и `uname -r`. Перезагрузите систему, если их вывод различается.

Систему можно настроить для автоматической проверки обновлений раз в день, добавив следующую запись в /etc/crontab:

```
@daily                                  root    freebsd-update cron
```

Если существуют патчи, они будут автоматически загружены, но не применены. Пользователь `root` получит письмо, чтобы патчи можно было просмотреть и установить вручную с помощью `freebsd-update install`.

Если что-то пойдет не так, `freebsd-update` может откатить последние изменения с помощью следующей команды:

```
# freebsd-update rollback
Uninstalling updates... done.
```

Вновь, система должна быть перезапущена, если ядро или какие-либо модули ядра были изменены, и все затронутые бинарные файлы должны быть перезапущены.

Только ядро GENERIC может быть автоматически обновлено с помощью `freebsd-update`. Если установлено пользовательское ядро, его необходимо пересобрать и переустановить после завершения установки обновлений `freebsd-update`. Имя ядра по умолчанию — *GENERIC*. Для проверки его установки можно использовать команду uname(1).

Всегда сохраняйте копию ядра GENERIC в /boot/GENERIC. Оно может быть полезно при диагностике различных проблем и при обновлении версий. Инструкции по получению копии ядра GENERIC можно найти в разделе Собственные ядра в FreeBSD 9.X и более поздних версиях.

Если конфигурация по умолчанию в /etc/freebsd-update.conf не была изменена, `freebsd-update` установит обновленные исходные коды ядра вместе с остальными обновлениями. Затем пересборка и переустановка нового пользовательского ядра могут быть выполнены обычным способом.

Обновления, распространяемые через `freebsd-update`, не всегда затрагивают ядро. Нет

необходимости пересобирать собственное ядро, если исходные коды ядра не были изменены командой `freebsd-update install`. Однако `freebsd-update` всегда обновляет файл /usr/src/sys/conf/newvers.sh. Текущий уровень патча, обозначаемый числом после `-p` в выводе `uname -r`, берётся из этого файла. Пересборка собственного ядра, даже если ничего больше не менялось, позволяет `uname` точно отображать текущий уровень патча системы. Это особенно полезно при обслуживании нескольких систем, так как позволяет быстро оценить установленные обновления на каждой из них.

27.3.3. Выполнение обновлений с изменением младших и старших версий

Обновления на новую минорную или мажорную версии могут быть выполнены путем указания `freebsd-update` целевой версии выпуска.

После каждого нового выпуска `RELEASE` серверы сборки пакетов FreeBSD в течение ограниченного периода времени **не** используют новую версию операционной системы. Это обеспечивает преемственность для многих пользователей, которые не обновляются сразу после объявления о выпуске. Например:

- Пакеты для пользователей версий 13.1 и 13.2 будут собираться на сервере под управлением 13.1, пока поддержка 13.1 не прекратится

— и, что критически важно:

- модуль ядра, собранный в версии 13.1, **может** не подойти для версии 13.2.

Итак, при любом обновлении ОС, будь то обновление младшей или старшей версии, если ваш пакет требует включить какой-либо модуль ядра:

- **будьте готовы собрать модуль из исходного кода**

Если система работает на собственном ядре, убедитесь перед началом обновления, что копия ядра GENERIC осталась в /boot/GENERIC. Обратитесь к разделу Собственные ядра системы в FreeBSD 9.X и более поздних версиях для получения инструкций о том, как получить копию ядра GENERIC.

Перед обновлением до новой версии убедитесь, что текущая установка FreeBSD обновлена с учётом исправлений безопасности и ошибок:

```
# freebsd-update fetch
# freebsd-update install
```

Следующая команда, выполненная на системе FreeBSD 13.1, обновит её до версии FreeBSD 13.2:

```
# freebsd-update -r 13.2-RELEASE upgrade
```

После получения команды `freebsd-update` оценит файл конфигурации и текущую систему, чтобы собрать информацию, необходимую для выполнения обновления. На экране появится список компонентов, которые были и не были обнаружены. Например:

```
Looking up update.FreeBSD.org mirrors... 1 mirrors found.
Fetching metadata signature for 13.1-RELEASE from update1.FreeBSD.org... done.
Fetching metadata index... done.
Inspecting system... done.

The following components of FreeBSD seem to be installed:
kernel/smp src/base src/bin src/contrib src/crypto src/etc src/games
src/gnu src/include src/krb5 src/lib src/libexec src/release src/rescue
src/sbin src/secure src/share src/sys src/tools src/ubin src/usbin
world/base world/info world/lib32 world/manpages

The following components of FreeBSD do not seem to be installed:
kernel/generic world/catpages world/dict world/doc world/games
world/proflibs

Does this look reasonable (y/n)? y
```

На этом этапе `freebsd-update` попытается загрузить все файлы, необходимые для обновления. В некоторых случаях пользователю могут быть заданы вопросы относительно того, что установить или как продолжить.

При использовании собственного ядра вышеуказанный шаг вызовет предупреждение, подобное следующему:

```
WARNING: This system is running a "MYKERNEL" kernel, which is not a
kernel configuration distributed as part of FreeBSD 13.1-RELEASE.
This kernel will not be updated: you MUST update the kernel manually
before running "/usr/sbin/freebsd-update install"
```

Это предупреждение можно безопасно проигнорировать на данном этапе. Обновлённое ядро GENERIC будет использовано как промежуточный шаг в процессе обновления.

После загрузки всех патчей в локальную систему они будут применены. Этот процесс может занять некоторое время в зависимости от скорости и загруженности машины. Затем будут объединены конфигурационные файлы. Процесс объединения требует вмешательства пользователя, так как файл может быть объединён автоматически или на экране появится редактор для ручного объединения. Результаты каждого успешного объединения будут отображаться пользователю по мере выполнения процесса. Неудачное или пропущенное объединение приведет к прерыванию процесса. Пользователям рекомендуется создать резервную копию /etc и вручную объединить важные файлы, такие как master.passwd или

group, позже.

Система пока не изменяется, так как все исправления и слияния происходят в другом каталоге. Как только все исправления будут успешно применены, все конфигурационные файлы объединены и процесс, по всей видимости, пройдёт без проблем, пользователь может записать изменения на диск с помощью следующей команды:

```
# freebsd-update install
```

Ядро и модули ядра будут пропатчены первыми. Если система работает с пользовательским ядром, используйте nextboot(8), чтобы установить ядро для следующей загрузки в обновлённый /boot/GENERIC:

```
# nextboot -k GENERIC
```

Перед перезагрузкой с ядром GENERIC убедитесь, что оно содержит все драйверы, необходимые для корректной загрузки системы и подключения к сети, если обновляемая машина доступна удалённо. В частности, если текущее кастомное ядро включает встроенную функциональность, обычно предоставляемую модулями ядра, убедитесь, что временно загрузили эти модули в ядро GENERIC с помощью механизма /boot/loader.conf. Также рекомендуется отключить несущественные сервисы, а также любые монтирования дисков и сетевые подключения до завершения процесса обновления.

Машину теперь следует перезагрузить с обновлённым ядром:

```
# shutdown -r now
```

После того как система снова станет доступной, перезапустите `freebsd-update` с помощью следующей команды. Поскольку состояние процесса сохранено, `freebsd-update` начнёт не с начала, а перейдёт к следующему этапу и удалит все старые общие библиотеки и объектные файлы.

```
# freebsd-update install
```

В зависимости от того, были ли изменены номера версий библиотек, может быть только две фазы установки вместо трёх.

Обновление завершено. Если было выполнено обновление старшей версии, переустановите все порты и пакеты, как описано в Обновление пакетов после обновления старшей версии.

Собственные ядра в FreeBSD 9.X и более поздних версиях

Перед использованием `freebsd-update` убедитесь, что копия ядра GENERIC осталось в /boot/GENERIC. Если собственное ядро собиралось только один раз, ядро в /boot/kernel.old является ядром `GENERIC`. Просто переименуйте этот каталог в /boot/GENERIC.

Если собственное ядро собиралось более одного раза или неизвестно, сколько раз оно пересобиралось, необходимо получить копию ядра `GENERIC`, соответствующую текущей версии операционной системы. Если есть физический доступ к системе, копию ядра `GENERIC` можно установить с установочного носителя:

```
# mount /media
# cd /media/usr/freebsd-dist
# tar -C/ -xvf kernel.txz boot/kernel/kernel
```

Или ядро `GENERIC` может быть пересобрано и установлено из исходного кода:

```
# cd /usr/src
# make kernel __MAKE_CONF=/dev/null SRCCONF=/dev/null
```

Чтобы ядро было идентифицировано как `GENERIC` с помощью `freebsd-update`, конфигурационный файл GENERIC не должен быть каким-либо образом изменён. Также рекомендуется собирать ядро без каких-то дополнительных специальных опций.

Перезагрузка с ядром GENERIC не требуется, так как `freebsd-update` нужен только файл /boot/GENERIC.

Обновление пакетов после обновления старшей версии

Обычно установленные приложения продолжают работать без проблем после обновления младших версий. Старшие версии используют разные бинарные интерфейсы приложений (ABI), что приведёт к неработоспособности большинства сторонних приложений. После обновления старшей версии необходимо обновить все установленные пакеты и порты. Пакеты можно обновить с помощью `pkg upgrade`. Для обновления установленных портов используйте утилиты, например ports-mgmt/portmaster.

Принудительное обновление всех установленных пакетов заменит пакеты на новые версии из репозитория, даже если номер версии не увеличился. Это необходимо из-за изменения версии ABI при обновлении между старшими версиями FreeBSD. Принудительное обновление можно выполнить с помощью команды:

```
# pkg-static upgrade -f
```

Перестройка всех приложений, установленных из коллекции портов, может быть выполнена с помощью следующей команды:

```
# portmaster -af
```

Эта команда отобразит экраны конфигурации для каждого приложения, имеющего настраиваемые параметры, и будет ожидать взаимодействия пользователя с этими экранами. Чтобы предотвратить такое поведение и использовать только параметры по умолчанию, добавьте `-G` в указанную выше команду.

После завершения обновления программного обеспечения завершите процесс обновления, выполнив последний вызов `freebsd-update`, чтобы устранить все нерешенные вопросы, оставшиеся в процессе обновления:

```
# freebsd-update install
```

Если временно использовалось ядро GENERIC, сейчас самое время собрать и установить новое пользовательское ядро, следуя инструкциям из раздела Настройка ядра FreeBSD.

Перезагрузите машину с новой версией FreeBSD. Процесс обновления завершен.

27.3.4. Сравнение состояния системы

Состояние установленной версии FreeBSD можно проверить с помощью `freebsd-update IDS`, сравнив его с известной исправной копией. Эта команда оценивает текущие версии системных утилит, библиотек и конфигурационных файлов и может использоваться как встроенная система обнаружения вторжений (IDS — Intrusion Detection System).

Эта команда не является заменой настоящей системы обнаружения вторжений, такой как security/snort. Поскольку `freebsd-update` хранит данные на диске, возможность их подделки очевидна. Хотя эту возможность можно уменьшить, используя `kern.securelevel` и сохраняя данные `freebsd-update` на файловой системе только для чтения, когда они не используются, лучшее решение — сравнить систему с защищённым диском, таким как DVD или надёжно хранимое внешнее USB-устройство. Альтернативный метод обеспечения функциональности системы обнаружения вторжений с использованием встроенной утилиты описан в Проверка двоичных файлов

Для начала сравнения укажите выходной файл для сохранения результатов:

```
# freebsd-update IDS >> outfile.ids
```

Система будет проверена, и в указанный выходной файл будет отправлен длинный список файлов вместе с хеш-значениями SHA256, как для известного значения в релизе, так и для текущей установки.

Записи в списке очень длинные, но формат вывода легко анализировать. Например, чтобы получить список всех файлов, которые отличаются от файлов в выпуске, выполните следующую команду:

```
# cat outfile.ids | awk '{ print $1 }' | more
/etc/master.passwd
/etc/motd
/etc/passwd
/etc/pf.conf
```

Этот пример вывода был сокращён, так как существует гораздо больше файлов. Некоторые файлы имеют естественные изменения. Например, /etc/passwd будет изменён, если в систему были добавлены пользователи. Модули ядра могут отличаться, так как `freebsd-update` мог их обновить. Чтобы исключить определённые файлы или каталоги, добавьте их в параметр `IDSIgnorePaths` в файле /etc/freebsd-update.conf.

27.4. Обновление загрузочного кода

Следующие руководства описывают процесс обновления загрузочного кода и загрузчиков: gpart(8), gptboot(8), gptzfsboot(8) и loader.efi(8).

27.5. Обновление документации

Документация является неотъемлемой частью операционной системы FreeBSD. Хотя актуальная версия документации FreeBSD всегда доступна на сайте FreeBSD (Портал документации), может быть удобно иметь актуальную локальную копию веб-сайта FreeBSD, руководств, FAQ и статей.

Этот раздел описывает, как использовать исходный код или коллекцию портов FreeBSD для поддержания локальной копии документации FreeBSD в актуальном состоянии.

Для получения информации о редактировании и отправке исправлений в документацию обратитесь к руководству "Проект документации FreeBSD: введение для новых участников" (Проект документации FreeBSD: введение для новых участников).

27.5.1. Обновление документации из исходных текстов

Пересборка документации FreeBSD из исходных текстов требует набора инструментов, которые не входят в базовую систему FreeBSD. Необходимые инструменты можно установить, следуя этим шагам из вводного руководства Проекта документации FreeBSD.

После установки используйте `git`, чтобы получить чистую копию исходного кода документации:

```
# git clone https://git.FreeBSD.org/doc.git /usr/doc
```

Первоначальная загрузка исходного кода документации может занять некоторое время. Дождитесь завершения процесса.

Будущие обновления исходного кода документации можно получить, выполнив:

```
# git pull
```

После того как актуальный снимок исходников документации будет загружен в /usr/doc, всё готово для обновления установленной документации.

Полное обновление можно выполнить, набрав:

```
# cd /usr/doc
# make
```

27.6. Отслеживание ветки разработки

FreeBSD имеет две ветви разработки: FreeBSD-CURRENT и FreeBSD-STABLE.

Этот раздел содержит объяснение каждой ветки и её целевой аудитории, а также инструкции по поддержанию системы в актуальном состоянии для каждой из веток.

27.6.1. Использование FreeBSD-CURRENT

FreeBSD-CURRENT — это "передовой край" разработки FreeBSD, и от пользователей FreeBSD-CURRENT ожидается высокий уровень технической подготовки. Менее опытным пользователям, которые хотят следить за веткой разработки, рекомендуется использовать FreeBSD-STABLE.

FreeBSD-CURRENT - это самые последние исходные коды FreeBSD, включающие текущие работы, экспериментальные изменения и переходные механизмы, которые могут как присутствовать, так и отсутствовать в следующем официальном выпуске. Хотя многие разработчики FreeBSD компилируют исходный код FreeBSD-CURRENT ежедневно, бывают короткие периоды, когда сборка кода может быть невозможна. Эти проблемы решаются как можно быстрее, но приведёт ли FreeBSD-CURRENT к катастрофе или новым возможностям, может зависеть от того, когда исходный код был синхронизирован.

FreeBSD-CURRENT предоставляется для трёх основных категорий пользователей:

1. Участники сообщества FreeBSD, которые активно работают над какой-либо частью дерева исходного кода.
2. Участники сообщества FreeBSD, которые активно тестируют. Они готовы тратить время на решение проблем, высказывать тематические предложения по изменениям и общему направлению развития FreeBSD, а также предоставлять патчи.
3. Пользователи, которые хотят следить за происходящим, использовать текущие исходные коды для справки или иногда делать комментарии или вклад в код.

Версия FreeBSD-CURRENT *не* должна рассматриваться как быстрый способ получить новые функции до следующего релиза, так как предрелизные функции ещё не полностью протестированы и, скорее всего, содержат ошибки. Это не быстрый способ исправления багов, так как каждый коммит с такой же вероятностью может добавить новые ошибки, как

и исправить существующие. FreeBSD-CURRENT ни в коем случае не является "официально поддерживаемой".

Для отслеживания FreeBSD-CURRENT:

1. Присоединитесь к спискам рассылки Список рассылки, посвящённый обсуждению FreeBSD-CURRENT и Коммиты в ветку main репозитория исходного кода. Это *важно* для того, чтобы видеть комментарии пользователей о текущем состоянии системы и получать важные уведомления о состоянии FreeBSD-CURRENT.

   Список Коммиты в ветку main репозитория исходного кода фиксирует запись журнала изменений для каждого внесённого изменения, а также любую сопутствующую информацию о возможных побочных эффектах.

   Чтобы присоединиться к этим спискам, перейдите на Сервер списков рассылки FreeBSD, выберите список, к которому хотите подписаться, и следуйте инструкциям. Для отслеживания изменений во всём дереве исходного кода, а не только в FreeBSD-CURRENT, подпишитесь на Коммиты во все ветки репозитория исходного кода.

2. Синхронизация с исходным кодом FreeBSD-CURRENT. Обычно для получения кода -CURRENT из ветки `main` репозитория FreeBSD Git используется `git` (подробности см. в “Использование Git”).

3. Из-за размера репозитория некоторые пользователи предпочитают синхронизировать только те разделы исходного кода, которые их интересуют или для которых они готовят патчи. Однако пользователи, планирующие собирать операционную систему из исходных кодов, должны загрузить *весь* FreeBSD-CURRENT, а не только отдельные его части.

   Перед компиляцией FreeBSD-CURRENT внимательно прочитайте /usr/src/Makefile и следуйте инструкциям в Обновление FreeBSD из исходного кода. Ознакомьтесь с Список рассылки, посвящённый обсуждению FreeBSD-CURRENT и /usr/src/UPDATING, чтобы быть в курсе других процедур начальной загрузки, которые иногда становятся необходимыми на пути к следующему выпуску.

4. Будьте активными! Пользователям FreeBSD-CURRENT рекомендуется предлагать свои идеи по улучшению или исправлению ошибок. Предложения с приложенным кодом всегда приветствуются.

27.6.2. Использование FreeBSD-STABLE

FreeBSD-STABLE - это ветка разработки, из которой создаются основные выпуски. Изменения попадают в эту ветку медленнее и с общим предположением, что они сначала были протестированы в FreeBSD-CURRENT. Это *всё ещё* ветка разработки, и в любой момент исходный код FreeBSD-STABLE может быть или не быть пригодным для общего использования. Это просто ещё один инженерный трек разработки, а не ресурс для конечных пользователей. Пользователям, у которых нет ресурсов для тестирования, следует использовать последний выпуск FreeBSD.

Те, кто заинтересован в отслеживании или участии в процессе разработки FreeBSD,

особенно в контексте следующего выпуска FreeBSD, могут рассмотреть возможность подписки на FreeBSD-STABLE.

Хотя ветка FreeBSD-STABLE должна компилироваться и работать в любое время, это не может быть гарантировано. Поскольку больше людей используют FreeBSD-STABLE, чем FreeBSD-CURRENT, неизбежно, что иногда в FreeBSD-STABLE будут обнаружены ошибки и крайние случаи, которые не были очевидны в FreeBSD-CURRENT. По этой причине не следует слепо следовать за FreeBSD-STABLE. Особенно важно *не* обновлять какие-либо производственные серверы до FreeBSD-STABLE без тщательного тестирования кода в среде разработки или тестирования.

Для отслеживания FreeBSD-STABLE:

1. Присоединяйтесь к рассылке Список рассылки, посвящённый обсуждению FreeBSD-STABLE, чтобы быть в курсе зависимостей сборки, которые могут появиться в FreeBSD-STABLE, или других вопросов, требующих особого внимания. Разработчики также будут объявлять в этой рассылке о рассмотрении спорных исправлений или обновлений, давая пользователям возможность высказаться, если у них есть замечания по предлагаемым изменениям.

   Присоединяйтесь к соответствующему списку рассылки git для отслеживаемой ветки. Например, пользователи, отслеживающие ветку 14-STABLE, должны подписаться на Коммиты в ветки stable репозитория исходного кода. Этот список рассылает сообщения с комментарием коммита для каждого внесённого изменения, как только оно сделано, а также любую соответствующую информацию о возможных побочных эффектах.

   Чтобы присоединиться к этим спискам, перейдите на Сервер списков рассылки FreeBSD, выберите список для подписки и следуйте инструкциям. Для отслеживания изменений во всём дереве исходного кода подпишитесь на Коммиты во все ветки репозитория исходного кода.

2. Для установки новой системы FreeBSD-STABLE установите последний выпуск FreeBSD-STABLE с сайтов зеркал FreeBSD или используйте ежемесячную промежуточную сборку, собранную из FreeBSD-STABLE. Дополнительную информацию о промежуточных сборках можно найти на www.freebsd.org/ru/snapshots.

   Для компиляции или обновления существующей системы FreeBSD до FreeBSD-STABLE используйте `git` для получения исходного кода нужной ветки. Имена веток, например `stable/13`, перечислены на сайте www.freebsd.org/ru/releng.

3. Перед компиляцией или обновлением до FreeBSD-STABLE внимательно прочитайте файл /usr/src/Makefile и следуйте инструкциям в Обновление FreeBSD из исходного кода. Ознакомьтесь с Список рассылки, посвящённый обсуждению FreeBSD-STABLE и файлом /usr/src/UPDATING, чтобы быть в курсе других процедур начальной загрузки, которые иногда становятся необходимыми на пути к следующему выпуску.

27.6.3. N-номер

При поиске ошибок важно знать, какие версии исходного кода использовались для создания системы, в которой обнаружена проблема. FreeBSD предоставляет информацию о

версии, встроенную в ядро. uname(1) извлекает эту информацию, например:

```
% uname -v
FreeBSD 14.0-CURRENT #112 main-n247514-031260d64c18: Tue Jun 22 20:43:19 MDT 2021
fred@machine:/usr/home/fred/obj/usr/home/fred/git/head/amd64.amd64/sys/FRED
```

Последнее поле содержит информацию о названии ядра, человеке, который его собрал, и месте, где оно было скомпилировано. Рассматривая 4-е поле, можно увидеть, что оно состоит из нескольких частей:

```
main-n247514-031260d64c18

main          ①
n247514       ②
031260d64c18      ③
④
```

① Имя ветки Git. Примечание: сравнения n-номеров действительны только для веток, опубликованных проектом (`main`, `stable/XX` и `releng/XX`). Локальные ветки будут иметь n-номера, которые могут пересекаться с коммитами их родительской ветки.

② n-номер — это линейный счетчик коммитов от начала репозитория Git, начиная с хэша Git, указанного в строке.

③ Хэш Git дерева исходного кода

④ Иногда добавляется суффикс `-dirty`, если ядро было собрано в дереве с незафиксированными изменениями. В данном примере он отсутствует, так как ядро FRED было собрано из чистой копии репозитория.

Команда `git rev-list` используется для нахождения n-номера, соответствующего хэшу Git. Например:

```
% git rev-list --first-parent --count 031260d64c18 ①
247514 ②
```

① Хеш git для перевода (используется тот же хеш из приведённого выше примера)

② n-номер.

Обычно этот номер не так важен. Однако, когда фиксы ошибок попадают в репозиторий, это число позволяет быстро определить, присутствует ли исправление в текущей работающей системе. Разработчики часто ссылаются на хэш коммита (или предоставляют URL, содержащий этот хэш), но не указывают n-номер, так как хэш является легко видимым идентификатором изменения, в отличие от n-номера. В уведомлениях о безопасности и бюллетенях исправлений также указывается n-номер, который можно напрямую сравнить с вашей системой. Если используются неполные (shallow) клоны Git, n-номера не могут быть надежно надёжно сравнены, так как команда git rev-list подсчитывает все ревизии в репозитории, которые неполный клон пропускает.

27.7. Обновление FreeBSD из исходного кода

Обновление FreeBSD путем компиляции из исходного кода предоставляет несколько преимуществ по сравнению с бинарными обновлениями. Код может быть собран с опциями, позволяющими использовать преимущества конкретного оборудования. Части базовой системы могут быть собраны с нестандартными настройками или полностью исключены, если они не нужны или нежелательны. Процесс сборки занимает больше времени для обновления системы по сравнению с установкой бинарных обновлений, но позволяет полностью настроить систему для создания адаптированной версии FreeBSD.

27.7.1. Быстрый старт

Это краткая справка по типовым шагам, используемым для обновления FreeBSD путем сборки из исходного кода. Более подробное описание процесса приведено в следующих разделах.

При переходе с mergemaster(8) на etcupdate(8) первый запуск может некорректно объединить изменения, создавая ложные конфликты. Чтобы избежать этого, выполните следующие действия **перед** обновлением исходных кодов и сборкой нового мира:

```
# etcupdate extract ①
# etcupdate diff ②
```

① Загрузите базы данных стандартных файлов /etc; дополнительную информацию смотрите в etcupdate(8).

② Проверьте изменения после начальной настройки. Удалите все локальные изменения, которые больше не нужны, чтобы уменьшить вероятность конфликтов при будущих обновлениях.

- Обновление и сборка

```
# git -C /usr/src pull ①
check /usr/src/UPDATING  ②
# cd /usr/src            ③
# make -j4 buildworld    ④
# make -j4 kernel        ⑤
# shutdown -r now        ⑥
# etcupdate -p           ⑦
# cd /usr/src            ⑧
# make installworld      ⑨
# etcupdate -B           ⑩
# shutdown -r now        ⑪
```

① Получите последнюю версию исходного кода. Подробнее о получении и обновлении исходного кода см. в Обновление исходного кода.

② Проверьте /usr/src/UPDATING на наличие необходимых действий перед или после сборки из исходного кода.

③ Перейдите в исходный каталог.

④ Соберите систему (world), всё кроме ядра системы.

⑤ Скомпилируйте и установите ядро системы. Это эквивалентно `make buildkernel installkernel`.

⑥ Перезагрузите систему с новым ядром.

⑦ Обновите и объедините конфигурационные файлы в /etc/, это обязательно перед выполнением installworld.

⑧ Перейдите в исходный каталог.

⑨ Установить систему (world).

⑩ Обновите и объедините конфигурационные файлы в /etc/.

⑪ Перезагрузите систему для использования новой собранной системы (world) и ядра.

27.7.2. Подготовка к обновлению исходного кода

Прочитайте файл /usr/src/UPDATING. В этом файле описаны все необходимые действия, которые нужно выполнить до или после обновления.

27.7.3. Обновление исходного кода

Исходный код FreeBSD находится в /usr/src/. Предпочтительный метод обновления этого исходного кода — через систему управления версиями Git. Проверьте, что исходный код находится под управлением версий:

```
# cd /usr/src
# git remote --v
origin  https://git.freebsd.org/src.git (fetch)
origin  https://git.freebsd.org/src.git (push)
```

Это указывает, что /usr/src/ находится под управлением версий и может быть обновлён с помощью git(1):

```
# git -C /usr/src pull
```

Процесс обновления может занять некоторое время, если каталог не обновлялся в течение долгого периода. После завершения исходный код будет актуальным, и можно приступать к процессу сборки, описанному в следующем разделе.

Получение исходного кода:

Если вывод содержит `fatal: not a git repository`, значит, файлы отсутствуют

> или были установлены другим способом. Необходимо выполнить новое получение исходного кода.

Таблица 41. Версии FreeBSD и ветви репозиториев

uname -r Output	Путь репозитория	Описание
X.Y-RELEASE	releng/X.Y	Версия Release с добавлением только критических исправлений безопасности и ошибок. Эта ветка рекомендуется для большинства пользователей.
X.Y-STABLE	stable/X	Версия Release и все дополнительные разработки в этой ветке. *STABLE* означает, что двоичный интерфейс приложений (ABI) не изменяется, поэтому программное обеспечение, скомпилированное для более ранних версий, продолжает работать. Например, программное обеспечение, скомпилированное для работы на FreeBSD 14.5, будет работать и на FreeBSD {rel-major}-STABLE, собранной позже. Ветки STABLE иногда содержат ошибки или несовместимости, которые могут повлиять на пользователей, хотя они обычно быстро исправляются.
X-CURRENT	main	Последняя невыпущенная разрабатываемая версия FreeBSD. Ветка CURRENT может содержать серьёзные ошибки или проблемы совместимости и рекомендуется только для опытных пользователей.

Определите версию FreeBSD с помощью uname(1):

```
# uname -r
13.2-RELEASE
```

На основе Версии FreeBSD и ветви репозиториев, исходный код для обновления 13.2-RELEASE имеет путь в репозитории releng/13.2. Этот путь используется при проверке исходного кода:

```
# mv /usr/src /usr/src.bak ①
# git clone --branch releng/13.2 https://git.FreeBSD.org/src.git /usr/src ②
```

① Переместите старый каталог в другое место. Если в этом каталоге нет локальных изменений, его можно удалить.

② Путь из Версии FreeBSD и ветви репозитория добавляется к URL репозитория. Третий параметр — это целевой каталог для исходного кода в локальной системе.

27.7.4. Сборка из исходного кода

Система (world) , или вся операционная система, за исключением ядра, компилируется. Это делается в первую очередь, чтобы предоставить актуальные инструменты для сборки ядра. Затем компилируется само ядро:

```
# cd /usr/src
# make buildworld
# make buildkernel
```

Скомпилированный код записывается в /usr/obj.

Вот основные шаги. Дополнительные параметры для управления сборкой описаны ниже.

Выполнение чистой сборки

Некоторые версии системы сборки FreeBSD оставляют ранее скомпилированный код во временном каталоге объектов /usr/obj. Это может ускорить последующие сборки, избегая перекомпиляции неизменившегося кода. Для принудительной чистой пересборки всего используйте `cleanworld` перед началом сборки:

```
# make cleanworld
```

Установка количества задач

Увеличение количества задач сборки на многоядерных процессорах может повысить скорость сборки. Определите количество ядер с помощью `sysctl hw.ncpu`. Процессоры различаются, как и системы сборки, используемые в разных версиях FreeBSD, поэтому тестирование — единственный надёжный способ определить, как разное количество задач влияет на скорость сборки. В качестве отправной точки рассмотрите значения от половины до удвоенного количества ядер. Количество задач указывается с помощью `-j`.

Пример 39. Увеличение количества заданий сборки

Сборка системы и ядра с использованием четырёх задач:

```
# make -j4 buildworld buildkernel
```

Сборка только ядра

`buildworld` должен быть выполнен, если исходный код изменился. После этого `buildkernel` для сборки ядра может быть запущен в любое время. Чтобы собрать только ядро:

```
# cd /usr/src
# make buildkernel
```

Сборка собственного ядра

Стандартное ядро FreeBSD основано на *конфигурационном файле ядра* GENERIC. Ядро GENERIC включает наиболее часто востребованные драйверы устройств и параметры. Иногда полезно или необходимо собрать собственное ядро, добавляя или удаляя драйверы

устройств и параметры для соответствия конкретным требованиям.

Например, разработчик компактного встраиваемого компьютера с крайне ограниченным объёмом оперативной памяти может удалить ненужные драйверы устройств или опции, чтобы немного уменьшить размер ядра.

Файлы конфигурации ядра находятся в /usr/src/sys/arch/conf/, где *arch* — это результат выполнения команды `uname -m`. На большинстве компьютеров это `amd64`, что соответствует каталогу с конфигурационными файлами /usr/src/sys/amd64/conf/.

/usr/src можно удалить или создать заново, поэтому предпочтительнее хранить конфигурационные файлы собственного ядра в отдельном каталоге, например, в /root. Свяжите конфигурационный файл ядра с каталогом conf. Если этот каталог будет удалён или перезаписан, конфигурацию ядра можно снова связать с новым каталогом.

Собственный конфигурационный файл можно создать, скопировав файл GENERIC. В этом примере новое собственное ядро предназначено для сервера хранения данных, поэтому он назван STORAGESERVER:

```
# cp /usr/src/sys/amd64/conf/GENERIC /root/STORAGESERVER
# cd /usr/src/sys/amd64/conf
# ln -s /root/STORAGESERVER .
```

Затем отредактируйте файл /root/STORAGESERVER, добавляя или удаляя устройства или параметры, как показано в config(5).

Собственное ядро собирается путем установки `KERNCONF` в файл конфигурации ядра в командной строке:

```
# make buildkernel KERNCONF=STORAGESERVER
```

27.7.5. Установка скомпилированного кода

После завершения шагов `buildworld` и `buildkernel` новые ядро и система устанавливаются:

```
# cd /usr/src
# make installkernel
# shutdown -r now
# cd /usr/src
# make installworld
# shutdown -r now
```

Если было собрано собственное ядро, `KERNCONF` также должен быть установлен для использования нового собственного ядра:

```
# cd /usr/src
# make installkernel KERNCONF=STORAGESERVER
# shutdown -r now
# cd /usr/src
# make installworld
# shutdown -r now
```

27.7.6. Завершение обновления

Для окончания обновления выполняется несколько завершающих задач. Изменённые конфигурационные файлы объединяются с новыми версиями, устаревшие библиотеки находятся и удаляются, после чего система перезагружается.

Объединение конфигурационных файлов с помощью etcupdate(8)

etcupdate(8) — это инструмент для управления обновлениями файлов, которые не обновляются в процессе выполнения `installworld`, таких как файлы в /etc/. Он управляет обновлениями, выполняя трёхстороннее слияние изменений, внесённых в эти файлы, с локальными версиями. etcupdate(8) разработан для минимизации вмешательства пользователя.

В общем, etcupdate(8) не требует специальных аргументов для своей работы. Однако есть удобная промежуточная команда для проверки того, что будет сделано при первом использовании etcupdate(8):

```
# etcupdate diff
```

Эта команда позволяет пользователю отслеживать изменения конфигурации.

Если etcupdate(8) не может автоматически объединить файл, конфликты слияния можно разрешить вручную, выполнив:

```
# etcupdate resolve
```

При переходе с mergemaster(8) на etcupdate(8) первый запуск может некорректно объединить изменения, создавая ложные конфликты. Чтобы избежать этого, выполните следующие действия **перед** обновлением исходных кодов и сборкой нового мира:

```
# etcupdate extract ①
# etcupdate diff ②
```

① Загрузите базы данных стандартных файлов /etc; дополнительную информацию смотрите в etcupdate(8).

② Проверьте изменения после начальной настройки. Удалите все локальные изменения, которые больше не нужны, чтобы уменьшить вероятность конфликтов при будущих обновлениях.

Проверка устаревших файлов и библиотек

Некоторые устаревшие файлы или каталоги могут остаться после обновления. Эти файлы могут быть найдены:

```
# make check-old
```

и удалены:

```
# make delete-old
```

Некоторые устаревшие библиотеки также могут остаться. Их можно обнаружить с помощью:

```
# make check-old-libs
```

и удалить

```
# make delete-old-libs
```

Программы, которые всё ещё использовали эти старые библиотеки, перестанут работать после удаления библиотек. Эти программы необходимо пересобрать или заменить после удаления старых библиотек.

Когда известно, что все старые файлы или каталоги можно безопасно удалить, нажатие `y` и `Enter` для подтверждения удаления каждого файла можно избежать, установив параметр `BATCH_DELETE_OLD_FILES` в команде. Например:

```
# make BATCH_DELETE_OLD_FILES=yes delete-old-libs
```

Перезагрузка после обновления

Последним шагом после обновления является перезагрузка компьютера, чтобы все изменения вступили в силу:

```
# shutdown -r now
```

27.8. Обновление FreeBSD пакетами

Начиная с версии 14.0-RELEASE, проект FreeBSD публикует набор пакетов базовой системы, используя pkg(8). Теперь удобством pkg можно пользоваться не только в коллекции портов FreeBSD, но и для самой системы. pkg(8) применим и к FreeBSD. Базовые пакеты и их использование традиционно называют *pkgbase*.

Публикация официальных пакетов началась в октябре 2023 года. Их использование в настоящее время считается экспериментальным для выпуска FreeBSD 14.

Начиная с версии 15.0-RELEASE, freebsd-base(7) будет поддерживаться в режиме предварительной (tech preview) версии для:

- установки
- минорных обновлений (с изменением номера версии после точки)
- мажорных обновлений (с изменением старшего номера версии)

freebsd-base в конечном итоге заменит:

- дистрибутивные наборы в виде tar-архивов, такие как `base.txz` или `kernel.txz`, которые традиционно использовались для установки ОС с помощью bsdinstall(8)
- freebsd-update(8) для обновлений операционной системы.

Текущий план заключается в том, чтобы freebsd-update работал с pkg.

Замена дистрибутивных наборов займёт некоторое время, пока freebsd-base не станет полностью стабильным!

Пакеты базовой системы дополняют сборку и установку из исходных кодов, которая по-прежнему доступна для тех, кто желает собирать собственные ядра или пользовательское пространство. Также возможно собирать пользовательские пакеты базовой системы из локальных исходных кодов, равно как и просто полагаться на официально предоставляемые пакеты.

27.8.1. Преобразование хоста для использования freebsd-base

FreeBSD 14.0-RELEASE или новее может быть преобразована для использования пакетов Базовой Системы. Более ранние версии должны сначала обновиться с помощью freebsd-update, а затем выполнить преобразование.

Для преобразования рекомендован инструмент под названием pkgbasify, спонсированный Фондом FreeBSD. Следуйте инструкции, приведённой там.

freebsd-update(8) будет усовершенствован для использования pkg и pkgbasify.

Обратите внимание, что для данной миграции требуется до 5 гигабайт дополнительного свободного пространства для загрузки, распаковки и перемещения конфликтующих файлов. Инструмент pkgbasify не проверяет

> это, и пользователь обязан самостоятельно убедиться в наличии достаточного пространства перед запуском миграции.

pkgbasify(8) (или любой результат, который даст D51594 или запрос порта) выполняет 6 основных задач:

- Создаёт резервную среду загрузки (только для ZFS) с помощью bectl(8)
- Создаёт новые файлы конфигурации репозитория пакетов
- Обновляет существующие компоненты системы, такие как базовая система, ядро, lib32, отладочные файлы
- Объединяет существующие и новые файлы конфигурации
- Обновляет базы данных passwd и capabilities
- Перезапускает sshd немедленно

27.8.2. Обновление хоста с использованием freebsd-base

> Это все ещё находится в разработке, поэтому, пожалуйста, будьте осторожны, особенно при преобразовании существующей системы для использования freebsd-base.

Создайте папку для пользовательских конфигурационных файлов репозитория pkg, если она ещё не существует.

```
# mkdir -p /usr/local/etc/pkg/repos/
```

Для использования репозитория freebsd-base создайте файл конфигурации репозитория пакетов с именем `FreeBSD-base.conf`:

```
FreeBSD-base {
    url = "pkg+https://pkg.freebsd.org/${ABI}/base_release_${VERSION_MINOR}";
    mirror_type = "srv";
    signature_type = "fingerprints";
    fingerprints = "/usr/share/keys/pkg";
    enabled = yes;
}
```

для получения дополнительной информации о конкретных параметрах конфигурации см. pkg.conf(5).

Существуют различные ветки на выбор (путем соответствующего изменения url):

Таблица 42. Ветки пакетов базовой системы

Ветка	Периодичность	URL

main	дважды в день - 12:00 и 00:00 по UTC	pkg+https://pkg.freebsd.org/${ABI}/base_latest
main	еженедельно – воскресенье в 12:00 UTC	pkg+https://pkg.freebsd.org/${ABI}/base_weekly
stable/14	дважды в день – 12:00 и 00:00 по UTC	pkg+https://pkg.freebsd.org/${ABI}/base_latest
stable/14	еженедельно – воскресенье в 12:00 UTC	pkg+https://pkg.freebsd.org/${ABI}/base_weekly
releng/14.3	дважды в день – 12:00 и 00:00 по UTC	pkg+https://pkg.freebsd.org/${ABI}/base_release_3
releng/15.0	дважды в день – 12:00 и 00:00 по UTC	pkg+https://pkg.freebsd.org/${ABI}/base_release_0
stable/15	дважды в день – 12:00 и 00:00 по UTC	pkg+https://pkg.freebsd.org/${ABI}/base_latest
stable/15	еженедельно – воскресенье в 12:00 UTC	pkg+https://pkg.freebsd.org/${ABI}/base_weekly

Для обновления системы измените файл конфигурации в соответствии с желаемым выпуском и выполните:

```
# pkg update -r FreeBSD-base
# pkg upgrade -r FreeBSD-base
```

проверьте, корректны ли эти пакеты, и примите изменения.

Перезагрузите операционную систему, выполнив следующую команду:

```
# shutdown -r now
```

Крупные обновления

С ZFS загрузочные среды позволяют выполнить обновление без прерывания работы программного обеспечения. Если система не использует root-on-ZFS: сделайте резервную копию перед обновлением.

Подготовка

Измените `/usr/local/etc/pkg/repos/FreeBSD-base.conf`, чтобы указать на правильный основной выпуск, например `base_release_0 ` для 15.0-RELEASE, чтобы он выглядел так:

Этот файл может быть изменён для включения в RELEASE с другим местоположением.

```
FreeBSD-base {
    url = "pkg+https://pkg.freebsd.org/${ABI}/base_release_0";
    mirror_type = "srv";
    signature_type = "fingerprints";
    fingerprints = "/usr/share/keys/pkg";
    enabled = yes;
}
```

Поскольку pkgbase в FreeBSD 14 является экспериментальным, то и крупное обновление pkgbase до версии 15 также носит экспериментальный характер. На момент написания в некоторых крайних случаях крупное обновление удаляет pkg, что приводит к ошибкам сегментации. Это считается (известной проблемой) для 15.0-RELEASE. Чтобы обойти эту проблему, заблокируйте pkg перед обновлением.

```
# pkg -c /mnt/upgrade lock pkg
```

Экспериментальное основное обновление с FreeBSD 14 до 15 (или 16.0-CURRENT) должно сопровождаться установкой метапакета `FreeBSD-set-minimal`.

Сохраните список сторонних пакетов на случай, если они понадобятся в дальнейшем:

```
pkg prime-origins | sort -u > /var/tmp/pkg-prime-origins.txt
```

Переход на новую основную версию с ZFS

Создайте загрузочное окружение с помощью bectl(8) и назовите его в соответствии с версией, на которую выполняется обновление

```
# bectl create 15.0-RELEASE
```

Создайте каталог, затем смонтируйте новую загрузочную среду туда, чтобы обеспечить работу с pkg chroot(8) и rootdir.

```
# mkdir /mnt/upgrade
```

```
# bectl mount 15.0-RELEASE /mnt/upgrade
```

Следующий шаг обновит загрузочное окружение до указанной версии.

Этот шаг может удалить пакеты, не входящие в базовую систему, включая запущенную среду рабочего стола.

Установите переменную окружения ABI для обновления основной версии (замените amd64 на архитектуру и 15 на целевую версию). Задайте для pkg-static каталог chroot, указывающий на точку монтирования загрузочного окружения.

```
# env ABI=FreeBSD:15:amd64 pkg-static -c /mnt/upgrade upgrade -r FreeBSD-base
```

Появится запрос о необходимости игнорировать несоответствие версий, который выглядит следующим образом:

```
Newer FreeBSD version for package FreeBSD-zoneinfo:
To ignore this error set IGNORE_OSVERSION=yes
- package: 1500058
- running userland: 1500000
Ignore the mismatch and continue? [y/N]:
```

Проверьте и подтвердите это.

Чтобы проверить успешность операции, выполните chroot в точку монтирования загрузочного окружения и выполните команду `freebsd-version -kru`.

```
# chroot /mnt/upgrade
# freebsd-version -ku
# exit
```

Временно активируйте загрузочную среду и перезагрузитесь.

```
# bectl activate -t 15.0-RELEASE
# shutdown -r now
```

Если система работает не так, как ожидалось: перезагрузите ОС, чтобы прекратить использование временно активной загрузочной среды. Следующая загрузка будет использовать среду, существовавшую до обновления.

Если система стабильно работает после перезагрузки, не забудьте активировать эту загрузочную среду на постоянной основе, чтобы предотвратить случайную загрузку старой системы.

```
bectl activate 15.0-RELEASE
```

Переход на новую основную версию без ZFS

Установите переменную окружения ABI для обновления основной версии (замените amd64 на архитектуру и 15 на целевую версию).

```
# env ABI=FreeBSD:15:amd64 pkg-static upgrade -r FreeBSD-base
```

Появится запрос о необходимости игнорировать несоответствие версий, который выглядит следующим образом:

```
Newer FreeBSD version for package FreeBSD-zoneinfo:
To ignore this error set IGNORE_OSVERSION=yes
- package: 1500058
- running userland: 1500000
Ignore the mismatch and continue? [y/N]:
```

Проверьте и подтвердите это.

Чтобы проверить успешность обновления, выполните команду `freebsd-version -kru`.

Затем перезагрузите систему.

Задачи после обновления

Если pkg был заблокирован во время обновления, то в качестве временного решения удалите блокировку следующим образом:

```
# pkg unlock pkg
```

После обновления до новой основной версии может потребоваться обновление установленных пакетов для соответствия версии ABI.

```
# pkg upgrade
```

Рассмотрите возможность получения помощи по ссылке Поддержка FreeBSD, если возникли проблемы.

27.8.3. Ручная сборка pkgbase и публикация в локальной сети

При создании пользовательских пакетов pkgbase клонируйте дерево исходного кода FreeBSD:

```
# cd /usr/src
# git clone https://github.com/freebsd/freebsd-src.git /usr/src
```

Перейдите на ветку (checkout) релиза, для которого будут собираться пакеты:

```
# git checkout releng/14.3
```

Запустите процесс сборки, в зависимости от доступных ресурсов это может занять некоторое время. Установите количество параллельных процессов в соответствии с числом ядер процессора.

Этот пример написан для 8-ядерного процессора:

```
# make -j8 buildworld && make -j8 buildkernel && make -j8 packages
```

При частой сборке рекомендуется использовать devel/ccache для ускорения последующих сборок из кэша.

После сборки пакеты будут сохранены в `/usr/obj/usr/src/repo/FreeBSD:14:amd64/14.3p2` или аналогичный каталог, в зависимости от версии сборки.

Чтобы опубликовать эти пакеты в сети, настройте службу nginx и используйте следующую конфигурацию location в настройках HTTP-сервера:

```
 location /FreeBSD:14:amd64 {
   alias /usr/obj/usr/src/repo/FreeBSD:14:amd64/;
   autoindex on;
 }
```

И перезагрузите службу nginx.

Если не используется https, используйте небольшой файл конфигурации на клиентах для указания только что собранной версии pkgbase, отредактировав `/usr/local/etc/pkg/repos/upgrade.conf`:

```
upgrade: {
    url          = http://ip.of.the.server/FreeBSD:14:amd64/14.3p2
    enabled      = yes
}
```

и используйте его, как описано выше (но используйте у опции -r параметр upgrade вместо FreeBSD-base).

27.9. Распространение обновлений на несколько машин

Когда несколько машин должны отслеживать одно и то же дерево исходных кодов, это приводит к расточительному использованию дискового пространства, пропускной способности сети и процессорного времени, если каждая система загружает исходные коды и пересобирает всё самостоятельно. Решение заключается в том, чтобы одна машина выполняла основную часть работы, а остальные монтировали результаты через NFS. В этом разделе описывается метод организации такого процесса. Дополнительные сведения об использовании NFS см. в Network File System (NFS).

Сначала определите набор машин, которые будут запускать один и тот же набор бинарных файлов, известный как *набор сборки*. На каждой машине может быть своё собственное ядро, но пользовательские бинарные файлы должны быть одинаковыми. Из этого набора выберите машину, которая будет *машиной для сборки*, на которой будут собираться система и ядро. В идеале это должна быть производительная машина с достаточным количеством свободных ресурсов CPU для выполнения команд `make buildworld` и `make buildkernel`.

Выберите машину, которая будет *тестовой машиной* для проверки обновлений программного обеспечения перед их внедрением в производство. Это *должна* быть машина, которая может позволить себе длительный простой. Она может быть той же машиной, что и сборщик, но это не обязательно.

Все машины в этом наборе сборки должны монтировать /usr/obj и /usr/src сборочной машины через NFS. Для нескольких наборов сборки /usr/src должен находиться на одной сборочной машине и монтироваться по NFS на остальных.

Убедитесь, что файлы /etc/make.conf и /etc/src.conf на всех машинах в наборе сборки соответствуют таковым на машине сборки. Это означает, что машина сборки должна собирать все части базовой системы, которые будут устанавливаться на любой машине из набора сборки. Кроме того, каждая машина сборки должна иметь имя ядра, указанное с помощью `KERNCONF` в /etc/make.conf, а машина сборки должна перечислить их все в своём `KERNCONF`, указав собственное ядро первым. Машина сборки должна иметь конфигурационные файлы ядра для каждой машины в своём каталоге /usr/src/sys/arch/conf.

На машине для сборки соберите ядро и систему, как описано в Обновление FreeBSD из исходного кода, но не устанавливайте ничего на машину для сборки. Вместо этого установите собранное ядро на тестовую машину. На тестовой машине смонтируйте /usr/src и /usr/obj через NFS. Затем выполните `shutdown now`, чтобы перейти в однопользовательский режим для установки нового ядра и системы, а также запустите `etcupdate` как обычно. По завершении перезагрузитесь, чтобы вернуться к обычной многопользовательской работе.

После проверки того, что все на тестовой машине работает правильно, используйте ту же процедуру для установки нового программного обеспечения на каждой из остальных машин в наборе сборки.

Тот же метод можно применить к дереву портов. Первый шаг — предоставить общий доступ через NFS к /usr/ports для всех машин в наборе сборки. Чтобы настроить

/etc/make.conf для совместного использования distfiles, установите DISTDIR в общий каталог, доступный для записи пользователем, в которого отображается root при монтировании NFS. Каждая машина должна установить WRKDIRPREFIX в локальный каталог сборки, если порты собираются локально. В качестве альтернативы, если система сборки предназначена для создания и распространения пакетов на машины в наборе сборки, установите PACKAGES в системе сборки в каталог, аналогичный DISTDIR.

27.10. Сборка на хостах, отличных от FreeBSD

Исторически для сборки требовался хост с FreeBSD. В настоящее время FreeBSD можно собирать на Linux и macOS.

Для сборки на хосте, отличном от FreeBSD, рекомендуется использовать скрипт tools/build/make.py. Этот скрипт служит обёрткой для bmake — реализации make, используемой в FreeBSD. Он обеспечивает загрузку необходимых инструментов, включая make(1) из FreeBSD, и правильную настройку среды сборки. В частности, он устанавливает переменные внешнего инструментария, такие как XCC, XLD и другие. Кроме того, скрипт может передавать дополнительные аргументы командной строки, например -j 4 для параллельной сборки или конкретные цели make, в bmake.

Вместо скрипта tools/build/make.py также можно использовать свежую версию bmake. Однако в этом случае необходимые переменные окружения нужно задавать вручную (проще всего получить их список, выполнив tools/build/make.py --debug).

В противном случае список требований для сборки FreeBSD довольно короткий. Фактически, он сводится к установке нескольких зависимостей.

На macOS единственная зависимость — LLVM. Необходимые зависимости можно установить с помощью менеджера пакетов (например, Homebrew):

```
brew install llvm
```

На дистрибутивах Linux установите Clang версии 10.0 или новее, а также заголовочные файлы для libarchive и libbz2 (обычно поставляются в пакетах libarchive-dev и libbz2-dev).

После установки зависимостей система должна быть способна собрать FreeBSD.

Например, следующая команда tools/build/make.py собирает систему (world):

```
MAKEOBJDIRPREFIX=/tmp/obj tools/build/make.py -j 8 TARGET=arm64 TARGET_ARCH=aarch64
buildworld
```

Он собирает систему для целевой архитектуры aarch64:arm64 на 8 CPU и использует /tmp/obj для объектных файлов. Обратите внимание, что переменные MAKEOBJDIRPREFIX, TARGET и TARGET_ARCH обязательны при сборке на хостах, отличных от FreeBSD. Также убедитесь, что создан каталог для объектных файлов, указанный в переменной окружения

`MAKEOBJDIRPREFIX`.

Обратитесь к arch(7) и build(7) для получения дополнительной информации.

Глава 28. DTrace

28.1. Обзор

DTrace, также известный как Dynamic Tracing, был разработан Sun™ в качестве инструмента для выявления узких мест в производительности рабочих и предпроизводственных систем. Помимо диагностики проблем с производительностью, DTrace можно использовать для исследования и отладки неожиданного поведения как в ядре FreeBSD, так и в пользовательских программах.

DTrace — это выдающийся инструмент для профилирования, обладающий впечатляющим набором возможностей для диагностики проблем в системе. Его также можно использовать для запуска предварительно написанных скриптов, чтобы воспользоваться его функциональностью. Пользователи могут создавать собственные утилиты, используя язык DTrace D, что позволяет настраивать профилирование в соответствии с конкретными потребностями.

Реализация FreeBSD обеспечивает полную поддержку DTrace в ядре и экспериментальную поддержку DTrace в пользовательском пространстве. DTrace в пользовательском пространстве позволяет пользователям выполнять трассировку границ функций для программ пользовательского пространства с использованием провайдера `pid`, а также вставлять статические зонды в программы пользовательского пространства для последующей трассировки. Некоторые порты, такие как databases/postgresql12-server и lang/php74, имеют опцию DTrace для включения статических зондов.

Официальное руководство по DTrace поддерживается проектом illumos по адресу illumos Dynamic Tracing Guide.

Прочтите эту главу, чтобы узнать:

- Что такое DTrace и какие возможности он предоставляет.
- Различия между реализацией DTrace в Solaris™ и реализацией, предоставляемой FreeBSD.
- Как включить и использовать DTrace в FreeBSD.

Прежде чем читать эту главу, необходимо:

- Понимать основы UNIX® и FreeBSD (Основы FreeBSD).
- Иметь некоторое представление о безопасности и о том, как она относится к FreeBSD (Безопасность).

28.2. Различия в реализациях

В то время как DTrace в FreeBSD схож с тем, что представлен в Solaris™, различия существуют. Основное различие заключается в том, что в FreeBSD DTrace реализован в виде набора модулей ядра, и DTrace нельзя использовать до загрузки этих модулей. Чтобы загрузить все необходимые модули:

```
# kldload dtraceall
```

Начиная с FreeBSD 10.0-RELEASE, модули автоматически загружаются при запуске dtrace(1).

FreeBSD использует параметр ядра DDB_CTF для включения поддержки загрузки данных ctf(5) из модулей ядра и самого ядра. CTF — это Compact C Type Format от Solaris™, который инкапсулирует упрощённую форму отладочной информации, аналогичную DWARF и устаревшему stabs. Данные CTF добавляются в бинарные файлы с помощью инструментов сборки ctfconvert(1) и ctfmerge(1). Утилита ctfconvert обрабатывает отладочные секции DWARFELF, созданные компилятором, а ctfmerge объединяет секции CTFELF из объектных файлов в исполняемые файлы или разделяемые библиотеки.

В FreeBSD есть некоторые провайдеры, которых нет в Solaris™. Наиболее примечательным является провайдер dtmalloc, который позволяет трассировать malloc(9) по типам в ядре FreeBSD. Некоторые провайдеры, присутствующие в Solaris™, такой как cpc, отсутствуют в FreeBSD. Они могут появиться в будущих версиях FreeBSD. Кроме того, некоторые провайдеры, доступные в обеих операционных системах, несовместимы в том смысле, что их пробы имеют разные типы аргументов. Таким образом, скрипты D, написанные для Solaris™, могут работать или не работать без изменений в FreeBSD, и наоборот.

Из-за различий в безопасности, только пользователь root может использовать DTrace в FreeBSD. В Solaris™ есть несколько проверок безопасности низкого уровня, которые пока отсутствуют в FreeBSD. Поэтому доступ к /dev/dtrace/dtrace строго ограничился только пользователем root.

DTrace распространяется под лицензией Common Development and Distribution License (CDDL). Чтобы ознакомиться с текстом этой лицензии в FreeBSD, см. /usr/src/cddl/contrib/opensolaris/OPENSOLARIS.LICENSE или просмотрите её онлайн по адресу http://opensource.org/licenses/CDDL-1.0. Хотя ядро FreeBSD с поддержкой DTrace лицензировано под BSD, лицензия CDDL применяется при распространении модулей в бинарной форме или при загрузке бинарных файлов.

28.3. Включение поддержки DTrace

В FreeBSD 9.2 и 10.0 поддержка DTrace встроена в ядро GENERIC. Пользователям более ранних версий FreeBSD или тем, кто предпочитает статически компилировать поддержку DTrace, следует добавить следующие строки в пользовательский конфигурационный файл ядра и перекомпилировать ядро, следуя инструкциям в Настройка ядра FreeBSD:

```
options         KDTRACE_HOOKS
options         DDB_CTF
makeoptions DEBUG=-g
makeoptions WITH_CTF=1
```

Пользователи архитектуры AMD64 также должны добавить эту строку:

```
options         KDTRACE_FRAME
```

Эта опция обеспечивает поддержку dtrace_fbt(4). Хотя DTrace будет работать без этой опции, поддержка трассировки входов и выходов функций будет ограничена.

После перезагрузки системы FreeBSD с новым ядром или загрузки модулей DTrace с помощью `kldload dtraceall`, установите текущий DTrace Toolkit (sysutils/dtrace-toolkit) — набор готовых скриптов для сбора системной информации. Включает скрипты для проверки открытых файлов, использования памяти, загрузки CPU и многое другое. Некоторые скрипты также присутствуют в базовой системе FreeBSD — см. /usr/share/dtrace.

Скрипты в каталоге /usr/share/dtrace были специально портированы для FreeBSD. Не все скрипты из DTrace Toolkit будут работать на FreeBSD без изменений, и для некоторых может потребоваться доработка.

Набор инструментов DTrace включает множество скриптов на специальном языке DTrace. Этот язык называется D и очень похож на C++. Подробное обсуждение языка выходит за рамки данного документа. Обратитесь к странице d(7) для обзора языка D в FreeBSD. Язык D также подробно рассмотрен в illumos Dynamic Tracing Guide.

28.4. Включение DTrace во внешних модулях ядра

Чтобы добавить поддержку DTrace во внешний модуль ядра, что полезно для разработки и отладки, включите следующую строку в Makefile модуля:

```
CFLAGS+= -DKDTRACE_HOOKS
```

Этот флаг включает DTrace-хуки во время компиляции, позволяя проводить расширенную отладку и мониторинг модуля. Убедитесь, что модуль перекомпилирован после этого изменения, чтобы активировать функциональность DTrace.

28.5. Использование DTrace

Скрипты DTrace состоят из списка одного или нескольких *проб* (точек инструментирования), где каждая проба связана с действием. Когда условие для пробы выполняется, запускается соответствующее действие. Например, действие может происходить при открытии файла, запуске процесса или выполнении строки кода. Действие может заключаться в записи некоторой информации или изменении контекстных переменных. Чтение и запись контекстных переменных позволяют пробам обмениваться информацией и совместно анализировать взаимосвязь различных событий.

Для просмотра всех проб администратор может выполнить следующую команду:

```
# dtrace -l | more
```

У каждого зонда есть `ID`, `PROVIDER` (например, `dtrace` или `fbt`), `MODULE` и `FUNCTION NAME`. Дополнительную информацию об этой команде можно найти в dtrace(1).

Примеры в этом разделе дают общее представление о том, как использовать два полностью поддерживаемых скрипта из DTrace Toolkit: hotkernel и procsystime.

Скрипт hotkernel предназначен для определения функции, которая использует наибольшее время ядра. Он выводит информацию, похожую на следующую:

```
# cd /usr/local/share/dtrace-toolkit
# ./hotkernel
Sampling... Hit Ctrl-C to end.
```

Как указано, используйте комбинацию клавиш `Ctrl` + `C`, чтобы остановить процесс. После завершения скрипт отобразит список функций ядра и информацию о времени, сортируя вывод в порядке возрастания времени:

```
kernel`_thread_lock_flags                                2   0.0%
0xc1097063                                               2   0.0%
kernel`sched_userret                                     2   0.0%
kernel`kern_select                                       2   0.0%
kernel`generic_copyin                                    3   0.0%
kernel`_mtx_assert                                       3   0.0%
kernel`vm_fault                                          3   0.0%
kernel`sopoll_generic                                    3   0.0%
kernel`fixup_filename                                    4   0.0%
kernel`_isitmyx                                          4   0.0%
kernel`find_instance                                     4   0.0%
kernel`_mtx_unlock_flags                                 5   0.0%
kernel`syscall                                           5   0.0%
kernel`DELAY                                             5   0.0%
0xc108a253                                               6   0.0%
kernel`witness_lock                                      7   0.0%
kernel`read_aux_data_no_wait                             7   0.0%
kernel`Xint0x80_syscall                                  7   0.0%
kernel`witness_checkorder                                7   0.0%
kernel`sse2_pagezero                                     8   0.0%
kernel`strncmp                                           9   0.0%
kernel`spinlock_exit                                    10   0.0%
kernel`_mtx_lock_flags                                  11   0.0%
kernel`witness_unlock                                   15   0.0%
kernel`sched_idletd                                    137   0.3%
0xc10981a5                                           42139  99.3%
```

Этот скрипт также работает с модулями ядра. Чтобы использовать эту возможность, запустите скрипт с ключом `-m`:

```
# ./hotkernel -m
```

```
Sampling... Hit Ctrl-C to end.
^C
MODULE                                                  COUNT   PCNT
0xc107882e                                                  1   0.0%
0xc10e6aa4                                                  1   0.0%
0xc1076983                                                  1   0.0%
0xc109708a                                                  1   0.0%
0xc1075a5d                                                  1   0.0%
0xc1077325                                                  1   0.0%
0xc108a245                                                  1   0.0%
0xc107730d                                                  1   0.0%
0xc1097063                                                  2   0.0%
0xc108a253                                                 73   0.0%
kernel                                                    874   0.4%
0xc10981a5                                             213781  99.6%
```

Скрипт procsystime захватывает и выводит время использования системных вызовов для заданного идентификатора процесса (PID) или имени процесса. В следующем примере был создан новый экземпляр /bin/csh. Затем был запущен procsystime, который оставался в ожидании, пока в другом экземпляре csh было набрано несколько команд. Вот результаты этого теста:

```
# ./procsystime -n csh
Tracing... Hit Ctrl-C to end...
^C

Elapsed Times for processes csh,

         SYSCALL          TIME (ns)
          getpid               6131
       sigreturn               8121
           close              19127
           fcntl              19959
             dup              26955
         setpgid              28070
            stat              31899
       setitimer              40938
           wait4              62717
       sigaction              67372
     sigprocmask             119091
    gettimeofday             183710
           write             263242
          execve             492547
           ioctl             770073
           vfork            3258923
      sigsuspend            6985124
            read         3988049784
```

Как показано, системный вызов read(2) использовал наибольшее время в наносекундах,

тогда как системный вызов getpid(2) использовал наименьшее количество времени.

Глава 29. Режим устройства USB / USB OTG

29.1. Обзор

Эта глава посвящена использованию USB Device Mode и USB On The Go (USB OTG) в FreeBSD. В неё входят виртуальные последовательные консоли, виртуальные сетевые интерфейсы и виртуальные USB-накопители.

При работе на оборудовании, поддерживающем режим USB-устройства или USB OTG, который встроен во многие материнские платы, стек USB FreeBSD может работать в *режиме устройства*. Режим устройства позволяет компьютеру представлять себя как различные классы USB-устройств, включая последовательные порты, сетевые адаптеры и устройства хранения данных, или их комбинацию. USB-хост, такой как ноутбук или настольный компьютер, может обращаться к ним так же, как к физическим USB-устройствам. Режим устройства иногда называют «режимом USB-гаджета».

Существует два основных способа, которыми оборудование может обеспечивать функциональность режима устройства: с отдельным "клиентским портом", который поддерживает только режим устройства, и с портом USB OTG, который может обеспечивать как режим устройства, так и режим хоста. Для портов USB OTG стек USB автоматически переключается между режимом хоста и режимом устройства в зависимости от того, что подключено к порту. Подключение USB-устройства, такого как флеш-накопитель, приводит к переключению FreeBSD в режим хоста. Подключение USB-хоста, например компьютера, приводит к переключению FreeBSD в режим устройства. Однонаправленные "клиентские порты" всегда работают в режиме устройства.

То, что FreeBSD предоставляет USB-хосту, зависит от параметра `hw.usb.template` в sysctl. Некоторые шаблоны предоставляют одно устройство, например, последовательный терминал; другие предоставляют несколько устройств, которые можно использовать одновременно. Примером является шаблон 10, который предоставляет устройство хранения данных, последовательную консоль и сетевой интерфейс. Список доступных значений смотрите в usb_template(4).

Обратите внимание, что в некоторых случаях, в зависимости от оборудования и операционной системы хоста, для того чтобы хост заметил изменение конфигурации, может потребоваться либо физическое отключение и повторное подключение, либо принудительное повторное сканирование шины USB способом, зависящим от системы. Если на хосте запущена FreeBSD, можно использовать usbconfig(8) `reset`. Это также необходимо сделать после загрузки usb_template.ko, если USB-хост уже был подключен к разъему USBOTG.

Прочтите эту главу, чтобы узнать:

- Как настроить функциональность USB Device Mode в FreeBSD.
- Как настроить виртуальный последовательный порт на FreeBSD.
- Как подключиться к виртуальному последовательному порту из различных операционных систем.

- Как настроить FreeBSD для предоставления виртуального USB-сетевого интерфейса.
- Как настроить FreeBSD для предоставления виртуального USB-накопителя.

29.2. Виртуальные последовательные порты USB

29.2.1. Настройка последовательных портов в режиме USB-устройства

Поддержка виртуальных последовательных портов предоставляется шаблонами 3, 8 и 10. Обратите внимание, что шаблон 3 работает с Microsoft Windows 10 без необходимости в специальных драйверах и INF-файлах. Другие операционные системы хоста работают со всеми тремя шаблонами. Необходимо загрузить оба модуля ядра usb_template(4) и umodem(4).

Чтобы включить последовательные порты в режиме устройства USB, добавьте следующие строки в /etc/ttys:

```
ttyU0   "/usr/libexec/getty 3wire"  vt100   onifconsole secure
ttyU1   "/usr/libexec/getty 3wire"  vt100   onifconsole secure
```

Затем добавьте следующие строки в /etc/devd.conf:

```
notify 100 {
    match "system"      "DEVFS";
    match "subsystem"   "CDEV";
    match "type"        "CREATE";
    match "cdev"        "ttyU[0-9]+";
    action "/sbin/init q";
};
```

Перезагрузите конфигурацию, если devd(8) уже запущен:

```
# service devd restart
```

Убедитесь, что необходимые модули загружены, и правильный шаблон установлен при загрузке, добавив следующие строки в /boot/loader.conf, создав его, если он ещё не существует:

```
umodem_load="YES"
hw.usb.template=3
```

Для загрузки модуля и установки шаблона без перезагрузки используйте:

```
# kldload umodem
```

```
# sysctl hw.usb.template=3
```

29.2.2. Подключение к последовательным портам в режиме USB-устройств из FreeBSD

Для подключения к плате, настроенной для предоставления последовательных портов в режиме USB-устройства, подключите USB-хост, например ноутбук, к USB OTG или клиентскому USB-порту платы. Используйте команду `pstat -t` на хосте для вывода списка терминальных линий. В конце списка должен отобразиться USB-последовательный порт, например "ttyU0". Для установки соединения используйте:

```
# cu -l /dev/ttyU0
```

После нажатия клавиши `Enter` несколько раз вы увидите приглашение для входа в систему.

29.2.3. Подключение к последовательным портам в режиме USB-устройства из macOS

Для подключения к плате, настроенной для предоставления последовательных портов в режиме USB-устройства, подключите USB-хост, например ноутбук, к USB OTG или клиентскому USB-порту платы. Чтобы открыть соединение, используйте:

```
# cu -l /dev/cu.usbmodemFreeBSD1
```

29.2.4. Подключение к последовательным портам в режиме USB-устройства из Linux

Для подключения к плате, настроенной для предоставления последовательных портов в режиме USB-устройства, подключите USB-хост, например ноутбук, к USB OTG или клиентскому USB-порту платы. Чтобы открыть соединение, используйте:

```
# minicom -D /dev/ttyACM0
```

29.2.5. Подключение к последовательным портам в режиме USB-устройства из Microsoft Windows 10

Для подключения к плате, настроенной для предоставления последовательных портов в режиме USB-устройства, подключите USB-хост, например ноутбук, к USB OTG или клиентскому USB-порту платы. Чтобы открыть соединение, вам понадобится программа для работы с последовательным портом, например PuTTY. Для проверки имени COM-порта, используемого Windows, запустите Диспетчер устройств, раскройте раздел "Порты (COM и LPT)". Вы увидите имя, похожее на "USB Serial Device (COM4)". Запустите выбранную программу для работы с последовательным портом, например PuTTY. В диалоговом окне PuTTY установите "Connection type" в значение "Serial", введите COMx (полученный из

Диспетчера устройств) в поле "Serial line" и нажмите "Open".

29.3. Сетевые интерфейсы в режиме USB-устройства

Поддержка виртуальных сетевых интерфейсов обеспечивается шаблонами 1, 8 и 10. Обратите внимание, что ни один из них не работает с Microsoft Windows. Другие операционные системы хоста работают со всеми тремя шаблонами. Необходимо загрузить оба модуля ядра: usb_template(4) и if_cdce(4).

Убедитесь, что необходимые модули загружены, и правильный шаблон установлен при загрузке, добавив следующие строки в /boot/loader.conf, создав его, если он ещё не существует:

```
if_cdce_load="YES"
hw.usb.template=1
```

Для загрузки модуля и установки шаблона без перезагрузки используйте:

```
# kldload if_cdce
# sysctl hw.usb.template=1
```

29.4. Виртуальное устройство хранения данных USB

Драйвер cfumass(4) — это драйвер режима USB-устройства, впервые появившийся в FreeBSD 12.0.

Целевое устройство хранения данных (Mass Storage) предоставляется шаблонами 0 и 10. Необходимо загрузить оба модуля ядра: usb_template(4) и cfumass(4). Модуль cfumass(4) взаимодействует с подсистемой CTL, той же самой, что используется для целевых устройств iSCSI или Fibre Channel. Со стороны хоста инициаторы USB Mass Storage могут обращаться только к одному LUN — LUN 0.

29.4.1. Настройка USB-накопителя с использованием стартового скрипта cfumass

Самый простой способ настроить USB-накопитель только для чтения — использовать скрипт cfumass. Для этого скопируйте файлы, которые должны быть доступны на USB-хосте, в каталог `/var/cfumass` и добавьте следующую строку в /etc/rc.conf:

```
cfumass_enable="YES"
```

Для настройки цели без перезагрузки выполните следующую команду:

```
# service cfumass start
```

В отличие от последовательных и сетевых функций, шаблон не следует устанавливать в 0 или 10 в /boot/loader.conf. Это связано с тем, что LUN должен быть настроен до установки шаблона. Скрипт запуска cfumass автоматически устанавливает правильный номер шаблона при запуске.

29.4.2. Настройка USB-накопителей с использованием других методов

Оставшаяся часть этой главы содержит подробное описание настройки цели без использования файла `cfumass rc`. Это необходимо, например, если требуется предоставить доступную для записи LUN.

USB-накопитель не требует работы демона ctld(8), хотя его можно использовать при необходимости. Это отличается от iSCSI. Таким образом, есть два способа настройки цели: ctladm(8) или ctld(8). Оба способа требуют загрузки модуля ядра cfumass.ko. Модуль можно загрузить вручную:

```
# kldload cfumass
```

Если модуль cfumass.ko не встроен в ядро, можно настроить /boot/loader.conf для его загрузки при старте системы:

```
cfumass_load="YES"
```

LUN может быть создан без использования демона ctld(8):

```
# ctladm create -b block -o file=/data/target0
```

Это предоставляет содержимое файла образа /data/target0 как LUN для USB-хоста. Файл должен существовать до выполнения команды. Чтобы настроить LUN при загрузке системы, добавьте команду в /etc/rc.local.

ctld(8) также может использоваться для управления LUN. Создайте файл /etc/ctl.conf, добавьте строку в /etc/rc.conf, чтобы убедиться, что ctld(8) автоматически запускается при загрузке, а затем запустите демон.

Вот пример простого файла конфигурации /etc/ctl.conf. Полное описание параметров можно найти в ctl.conf(5).

```
target naa.50015178f369f092 {
    lun 0 {
        path /data/target0
        size 4G
    }
```

```
}
```

Пример создаёт одну цель с одним LUN. `naa.50015178f369f092` — это идентификатор устройства, состоящий из 32 случайных шестнадцатеричных цифр. Строка `path` определяет полный путь к файлу или zvol, который используется для LUN. Этот файл должен существовать до запуска ctld(8). Вторая строка необязательна и указывает размер LUN.

Чтобы убедиться, что демон ctld(8) запускается при загрузке, добавьте следующую строку в /etc/rc.conf:

```
ctld_enable="YES"
```

Чтобы запустить ctld(8) сейчас, выполните следующую команду:

```
# service ctld start
```

При запуске демона ctld(8) он читает файл /etc/ctl.conf. Если этот файл отредактирован после запуска демона, перезагрузите изменения, чтобы они вступили в силу немедленно:

```
# service ctld reload
```

Часть IV: Сетевые коммуникации

FreeBSD является одной из наиболее широко используемых операционных систем для высокопроизводительных сетевых серверов. Главы в этом разделе охватывают:

- Передача данных по последовательному порту
- PPP и PPP через Ethernet
- Электронная почта
- Работа с сетевыми серверами
- Межсетевые экраны
- Другие темы сложной сетевой настройки

Эти главы предназначены для ознакомления по мере необходимости. Их не обязательно читать в определённом порядке, а также нет необходимости прочитывать все из них перед использованием FreeBSD в сетевом окружении.

Глава 30. Последовательная передача данных

30.1. Обзор

UNIX® всегда поддерживал последовательную передачу данных, так как самые первые UNIX®-машины использовали последовательные линии для ввода и вывода информации. Многое изменилось со времен, когда стандартный терминал состоял из последовательного принтера со скоростью печати 10 символов в секунду и клавиатуры. В этой главе рассматриваются некоторые способы использования последовательной передачи данных в FreeBSD.

Прочтите эту главу, чтобы узнать:

- Как подключить терминалы к системе FreeBSD.
- Как использовать модем для дозвона до удалённых хостов.
- Как разрешить удалённым пользователям входить в систему FreeBSD с помощью модема.
- Как загрузить систему FreeBSD с последовательной консоли.

Прежде чем читать эту главу, необходимо:

- Знать, как настроить и установить собственное ядро.
- Понять права и процессы в FreeBSD.
- Иметь доступ к техническому руководству по последовательному оборудованию, которое будет использоваться с FreeBSD.

30.2. Терминология и оборудование для последовательной передачи данных

При связи по последовательному порту часто используются следующие термины:

bps

Биты в секунду (бит/с) — это скорость передачи данных.

DTE

Оборудование передачи данных (Data Terminal Equipment — DTE) — это одна из двух конечных точек в последовательной связи. Примером может служить компьютер.

DCE

Оборудование передачи данных (Data Communication Equipment — DCE) — это другая конечная точка в последовательной связи. Обычно это модем или последовательный терминал.

RS-232

Оригинальный стандарт, определяющий аппаратную последовательную связь. Впоследствии был переименован в TIA-232.

При упоминании скорости передачи данных в этом разделе не используется термин *бод*. Бод обозначает количество изменений электрического состояния за единицу времени, тогда как бит/с (bps) является корректным термином.

Для подключения последовательного терминала к системе FreeBSD необходим последовательный порт на компьютере и соответствующий кабель для соединения с последовательным устройством. Пользователям, уже знакомым с последовательным оборудованием и кабелями, можно смело пропустить этот раздел.

30.2.1. Последовательные кабели и порты

Существует несколько различных типов последовательных кабелей. Два наиболее распространённых типа — это нуль-модемные кабели и стандартные кабели RS-232. В документации к оборудованию должен быть указан тип необходимого кабеля.

Эти два типа кабелей отличаются тем, как провода подключены к разъему. Каждый провод представляет собой сигнал, с определёнными сигналами, перечисленными в Имена сигналов RS-232C. Стандартный последовательный кабель передаёт все сигналы RS-232C напрямую. Например, контакт "Передаваемые данные" на одном конце кабеля соединён с контактом "Передаваемые данные" на другом конце. Такой тип кабеля используется для подключения модема к системе FreeBSD, а также подходит для некоторых терминалов.

Нулевой модемный кабель меняет местами контакт "Передаваемые данные" одного конца с контактом "Принимаемые данные" другого конца. Разъём может быть либо DB-25, либо DB-9.

Нуль-модемный кабель можно изготовить, используя соединения контактов, приведённые в DB-25 — DB-25 Null-Modem Cable, DB-9 — DB-9 Null-Modem Cable и DB-9 — DB-25 Null-Modem Cable, DB-9 — DB-25 Null-Modem Cable. Хотя стандарт требует прямого соединения контакта 1 с контактом 1 («Protective Ground»), эту линию часто не используют. Некоторые терминалы работают только с контактами 2, 3 и 7, тогда как другим требуются иные конфигурации. В случае сомнений обратитесь к документации оборудования.

Таблица 43. Имена сигналов RS-232C

Аббревиатуры	Имена
RD	Received Data
TD	Transmitted Data
DTR	Готовность терминального оборудования (Data Terminal Ready)
DSR	Готовность терминального оборудования (Data Set Ready)
DCD	Обнаружение несущей (Data Carrier Detect)

Аббревиатуры	Имена
SG	Сигнальная земля (Signal Ground)
RTS	Запрос на передачу (Request to Send)
CTS	Готовность к приёму (Clear to Send)

Таблица 44. от DB-25 к DB-25 кабель нуль-модема

Сигнал	# пина		# пина	Сигнал
SG	7	соединяется с	7	SG
TD	2	соединяется с	3	RD
RD	3	соединяется с	2	TD
RTS	4	соединяется с	5	CTS
CTS	5	соединяется с	4	RTS
DTR	20	соединяется с	6	DSR
DTR	20	соединяется с	8	DCD
DSR	6	соединяется с	20	DTR
DCD	8	соединяется с	20	DTR

Таблица 45. от DB-9 к DB-9 кабель нуль-модема

Сигнал	# пина		# пина	Сигнал
RD	2	соединяется с	3	TD
TD	3	соединяется с	2	RD
DTR	4	соединяется с	6	DSR
DTR	4	соединяется с	1	DCD
SG	5	соединяется с	5	SG
DSR	6	соединяется с	4	DTR
DCD	1	соединяется с	4	DTR
RTS	7	соединяется с	8	CTS
CTS	8	соединяется с	7	RTS

Таблица 46. от DB-9 к DB-25 кабель нуль-модема

Сигнал	# пина		# пина	Сигнал
RD	2	соединяется с	2	TD
TD	3	соединяется с	3	RD
DTR	4	соединяется с	6	DSR
DTR	4	соединяется с	8	DCD
SG	5	соединяется с	7	SG

Сигнал	# пина		# пина	Сигнал
DSR	6	соединяется с	20	DTR
DCD	1	соединяется с	20	DTR
RTS	7	соединяется с	5	CTS
CTS	8	соединяется с	4	RTS

Когда один контакт на одном конце соединяется с парой контактов на другом конце, это обычно реализуется с помощью одного короткого провода между парой контактов в их разъёме и длинного провода к другому одиночному контакту.

Последовательные порты — это устройства, через которые данные передаются между хост-компьютером FreeBSD и терминалом. Существует несколько видов последовательных портов. Перед покупкой или изготовлением кабеля убедитесь, что он подходит к портам на терминале и системе FreeBSD.

Большинство терминалов оснащено портами DB-25. Персональные компьютеры могут иметь порты DB-25 или DB-9. Многопортовые последовательные платы могут быть оснащены портами RJ-12 или RJ-45. Для определения типа порта обратитесь к документации, прилагаемой к оборудованию, или визуально проверьте тип порта.

В FreeBSD каждый последовательный порт доступен через запись в /dev. Существует два разных типа записей:

- Поры входящих соединений именуются /dev/ttyuN, где *N* — это номер порта, начиная с нуля. Если терминал подключен к первому последовательному порту (COM1), используйте /dev/ttyu0 для обращения к терминалу. Если терминал находится на втором последовательном порту (COM2), используйте /dev/ttyu1, и так далее. Как правило, порт входящих соединений используется для терминалов. Для корректной работы порта требуется, чтобы последовательная линия подавала сигнал "Data Carrier Detect".
- Порты исходящих соединений называются /dev/cuauN в FreeBSD версий 8.X и выше и /dev/cuadN в FreeBSD версий 7.X и ниже. Порты исходящих соединений обычно не используются для терминалов, но применяются для модемов. Порт может быть использован, если последовательный кабель или терминал не поддерживают сигнал "Data Carrier Detect".

FreeBSD также предоставляет устройства инициализации (/dev/ttyuN.init и /dev/cuauN.init или /dev/cuadN.init) и устройства блокировки (/dev/ttyuN.lock и /dev/cuauN.lock или /dev/cuadN.lock). Устройства инициализации используются для установки параметров порта связи при каждом его открытии, например, `crtscts` для модемов, использующих сигнализацию `RTS/CTS` для управления потоком. Устройства блокировки применяются для фиксации флагов на портах, чтобы предотвратить изменение определённых параметров пользователями или программами. Дополнительную информацию о настройках терминала, блокировке и инициализации устройств, а также установке параметров терминала можно найти в руководствах termios(4), uart(4) и stty(1) соответственно.

30.2.2. Настройка последовательного порта

По умолчанию FreeBSD поддерживает четыре последовательных порта, обычно известных как COM1, COM2, COM3 и COM4. FreeBSD также поддерживает простые многопортовые последовательные интерфейсные карты, такие как BocaBoard 1008 и 2016, а также более интеллектуальные многопортовые карты, например, производства Digiboard. Однако стандартное ядро ищет только порты COM.

Чтобы проверить, распознает ли система последовательные порты, найдите сообщения загрузки системы, начинающиеся с `uart`:

```
# grep uart /var/run/dmesg.boot
```

Если система не распознает все необходимые последовательные порты, в файл /boot/device.hints можно добавить дополнительные записи. Этот файл уже содержит записи `hint.uart.0.*` для COM1 и `hint.uart.1.*` для COM2. При добавлении записи для порта COM3 используйте `0x3E8`, а для COM4 — `0x2E8`. Обычные адреса IRQ: `5` для COM3 и `9` для COM4.

Для определения стандартных настроек терминального ввода-вывода, используемых портом, укажите имя его устройства. В этом примере определяются настройки для порта вызова на COM2:

```
# stty -a -f /dev/ttyu1
```

Системная инициализация последовательных устройств управляется файлом /etc/rc.d/serial. Этот файл влияет на настройки по умолчанию для последовательных устройств. Чтобы изменить настройки устройства, используйте команду `stty`. По умолчанию изменённые настройки действуют до закрытия устройства, а при повторном открытии устройства возвращаются значения по умолчанию. Чтобы навсегда изменить настройки по умолчанию, откройте и настройте параметры инициализационного устройства. Например, чтобы включить режим `CLOCAL`, 8-битную передачу данных и управление потоком `XON/XOFF` для устройства ttyu5, введите:

```
# stty -f /dev/ttyu5.init clocal cs8 ixon ixoff
```

Чтобы предотвратить изменение определённых настроек приложением, внесите изменения в блокирующее устройство. Например, чтобы зафиксировать скорость ttyu5 на уровне 57600 бод, введите:

```
# stty -f /dev/ttyu5.lock 57600
```

Теперь любое приложение, открывающее ttyu5 и пытающееся изменить скорость порта, будет ограничено скоростью 57600 бод.

30.3. Терминалы

Терминалы обеспечивают удобный и недорогой способ доступа к системе FreeBSD, когда пользователь не находится за консолью компьютера или в подключённой сети. В этом разделе описывается, как использовать терминалы с FreeBSD.

Оригинальные системы UNIX® не имели консолей. Вместо этого пользователи входили в систему и запускали программы через терминалы, подключённые к последовательным портам компьютера.

Возможность установить сеанс входа через последовательный порт до сих пор присутствует практически во всех UNIX®-подобных операционных системах, включая FreeBSD. Подключив терминал к неиспользуемому последовательному порту, пользователь может войти в систему и запускать любые текстовые программы, которые обычно можно запустить на консоли или в окне `xterm`.

Многие терминалы могут быть подключены к системе FreeBSD. Старый запасной компьютер можно использовать в качестве терминала, подключённого к более мощному компьютеру с FreeBSD. Это позволяет превратить, казалось бы, однопользовательский компьютер в мощную многопользовательскую систему.

FreeBSD поддерживает три типа терминалов:

Простые терминалы

Простые терминалы — это специализированные аппаратные устройства, подключаемые к компьютерам через последовательные линии. Они называются простыми, потому что обладают лишь достаточной вычислительной мощностью для отображения, отправки и приёма текста. На этих устройствах нельзя запускать программы. Вместо этого простые терминалы подключаются к компьютеру, на котором выполняются необходимые программы.

Существуют сотни видов простых терминалов, выпускаемых разными производителями, и практически любой из них будет работать с FreeBSD. Некоторые продвинутые терминалы даже поддерживают отображение графики, но только определённые программные пакеты могут использовать эти дополнительные возможности.

Простые терминалы популярны в рабочих средах, где сотрудникам не нужен доступ к графическим приложениям.

Компьютеры, выступающие в качестве терминалов

Поскольку простой терминал обладает лишь минимальными возможностями для отображения, отправки и получения текста, в его роли может выступать практически любой компьютер. Всё, что для этого требуется, — это подходящий кабель и *программное обеспечение для эмуляции терминала*, запущенное на компьютере.

Такая конфигурация может быть полезной. Например, если один пользователь занят работой за консолью системы FreeBSD, другой пользователь может одновременно выполнять текстовые задачи на менее мощном персональном компьютере, подключённом как терминал к системе FreeBSD.

В базовой системе FreeBSD есть как минимум две утилиты для работы через последовательное соединение: cu(1) и tip(1).

Например, чтобы подключиться с клиентской системы под управлением FreeBSD к последовательному соединению другой системы:

```
# cu -l /dev/cuauN
```

Порты нумеруются начиная с нуля. Это означает, что COM1 соответствует /dev/cuau0.

Дополнительные программы доступны через Коллекцию портов, например, comms/minicom.

X Терминалы

X терминалы представляют собой наиболее продвинутый тип терминалов. Вместо подключения к последовательному порту они обычно работают через сеть, такую как Ethernet. В отличие от ограниченных текстовыми приложениями терминалов, они способны отображать любые приложения Xorg.

Эта глава не охватывает настройку, конфигурацию или использование X терминалов.

30.3.1. Настройка терминала

В этом разделе описывается, как настроить систему FreeBSD для включения сеанса входа через последовательный терминал. Предполагается, что система распознает последовательный порт, к которому подключен терминал, и что терминал подключен с помощью правильного кабеля.

В FreeBSD `init` читает файл /etc/ttys и запускает процесс `getty` на доступных терминалах. Процесс `getty` отвечает за чтение имени пользователя и запуск программы `login`. Порты в системе FreeBSD, которые разрешают вход в систему, перечислены в файле /etc/ttys. Например, первая виртуальная консоль ttyv0 имеет запись в этом файле, что разрешает вход на консоли. Этот файл также содержит записи для других виртуальных консолей, последовательных портов и псевдо-терминалов. Для физического терминала последовательный порт указывается как запись в /dev без части `/dev`. Например, /dev/ttyv0 указывается как `ttyv0`.

Файл /etc/ttys по умолчанию настраивает поддержку первых четырёх последовательных портов, от ttyu0 до ttyu3:

```
ttyu0   "/usr/libexec/getty std.115200"   dialup  off secure
ttyu1   "/usr/libexec/getty std.115200"   dialup  off secure
ttyu2   "/usr/libexec/getty std.115200"   dialup  off secure
ttyu3   "/usr/libexec/getty std.115200"   dialup  off secure
```

При подключении терминала к одному из этих портов измените запись по умолчанию, чтобы установить необходимую скорость и тип терминала, включить устройство (`on`) и,

если требуется, изменить настройку `secure` для порта. Если терминал подключен к другому порту, добавьте запись для этого порта.

В Настройка записей терминалов настраиваются два терминала в /etc/ttys. Первая запись настраивает терминал Wyse-50, подключённый к COM2. Вторая запись настраивает старый компьютер с программным обеспечением терминала Procomm, эмулирующим терминал VT-100. Компьютер подключён к шестому последовательному порту многопортовой последовательной платы.

Пример 40. Настройка записей терминалов

```
ttyu1  "/usr/libexec/getty std.38400"  wy50   on insecure
ttyu5   "/usr/libexec/getty std.19200"  vt100  on insecure
```

Первое поле указывает имя устройства последовательного терминала.

Второе поле указывает `getty` инициализировать и открыть линию, установить скорость линии, запросить имя пользователя и затем выполнить программу `login`. Необязательный *mun getty* настраивает характеристики линии терминала, такие как скорость передачи и контроль чётности. Доступные типы getty перечислены в /etc/gettytab. Почти во всех случаях подходят типы getty, начинающиеся с `std`, так как эти записи игнорируют контроль чётности. Для каждой скорости передачи от 110 до 115200 существует запись `std`. Дополнительную информацию можно найти в gettytab(5). При настройке типа getty убедитесь, что он соответствует параметрам связи, используемым терминалом. В данном примере Wyse-50 работает без контроля чётности и подключается на скорости 38400 бод. Компьютер также работает без контроля чётности и подключается на скорости 19200 бод.

Третье поле указывает тип терминала. Для коммутируемых портов обычно используется `unknown` или `dialup`, так как пользователи могут подключаться практически с любым типом терминала или программного обеспечения. Поскольку тип терминала не меняется для проводных терминалов, можно указать реальный тип терминала из /etc/termcap. В данном примере для Wyse-50 используется реальный тип терминала, а для компьютера с Procomm установлена эмуляция VT-100.

Четвёртое поле определяет, должен ли порт быть включён. Чтобы разрешить входы через этот порт, в поле должно быть указано значение `on`.

Последнее поле используется для указания, является ли порт безопасным. Пометка порта как `secure` означает, что он считается достаточно доверенным, чтобы разрешить вход `root` с этого порта. Небезопасные порты не позволяют вход под `root`. На небезопасном порту пользователи должны входить с непривилегированных учётных записей, а затем использовать `su` или аналогичный механизм для получения прав суперпользователя, как описано в “Учётная запись суперпользователя”. В целях безопасности рекомендуется изменить этот параметр на `insecure`.

После внесения изменений в /etc/ttys отправьте сигнал SIGHUP (завершения работы)

процессу `init`, чтобы заставить его перечитать конфигурационный файл:

```
# kill -HUP 1
```

Поскольку `init` всегда является первым процессом, запускаемым в системе, его идентификатор процесса всегда равен `1`.

Если всё настроено правильно, все кабели подключены, и терминалы включены, процесс `getty` должен быть запущен на каждом терминале, и на каждом терминале должна быть доступна приглашение для входа в систему.

30.3.2. Устранение проблем с подключением

Даже при самом тщательном внимании к деталям что-то может пойти не так при настройке терминала. Вот список распространённых симптомов и некоторые предлагаемые решения.

Если приглашение для входа не появляется, убедитесь, что терминал подключен и включен. Если в качестве терминала используется персональный компьютер, убедитесь, что на нём запущено программное обеспечение эмуляции терминала на правильном последовательном порту.

Убедитесь, что кабель надёжно подключен как к терминалу, так и к компьютеру с FreeBSD. Убедитесь, что используется правильный тип кабеля.

Убедитесь, что терминал и FreeBSD используют одинаковые скорость (bps) и настройки чётности. Для видеотерминала убедитесь, что регуляторы контрастности и яркости включены. Если это печатающий терминал, проверьте наличие бумаги и чернил.

Используйте `ps`, чтобы убедиться, что процесс `getty` запущен и обслуживает терминал. Например, следующий вывод показывает, что `getty` работает на втором последовательном порту, ttyu1, и использует запись `std.38400` в файле /etc/gettytab:

```
# ps -axww|grep ttyu
22189  d1  Is+    0:00.03 /usr/libexec/getty std.38400 ttyu1
```

Если процесс `getty` не запущен, убедитесь, что порт включен в файле /etc/ttys. Не забудьте выполнить `kill -HUP 1` после изменения файла /etc/ttys.

Если процесс `getty` работает, но терминал по-прежнему не отображает приглашение для входа, или если он отображает приглашение, но не принимает вводимые данные, возможно, терминал или кабель не поддерживают аппаратное подтверждение связи. Попробуйте изменить запись в /etc/ttys с `std.38400` на `3wire.38400`, затем выполните `kill -HUP 1` после изменения /etc/ttys. Запись `3wire` аналогична `std`, но игнорирует аппаратное подтверждение связи. Также может потребоваться уменьшить скорость передачи данных или включить программное управление потоком при использовании `3wire`, чтобы избежать переполнения буфера.

Если вместо приглашения к входу в систему появляется мусор, убедитесь, что терминал и FreeBSD используют одинаковые скорость (bps) и настройки чётности. Проверьте процессы `getty`, чтобы убедиться, что используется правильный тип *getty*. Если это не так, отредактируйте файл /etc/ttys и выполните команду `kill -HUP 1`.

Если символы отображаются удвоенными, а пароль виден при вводе, переключите терминал или программу эмуляции терминала из режима «полудуплексный» или «локальный эхо-контроль» в режим «полнодуплексный».

30.4. Входящие соединения по модему

Настройка системы FreeBSD для предоставления входящих соединений аналогична настройке терминалов, за исключением того, что вместо терминальных устройств используются модемы. FreeBSD поддерживает как внешние, так и внутренние модемы.

Внешние модемы удобнее, так как их часто можно настроить с помощью параметров, хранящихся в энергонезависимой памяти, и они обычно оснащены световыми индикаторами, отображающими состояние важных сигналов RS-232, что позволяет определить, работает ли модем правильно.

Внутренние модемы обычно не имеют энергонезависимой памяти, поэтому их настройка может ограничиваться установкой перемычек. Если у внутреннего модема есть индикаторные лампы, их трудно увидеть, когда корпус системы закрыт.

При использовании внешнего модема необходим соответствующий кабель. Достаточно стандартного последовательного кабеля RS-232C.

FreeBSD требует сигналов RTS и CTS для управления потоком данных на скоростях выше 2400 бит/с, сигнала CD для определения ответа на вызов или завершения соединения, а также сигнала DTR для сброса модема после завершения сеанса. Некоторые кабели подключены без всех необходимых сигналов, поэтому, если сеанс входа в систему не завершается при разрыве соединения, проблема может быть в кабеле. Дополнительную информацию об этих сигналах можно найти в разделе Последовательные кабели и порты.

Как и другие UNIX®-подобные операционные системы, FreeBSD использует аппаратные сигналы для определения ответа на вызов или завершения соединения, а также для сброса и разрыва соединения модема после вызова. FreeBSD избегает отправки команд модему или отслеживания его статусных отчётов.

FreeBSD поддерживает интерфейсы связи RS-232C (CCITT V.24) на основе NS8250, NS16450, NS16550 и NS16550A. Устройства 8250 и 16450 имеют однобайтовые буферы. Устройство 16550 предоставляет 16-байтовый буфер, что позволяет повысить производительность системы. Ошибки в обычных чипах 16550 не позволяют использовать 16-байтовый буфер, поэтому по возможности следует использовать устройства 16550A. Поскольку устройства с однобайтовым буфером требуют больше работы от операционной системы, чем устройства с 16-байтовым буфером, предпочтительнее использовать последовательные интерфейсные карты на основе 16550A. Если в системе много активных последовательных портов или ожидается высокая нагрузка, карты на основе 16550A обеспечивают более надёжную связь с низким уровнем ошибок.

Оставшаяся часть этого раздела демонстрирует, как настроить модем для приема входящих соединений, как взаимодействовать с модемом, а также предлагает несколько советов по устранению неполадок.

30.4.1. Настройка модема

Как и в случае с терминалами, `init` запускает процесс `getty` для каждого настроенного последовательного порта, используемого для входящих соединений. Когда пользователь дозванивается по линии модема и модемы соединяются, модем сообщает о сигнале "Carrier Detect". Ядро обнаруживает наличие несущей и указывает `getty` открыть порт и вывести приглашение `login:` с указанной начальной скоростью линии. В типичной конфигурации, если получены некорректные символы (обычно из-за несоответствия скорости соединения модема и настроенной скорости), `getty` пытается изменить скорость линии, пока не получит разборчивые символы. После того как пользователь вводит имя для входа, `getty` запускает `login`, который завершает процесс входа, запрашивая пароль пользователя и затем запуская его оболочку.

Существует два подхода к настройке модемов для коммутируемого доступа. Первый метод заключается в том, чтобы настроить модемы и систему таким образом, что независимо от скорости, с которой удалённый пользователь подключается, скорость интерфейса RS-232 остаётся фиксированной. Преимущество такой настройки в том, что удалённый пользователь всегда сразу видит приглашение системы к вводу логина. Недостаток же состоит в том, что система не знает реальной скорости передачи данных пользователя, поэтому полноэкранные программы, такие как Emacs, не могут адаптировать методы отрисовки экрана для улучшения отклика на медленных соединениях.

Второй способ заключается в настройке интерфейса RS-232 для изменения скорости в зависимости от скорости соединения удалённого пользователя. Поскольку `getty` не понимает отчёты о скорости соединения конкретного модема, он выводит сообщение `login:` на начальной скорости и отслеживает символы, которые приходят в ответ. Если пользователь видит непонятные символы, он должен нажать `Enter`, пока не увидит узнаваемое приглашение. Если скорости передачи данных не совпадают, `getty` воспринимает введенные пользователем символы как непонятные, переключается на следующую скорость и снова выводит приглашение `login:`. Обычно для появления корректного приглашения достаточно нажать одну-две клавиши. Этот процесс входа выглядит не так аккуратно, как метод с фиксированной скоростью, но пользователь с низкоскоростным соединением получит лучшую интерактивную реакцию от полноэкранных программ.

При фиксации скорости передачи данных модема на определённом значении изменения в файле /etc/gettytab обычно не требуются. Однако для конфигурации с согласованием скоростей могут понадобиться дополнительные записи, чтобы определить скорости, используемые для модема. В этом примере настраивается модем 14,4 Кбит/с с максимальной скоростью интерфейса 19,2 Кбит/с при 8-битных соединениях без контроля чётности. Здесь `getty` настраивается на начало связи для соединения V.32bis со скоростью 19,2 Кбит/с, затем перебирает скорости 9600 бит/с, 2400 бит/с, 1200 бит/с, 300 бит/с и снова возвращается к 19,2 Кбит/с. Перебор скоростей связи реализуется с помощью возможности `nx=` (следующая таблица). Каждая строка использует запись `tc=` (продолжение таблицы) для

наследования остальных параметров для конкретной скорости передачи данных.

```
#
# Additions for a V.32bis Modem
#
um|V300|High Speed Modem at 300,8-bit:\
        :nx=V19200:tc=std.300:
un|V1200|High Speed Modem at 1200,8-bit:\
        :nx=V300:tc=std.1200:
uo|V2400|High Speed Modem at 2400,8-bit:\
        :nx=V1200:tc=std.2400:
up|V9600|High Speed Modem at 9600,8-bit:\
        :nx=V2400:tc=std.9600:
uq|V19200|High Speed Modem at 19200,8-bit:\
        :nx=V9600:tc=std.19200:
```

Для модема со скоростью 28,8 Кбит/с или для использования сжатия на модеме 14,4 Кбит/с следует установить более высокую скорость передачи данных, как показано в этом примере:

```
#
# Additions for a V.32bis or V.34 Modem
# Starting at 57.6 Kbps
#
vm|VH300|Very High Speed Modem at 300,8-bit:\
        :nx=VH57600:tc=std.300:
vn|VH1200|Very High Speed Modem at 1200,8-bit:\
        :nx=VH300:tc=std.1200:
vo|VH2400|Very High Speed Modem at 2400,8-bit:\
        :nx=VH1200:tc=std.2400:
vp|VH9600|Very High Speed Modem at 9600,8-bit:\
        :nx=VH2400:tc=std.9600:
vq|VH57600|Very High Speed Modem at 57600,8-bit:\
        :nx=VH9600:tc=std.57600:
```

Для медленного процессора или сильно загруженной системы без последовательных портов на базе 16550A такая конфигурация может вызывать ошибки "silo" в `uart` на скорости 57,6 Кбит/с.

Конфигурация /etc/ttys аналогична Конфигурация записей терминалов, но в `getty` передаётся другой аргумент, а в качестве типа терминала используется `dialup`. Замените *xxx* на процесс, который `init` будет запускать на устройстве:

```
ttyu0   "/usr/libexec/getty xxx"   dialup on
```

Тип терминала `dialup` можно изменить. Например, установка `vt102` в качестве типа терминала по умолчанию позволяет пользователям использовать эмуляцию VT102 на их

удалённых системах.

Для конфигурации с фиксированной скоростью укажите скорость с допустимым типом, перечисленным в /etc/gettytab. В этом примере показана настройка для модема, скорость порта которого зафиксирована на 19,2 Кбит/с:

```
ttyu0   "/usr/libexec/getty std.19200"   dialup on
```

В конфигурации с согласованной скоростью запись должна ссылаться на соответствующую начальную запись "автонастройки скорости" в /etc/gettytab. Чтобы продолжить пример для модема с согласованной скоростью, начинающего работу с 19,2 Кбит/с, используйте следующую запись:

```
ttyu0   "/usr/libexec/getty V19200"   dialup on
```

После редактирования /etc/ttys дождитесь, пока модем будет правильно настроен и подключен, прежде чем подавать сигнал `init`:

```
# kill -HUP 1
```

Высокоскоростные модемы, такие как V.32, V.32bis и V.34, используют аппаратное управление потоком (`RTS/CTS`). Используйте `stty` для установки флага аппаратного управления потоком для порта модема. В этом примере устанавливается флаг `crtscts` для устройств инициализации входящих и исходящих соединений на COM2:

```
# stty -f /dev/ttyu1.init crtscts
# stty -f /dev/cuau1.init crtscts
```

30.4.2. Устранение неполадок

В этом разделе приведены несколько советов по устранению неполадок с модемом для коммутируемого доступа, который не подключается к системе FreeBSD.

Подключите модем к системе FreeBSD и загрузите систему. Если на модеме есть индикаторы состояния, следите за тем, загорается ли индикатор DTR модема при появлении приглашения `login:` на консоли системы. Если он загорается, это должно означать, что FreeBSD запустила процесс `getty` на соответствующем коммуникационном порту и ожидает принятия вызова модемом.

Если индикатор DTR не горит, войдите в систему FreeBSD через консоль и введите `ps ax`, чтобы проверить, запущен ли процесс `getty` на правильном порту в FreeBSD:

```
  114 ??  I      0:00.10 /usr/libexec/getty V19200 ttyu0
```

Если во втором столбце указано `d0` вместо `??` и модем ещё не принял звонок, это означает, что `getty` завершил открытие коммуникационного порта. Это может указывать на проблему с кабелем или неправильную настройку модема, так как `getty` не должен иметь возможности открыть коммуникационный порт до тех пор, пока модем не подал сигнал обнаружения несущей.

Если нет процессов `getty`, ожидающих открытия порта, дважды проверьте, что запись для порта корректна в /etc/ttys. Также проверьте /var/log/messages, чтобы увидеть, есть ли какие-либо сообщения в логе от `init` или `getty`.

Далее попробуйте подключиться к системе. Убедитесь, что на удалённой системе установлены 8 бит, отсутствие контроля чётности и 1 стоповый бит. Если приглашение не появляется сразу или отображается как бессмыслица, попробуйте нажимать `Enter` примерно раз в секунду. Если приглашение `login:` так и не появилось, попробуйте отправить `BREAK`. При использовании высокоскоростного модема попробуйте набрать номер ещё раз после фиксации скорости интерфейса набирающего модема.

Если приглашение `login:` по-прежнему не появляется, проверьте /etc/gettytab ещё раз и убедитесь, что:

- Изначальное название возможности, указанное в записи файла /etc/ttys, соответствует названию возможности в файле /etc/gettytab.
- Каждая запись `nx=` соответствует другому имени возможности в gettytab.
- Каждая запись `tc=` соответствует другому имени возможности в gettytab.

Если модем в системе FreeBSD не отвечает, убедитесь, что он настроен на ответ на звонок при активном сигнале DTR. Если модем, кажется, настроен правильно, проверьте, что линия DTR активна, по индикаторам модема.

Если это по-прежнему не работает, попробуйте отправить письмо в Список рассылки, посвящённый вопросам и ответам пользователей FreeBSD с описанием модема и возникшей проблемы.

30.5. Исходящие соединения по модему

Вот несколько советов по подключению хоста через модем к другому компьютеру. Это подходит для установки терминального сеанса с удалённым хостом.

Такой тип соединения может быть полезен для получения файла из Интернета, если возникли проблемы с использованием PPP. Если PPP не работает, используйте терминальную сессию для FTP-загрузки нужного файла. Затем воспользуйтесь zmodem для его передачи на машину.

30.5.1. Использование стандартного модема Hayes

В `tip` встроен стандартный Hayes-совместимый наборщик номера. Используйте `at=hayes` в /etc/remote.

Драйвер Hayes недостаточно интеллектуален, чтобы распознать некоторые расширенные

функции современных модемов, такие как сообщения BUSY, NO DIALTONE или CONNECT 115200. Отключите эти сообщения при использовании tip с помощью команды ATX0&W.

Таймаут набора номера для tip составляет 60 секунд. Модем должен использовать значение меньше, иначе tip решит, что возникли проблемы со связью. Попробуйте ATS7=45&W.

30.5.2. Использование команд AT

Создайте запись "direct" в файле /etc/remote. Например, если модем подключён к первому последовательному порту, /dev/cuau0, используйте следующую строку:

```
cuau0:dv=/dev/cuau0:br#19200:pa=none
```

Используйте наибольшую скорость передачи данных, которую поддерживает модем, в параметре br. Затем введите tip cuau0 для подключения к модему.

Или используйте cu от имени root с следующей командой:

```
# cu -lline -sspeed
```

Строка *line* указывает на последовательный порт, например, /dev/cuau0, а *speed* — это скорость, например, 57600. После завершения ввода AT-команд введите ~. для выхода.

30.5.3. Знак @ не работает

Знак @ в параметре номера телефона указывает tip искать номер в файле /etc/phones. Однако, знак @ также является специальным символом в файлах параметров, таких как /etc/remote, поэтому его необходимо экранировать обратной косой чертой:

```
pn=\@
```

30.5.4. Набор номера из командной строки

Поместите запись "generic" в /etc/remote. Например:

```
tip115200|Dial any phone number at 115200 bps:\
        :dv=/dev/cuau0:br#115200:at=hayes:pa=none:du:
tip57600|Dial any phone number at 57600 bps:\
        :dv=/dev/cuau0:br#57600:at=hayes:pa=none:du:
```

Это должно теперь работать:

```
# tip -115200 5551234
```

Пользователи, предпочитающие `cu` вместо `tip`, могут использовать общую запись `cu`:

```
cu115200|Use cu to dial any number at 115200bps:\
        :dv=/dev/cuau1:br#57600:at=hayes:pa=none:du:
```

и введите:

```
# cu 5551234 -s 115200
```

30.5.5. Установка скорости в бодах

Добавьте запись для `tip1200` или `cu1200`, но используйте подходящую скорость передачи (bps) с помощью возможности `br`. `tip` считает хорошим значением по умолчанию 1200 bps, поэтому ищет запись `tip1200`. Однако, не обязательно использовать именно 1200 bps.

30.5.6. Доступ к нескольким хостам через терминальный сервер

Вместо того чтобы каждый раз ждать подключения и вводить `CONNECT хост`, используйте возможность `cm` в `tip`. Например, следующие записи в /etc/remote позволят набрать `tip pain` или `tip muffin` для подключения к хостам `pain` или `muffin`, а `tip deep13` — для подключения к терминальному серверу.

```
pain|pain.deep13.com|Forrester's machine:\
        :cm=CONNECT pain\n:tc=deep13:
muffin|muffin.deep13.com|Frank's machine:\
        :cm=CONNECT muffin\n:tc=deep13:
deep13:Gizmonics Institute terminal server:\
        :dv=/dev/cuau2:br#38400:at=hayes:du:pa=none:pn=5551234:
```

30.5.7. Использование более одного телефонного номера с `tip`

Это часто оказывается проблемой, когда в университете есть несколько модемных линий и несколько тысяч студентов, пытающихся до них дозвониться.

Сделайте запись в /etc/remote и используйте `@` для возможности `pn`:

```
big-university:\
        :pn=\@:tc=dialout
dialout:\
        :dv=/dev/cuau3:br#9600:at=courier:du:pa=none:
```

Затем перечислите номера телефонов в /etc/phones:

```
big-university 5551111
big-university 5551112
```

```
big-university 5551113
big-university 5551114
```

`tip` будет пробовать каждый номер в указанном порядке, затем завершит работу. Для повторных попыток запустите `tip` в цикле `while`.

30.5.8. Использование управляющего символа

По умолчанию `Ctrl` + `P` — это символ "принуждения", используемый для указания `tip`, что следующий символ является буквальными данными. Символ принуждения можно изменить на любой другой с помощью экранирования `~s`, что означает "установить переменную".

Введите `~sforce=одиночный-символ`, затем нажмите Enter. *одиночный-символ* — это любой одиночный символ. Если *одиночный-символ* не указан, то управляющий символ будет нулевым символом, который вводится с помощью `Ctrl` + `2` или `Ctrl` + `Пробел`. Хорошим значением для *одиночный-символ* может быть `Shift` + `Ctrl` + `6`, который используется только на некоторых терминальных серверах.

Чтобы изменить управляющий символ, укажите следующее в ~/.tiprc:

```
force=single-char
```

30.5.9. Верхний регистр символов

Это происходит при нажатии `Ctrl` + `A`, что является «raise character» в `tip`, специально предназначенным для людей с неработающими клавишами caps-lock. Используйте `~s`, чтобы установить `raisechar` в разумное значение. Его можно установить таким же, как и «force character», если ни одна из функций не используется.

Вот пример файла ~/.tiprc для пользователей Emacs, которым нужно вводить `Ctrl` + `2` и `Ctrl` + `A`:

```
force=^^
raisechar=^^
```

`^^` — это `Shift` + `Ctrl` + `6`.

30.5.10. Копирование файлов с помощью `tip`

При обмене данными с другой UNIX®-подобной операционной системой файлы можно отправлять и получать с помощью команд `~p` (put) и `~t` (take). Эти команды выполняют `cat` и `echo` на удалённой системе для приёма и отправки файлов. Синтаксис следующий: `~p` локальный-файл [удалённый-файл] `~t` удалённый-файл [локальный-файл]

Проверка ошибок отсутствует, поэтому следует использовать другой протокол, например, zmodem.

30.5.11. Как использовать zmodem с `tip`?

Для получения файлов запустите программу отправки на удалённой стороне. Затем введите `~C rz`, чтобы начать их получение локально.

Для отправки файлов запустите программу приема на удалённой стороне. Затем введите `~C sz файлы`, чтобы отправить их на удалённую систему.

30.6. Настройка последовательной консоли

FreeBSD поддерживает возможность загрузки системы с использованием простого терминала на последовательном порту в качестве консоли. Такая конфигурация полезна для системных администраторов, которые хотят установить FreeBSD на машины без подключённых клавиатуры или монитора, а также для разработчиков, отлаживающих ядро или драйверы устройств.

Как описано в Процесс загрузки FreeBSD, FreeBSD использует трёхэтапную загрузку. Первые два этапа находятся в коде загрузочного блока, который хранится в начале раздела FreeBSD на загрузочном диске. Затем загрузочный блок загружает и запускает загрузчик в качестве кода третьего этапа.

Для настройки загрузки с последовательной консоли необходимо настроить код загрузочного блока, код загрузчика и ядро.

30.6.1. Быстрая настройка последовательной консоли

В этом разделе представлено краткое описание настройки последовательной консоли. Данная процедура может быть использована, когда к COM1 подключён алфавитно-цифровой терминал.

Процедура: Настройка последовательной консоли на COM1

1. Подключите последовательный кабель к COM1 и управляющему терминалу.
2. Для настройки вывода загрузочных сообщений на последовательный консоль выполните следующую команду от имени суперпользователя:

```
# echo 'console="comconsole"' >> /boot/loader.conf
```

3. Отредактируйте файл /etc/ttys и измените `off` на `on`, а `dialup` на `vt100` для записи ttyu0. В противном случае, для подключения через последовательную консоль не потребуется пароль, что создаст потенциальную уязвимость безопасности.
4. Перезагрузите систему, чтобы проверить, вступили ли изменения в силу.

Если требуется другая конфигурация, обратитесь к следующему разделу для более подробного объяснения настройки.

30.6.2. Углубленная настройка последовательной консоли

В этом разделе представлено более подробное объяснение шагов, необходимых для настройки последовательной консоли в FreeBSD.

Процедура: Настройка последовательной консоли

1. Подготовьте последовательный кабель.

   Используйте нуль-модемный кабель либо стандартный последовательный кабель с нуль-модемным адаптером. Подробнее о последовательных кабелях см. в разделе Последовательные кабели и порты.

2. Отключите клавиатуру.

   Многие системы проверяют наличие клавиатуры во время самотестирования при включении (POST) и выдают ошибку, если клавиатура не обнаружена. Некоторые компьютеры отказываются загружаться, пока клавиатура не будет подключена.

   Если компьютер сообщает об ошибке, но всё же загружается, дальнейшая настройка не требуется.

   Если компьютер отказывается загружаться без подключённой клавиатуры, настройте BIOS так, чтобы он игнорировал эту ошибку. Подробности о том, как это сделать, можно найти в руководстве к материнской плате.

   Попробуйте установить значение "Не установлена" для клавиатуры в BIOS. Этот параметр указывает BIOS не проверять наличие клавиатуры при включении, поэтому он не должен выдавать ошибку при её отсутствии. Если такой опции нет в BIOS, поищите параметр "Останов при ошибке". Установка его в значение "Все, кроме клавиатуры" или "Нет ошибок" даст тот же эффект.

   Если в системе есть мышь PS/2®, её также следует отключить. Мыши PS/2® используют общее оборудование с клавиатурой, и если оставить мышь подключённой, это может ввести проверку клавиатуры в заблуждение, заставив её считать, что клавиатура всё ещё подключена.

   Хотя большинство систем загружаются без клавиатуры, многие не загрузятся без графического адаптера. Некоторые системы можно настроить на загрузку без графического адаптера, изменив параметр "графический адаптер" в конфигурации BIOS на "Не установлен". Другие системы не поддерживают эту опцию и откажутся загружаться, если в системе нет оборудования для вывода изображения. На таких машинах следует оставить какой-либо графический адаптер, даже если это просто старая монохромная плата. Монитор подключать не обязательно.

3. Подключите простой терминал, старый компьютер с модемной программой или последовательный порт другого UNIX®-компьютера к последовательному порту.

4. Добавьте соответствующие записи `hint.uart.*` в файл /boot/device.hints для

последовательного порта. Некоторые многопортовые карты также требуют настройки параметров ядра. Рекомендуемые параметры и подсказки для устройств каждого поддерживаемого последовательного порта смотрите в uart(4).

5. Создайте boot.config в корневом каталоге раздела a на загрузочном диске.

   Этот файл указывает коду загрузочного блока, как загружать систему. Для активации последовательной консоли необходима одна или несколько следующих опций. При использовании нескольких опций включите их все в одну строку:

   `-h`

   Переключает между внутренней и последовательной консолями. Используйте это для смены устройств консоли. Например, для загрузки с внутренней (видео) консоли используйте `-h`, чтобы указать загрузчику и ядру использовать последовательный порт в качестве устройства консоли. Или же, для загрузки с последовательного порта, используйте `-h`, чтобы указать загрузчику и ядру использовать видеодисплей в качестве консоли.

   `-D`

   Переключает между одноконсольной и двухконсольной конфигурациями. В одноконсольной конфигурации консолью будет либо внутренняя консоль (видеодисплей), либо последовательный порт, в зависимости от состояния `-h`. В двухконсольной конфигурации и видеодисплей, и последовательный порт одновременно становятся консолью, независимо от состояния `-h`. Однако двухконсольная конфигурация действует только во время работы загрузочного блока. Как только загрузчик получает управление, консоль, указанная с помощью `-h`, становится единственной консолью.

   `-P`

   Заставляет загрузочный блок проверять наличие клавиатуры. Если клавиатура не обнаружена, опции `-D` и `-h` устанавливаются автоматически.

   Из-за ограничений места в текущей версии загрузочных блоков, `-P` способен обнаруживать только расширенные клавиатуры. Клавиатуры с менее чем 101 клавишей и без клавиш F11 и F12 могут не быть обнаружены. Клавиатуры на некоторых ноутбуках могут не находиться корректно из-за этого ограничения. Если это так, не используйте `-P`.

   Используйте `-P` для автоматического выбора консоли или `-h` для активации последовательной консоли. Подробности смотрите в boot(8) и boot.config(5).

   Параметры, за исключением `-P`, передаются загрузчику. Загрузчик определит, должна ли внутренняя видеосистема или последовательный порт быть консолью, анализируя состояние параметра `-h`. Это означает, что если указан `-D`, но параметр `-h` не указан в /boot.config, последовательный порт может использоваться в качестве консоли только во время загрузки блока, так как загрузчик будет использовать внутренний видеоадаптер в качестве консоли.

6. Загрузите компьютер.

   При загрузке FreeBSD загрузочные блоки выводят содержимое файла /boot.config на консоль. Например:

   ```
   /boot.config: -P
   Keyboard: no
   ```

   Вторая строка появляется только при наличии `-P` в /boot.config и указывает на наличие или отсутствие клавиатуры. Эти сообщения выводятся на последовательный или внутренний консоль, или на оба, в зависимости от опции в /boot.config:

Опции	**Сообщение отправляется на**
none	внутреннюю консоль
`-h`	последовательную консоль
`-D`	последовательную и внутреннюю консоли
`-Dh`	последовательную и внутреннюю консоли
`-P`, клавиатура подключена	внутреннюю консоль
`-P`, клавиатура отсутствует	последовательную консоль

   После сообщения будет небольшая пауза перед тем, как загрузочные блоки продолжат загрузку и выполнение загрузчика и до вывода следующих сообщений на консоль. В обычных условиях нет необходимости прерывать работу загрузочных блоков, но это можно сделать, чтобы убедиться, что всё настроено правильно.

   Нажмите любую клавишу, кроме `Enter`, на консоли, чтобы прервать процесс загрузки. Затем загрузочные блоки запросят дальнейшие действия:

   ```
   >> FreeBSD/i386 BOOT
   Default: 0:ad(0,a)/boot/loader
   boot:
   ```

   Убедитесь, что указанное сообщение появилось на последовательной или внутренней консоли, или на обоих, в соответствии с параметрами в /boot.config. Если сообщение появилось на правильной консоли, нажмите `Enter` для продолжения процесса загрузки.

   Если на последовательном терминале нет приглашения, значит, что-то не так с настройками. Введите `-h`, затем `Enter` или `Return`, чтобы указать загрузочному блоку (а затем загрузчику и ядру) выбрать последовательный порт для консоли. После загрузки системы вернитесь и проверьте, что пошло не так.

На третьем этапе процесса загрузки можно по-прежнему переключаться между внутренней консолью и последовательной консолью, установив соответствующие переменные окружения в загрузчике. Подробнее см. в loader(8).

Эта строка в /boot/loader.conf или /boot/loader.conf.local настраивает загрузчик и ядро для отправки загрузочных сообщений на последовательную консоль, независимо от параметров в /boot.config:

```
console="comconsole"
```

Эта строка должна быть первой в /boot/loader.conf, чтобы сообщения загрузки выводились на последовательную консоль как можно раньше.

Если эта строка отсутствует или имеет значение `console="vidconsole"`, загрузчик и ядро будут использовать консоль, указанную параметром `-h` в загрузочном блоке. Подробнее см. в loader.conf(5).

На данный момент загрузчик не имеет опции, аналогичной `-P` в загрузочном блоке, и не предусмотрена возможность автоматического выбора внутренней консоли и последовательной консоли в зависимости от наличия клавиатуры.

Хотя это и не обязательно, можно настроить вывод приглашения `login` через последовательный порт. Для этого отредактируйте запись для последовательного порта в /etc/ttys, следуя инструкциям в Настройка терминала. Если скорость последовательного порта была изменена, замените `std.115200` на новое значение.

30.6.3. Установка более высокой скорости последовательного порта

По умолчанию настройки последовательного порта: скорость 115200 бод, 8 бит данных, без контроля чётности и 1 стоповый бит. Чтобы изменить стандартную скорость консоли, используйте один из следующих вариантов:

- Отредактируйте файл /etc/make.conf и установите значение `BOOT_COMCONSOLE_SPEED` на новую скорость консоли. Затем перекомпилируйте и установите загрузочные блоки и загрузчик:

  ```
  # cd /sys/boot
  # make clean
  # make
  # make install
  ```

  Если последовательная консоль настроена иным способом, кроме загрузки с `-h`, или если последовательная консоль, используемая ядром, отличается от той, что используется загрузочными блоками, добавьте следующую опцию с нужной скоростью в файл конфигурации собственного ядра и соберите новое ядро:

  ```
  options CONSPEED=19200
  ```

- Добавьте параметр загрузки `-S19200` в /boot.config, заменив `19200` на нужную скорость.
- Добавьте следующие параметры в /boot/loader.conf. Замените `115200` на нужную скорость.

```
boot_multicons="YES"
boot_serial="YES"
comconsole_speed="115200"
console="comconsole,vidconsole"
```

30.6.4. Вход в отладчик DDB с последовательной линии

Для настройки возможности перехода в отладчик ядра с последовательной консоли добавьте следующие параметры в пользовательский конфигурационный файл ядра и соберите ядро, используя инструкции из Настройка ядра FreeBSD. Обратите внимание, что хотя это полезно для удалённой диагностики, это также опасно, если на последовательном порту генерируется ложный BREAK. Дополнительную информацию об отладчике ядра можно найти в ddb(4) и ddb(8).

```
options BREAK_TO_DEBUGGER
options DDB
```

Глава 31. PPP

31.1. Обзор

FreeBSD поддерживает протокол Point-to-Point (PPP), который может использоваться для установки сетевого или интернет-соединения с помощью модема. В этой главе описывается, как настроить услуги связи на основе модема в FreeBSD.

Прочтите эту главу, чтобы узнать:

- Как настроить, использовать и устранять неполадки PPP-подключения.
- Как настроить PPP over Ethernet (PPPoE).
- Как настроить PPP over ATM (PPPoA).

Прежде чем читать эту главу, необходимо:

- Знать основные термины и понятия сетевых технологий.
- Понимать основы и назначение коммутируемого соединения и PPP.

31.2. Настройка PPP

FreeBSD предоставляет встроенную поддержку для управления коммутируемыми PPP-соединениями с помощью ppp(8). Стандартное ядро FreeBSD поддерживает tun, который используется для взаимодействия с модемным оборудованием. Настройка выполняется путём редактирования как минимум одного конфигурационного файла, и предоставляются примеры таких файлов. Наконец, `ppp` используется для запуска и управления соединениями.

Для использования PPP-соединения необходимы следующие компоненты:

- Учётная запись для коммутируемого доступа у поставщика интернет-услуг (Internet Service Provider — ISP).
- Модем для коммутируемого доступа.
- Номер телефона для дозвона к ISP.
- Имя пользователя и пароль, назначенные ISP.
- IP-адрес одного или нескольких DNS-серверов. Обычно эти адреса предоставляет интернет-провайдер. Если он этого не сделал, FreeBSD можно настроить на использование согласования DNS.

Если отсутствует какая-либо необходимая информация, обратитесь к интернет-провайдеру.

Следующая информация может быть предоставлена интернет-провайдером, но не является обязательной:

- IP-адрес шлюза по умолчанию. Если эта информация неизвестна, интернет-провайдер автоматически предоставит правильное значение во время настройки соединения. При

настройке PPP в FreeBSD этот адрес обозначается как HISADDR.

- Маска подсети. Если провайдер не предоставил её, в конфигурационном файле ppp(8) будет использовано значение 255.255.255.255. *

  Если интернет-провайдер выделил статический IP-адрес и имя хоста, их следует указать в файле конфигурации. В противном случае эта информация будет автоматически предоставлена при настройке соединения.

Оставшаяся часть этого раздела демонстрирует, как настроить FreeBSD для распространённых сценариев PPP-подключения. Необходимый конфигурационный файл — это /etc/ppp/ppp.conf, а дополнительные файлы и примеры доступны в /usr/share/examples/ppp/.

На протяжении всего этого раздела во многих примерах файлов указаны номера строк. Эти номера добавлены для удобства обсуждения и не предназначены для включения в фактический файл.

При редактировании конфигурационного файла важно соблюдать правильные отступы. Строки, заканчивающиеся на :, начинаются с первой колонки (в начале строки), тогда как все остальные строки должны иметь отступ, как показано, с использованием пробелов или табуляции.

31.2.1. Базовая конфигурация

Для настройки PPP-соединения сначала отредактируйте файл /etc/ppp/ppp.conf, указав данные для подключения к интернет-провайдеру. Этот файл описывается следующим образом:

```
1     default:
2       set log Phase Chat LCP IPCP CCP tun command
3       ident user-ppp VERSION
4       set device /dev/cuau0
5       set speed 115200
6       set dial "ABORT BUSY ABORT NO\\sCARRIER TIMEOUT 5 \
7                 \"\" AT OK-AT-OK ATE1Q0 OK \\dATDT\\T TIMEOUT 40 CONNECT"
8       set timeout 180
9       enable dns
10
11    provider:
12      set phone "(123) 456 7890"
13      set authname foo
14      set authkey bar
15      set timeout 300
16      set ifaddr x.x.x.x/0 y.y.y.y/0 255.255.255.255 0.0.0.0
17      add default HISADDR
```

Строка 1

Определяет запись default. Команды в этой записи (строки со 2 по 9) выполняются

автоматически при запуске `ppp`.

Строка 2

Включает подробные параметры журналирования для тестирования соединения. Как только конфигурация начнёт работать удовлетворительно, эту строку следует сократить до:

```
set log phase tun
```

Строка 3

Отображает версию ppp(8) для PPP-программного обеспечения, работающего на другой стороне соединения.

Строка 4

Определяет устройство, к которому подключен модем, где COM1 соответствует /dev/cuau0, а COM2 соответствует /dev/cuau1.

Строка 5

Устанавливает скорость соединения. Если `115200` не работает на старом модеме, попробуйте `38400`.

Строки 6 и 7

Строка набора, записанная в синтаксисе expect-send. Дополнительную информацию смотрите в chat(8).

Обратите внимание, что эта команда продолжается на следующей строке для удобства чтения. Любая команда в ppp.conf может быть записана таким образом, если последним символом в строке является `\`.

Строка 8

Устанавливает время ожидания бездействия соединения в секундах.

Строка 9

Указывает узлу подтвердить настройки DNS. Если в локальной сети работает собственный DNS-сервер, эту строку следует закомментировать, добавив `#` в начале строки, или удалить.

Строка 10

Пустая строка для удобочитаемости. Пустые строки игнорируются ppp(8).

Строка 11

Определяет запись с именем `provider`. Это имя можно изменить на название ISP, чтобы использовать команду `load ISP` для установки соединения.

Строка 12

Используйте номер телефона вашего ISP. Несколько номеров можно указать, используя в качестве разделителя символ двоеточия (`:`) или вертикальной черты (`|`). Для перебора

номеров используйте двоеточие. Чтобы всегда сначала пытаться дозвониться по первому номеру и пробовать другие номера только в случае неудачи с первым, используйте символ вертикальной черты. Всегда заключайте весь набор номеров в кавычки ("), чтобы избежать сбоев при наборе.

Строки 13 и 14

Используйте имя пользователя и пароль для ISP.

Строка 15

Устанавливает таймаут неактивности по умолчанию для соединения в секундах. В данном примере соединение будет автоматически закрыто после 300 секунд бездействия. Чтобы отключить таймаут, установите значение в ноль.

Строка 16

Устанавливает адреса интерфейсов. Используемые значения зависят от того, получен ли статический IP-адрес от провайдера или он динамически назначается при подключении.

Если интернет-провайдер выделил статический IP-адрес и шлюз по умолчанию, замените *x.x.x.x* на статический IP-адрес, а *y.y.y.y* — на IP-адрес шлюза по умолчанию. Если провайдер предоставил только статический IP-адрес без адреса шлюза, замените *y.y.y.y* на `10.0.0.2/0`.

Если IP-адрес изменяется при каждом подключении, измените эту строку на следующее значение. Это указывает ppp(8) использовать протокол IP Configuration Protocol (IPCP) для согласования динамического IP-адреса:

```
set ifaddr 10.0.0.1/0 10.0.0.2/0 255.255.255.255 0.0.0.0
```

Line 17

Оставьте эту строку как есть, так как она добавляет маршрут по умолчанию к шлюзу. `HISADDR` будет автоматически заменён на адрес шлюза, указанный в строке 16. Важно, чтобы эта строка располагалась после строки 16.

В зависимости от того, запускается ли ppp(8) вручную или автоматически, может потребоваться создать файл /etc/ppp/ppp.linkup со следующим содержимым. Этот файл необходим при запуске `ppp` в режиме `-auto`. Он используется после установки соединения. На этом этапе IP-адрес уже будет назначен, и можно добавить записи в таблицу маршрутизации. При создании этого файла убедитесь, что значение *provider* совпадает с указанным в строке 11 файла ppp.conf.

```
provider:
      add default HISADDR
```

Этот файл также необходим, когда адрес шлюза по умолчанию "угадывается" в конфигурации статического IP-адреса. В этом случае удалите строку 17 из ppp.conf и создайте файл /etc/ppp/ppp.linkup с указанными двумя строками. Дополнительные примеры для этого файла можно найти в /usr/share/examples/ppp/.

По умолчанию `ppp` должен запускаться от имени `root`. Чтобы изменить это поведение, добавьте учётную запись пользователя, который должен запускать `ppp`, в группу `network` в файле /etc/group.

Затем предоставьте пользователю доступ к одной или нескольким записям в /etc/ppp/ppp.conf с помощью `allow`. Например, чтобы дать `fred` и `mary` разрешение только на запись `provider:`, добавьте следующую строку в раздел `provider:`:

```
allow users fred mary
```

Чтобы предоставить указанным пользователям доступ ко всем записям, поместите эту строку в раздел `default` вместо этого.

31.2.2. Расширенная настройка

Возможно настроить PPP для предоставления адресов DNS- и NetBIOS-серверов по запросу.

Для включения этих расширений в PPP версии 1.x можно добавить следующие строки в соответствующую секцию файла /etc/ppp/ppp.conf.

```
enable msext
set ns 203.14.100.1 203.14.100.2
set nbns 203.14.100.5
```

А для PPP версии 2 и выше:

```
accept dns
set dns 203.14.100.1 203.14.100.2
set nbns 203.14.100.5
```

Это сообщит клиентам адреса основного и дополнительного серверов имен, а также хост сервера имён NetBIOS.

В версии 2 и выше, если строка `set dns` опущена, PPP будет использовать значения, указанные в /etc/resolv.conf.

Проверка подлинности (аутентификация) PAP и CHAP

Некоторые интернет-провайдеры настраивают свои системы таким образом, что проверка подлинности пользователя при подключении выполняется с использованием механизмов PAP или CHAP. В таком случае провайдер не будет выводить приглашение `login:` при соединении, а сразу начнёт работу по протоколу PPP.

PAP менее безопасен, чем CHAP, но безопасность обычно не является проблемой, так как пароли, хотя и передаются в открытом виде с PAP, передаются только по последовательной линии. У взломщиков мало возможностей для "прослушивания".

Необходимо внести следующие изменения:

```
13      set authname MyUserName
14      set authkey MyPassword
15      set login
```

Line 13

Эта строка указывает имя пользователя PAP/CHAP. Вставьте правильное значение для *MyUserName*.

Line 14

Эта строка указывает пароль PAP/CHAP. Вставьте правильное значение вместо *MyPassword*. Дополнительная строка может быть добавлена, например:

```
16      accept PAP
```

или

```
16      accept CHAP
```

чтобы сделать это намерение очевидным, но PAP и CHAP по умолчанию принимаются оба.

Строка 15

Поставщик интернет-услуг обычно не требует входа на сервер при использовании PAP или CHAP. Поэтому отключите строку `set login`.

Использование возможности трансляции сетевых адресов (NAT) в PPP

PPP обладает возможностью использовать внутренний NAT без функций перенаправления ядра. Эту функциональность можно включить следующей строкой в /etc/ppp/ppp.conf:

```
nat enable yes
```

Или NAT может быть включён с помощью параметра командной строки `-nat`. Также существует параметр /etc/rc.conf с именем `ppp_nat`, который включён по умолчанию.

При использовании этой функции может быть полезно включить следующие параметры в файле /etc/ppp/ppp.conf для переадресации входящих соединений:

```
nat port tcp 10.0.0.2:ftp ftp
nat port tcp 10.0.0.2:http http
```

или вообще не доверять внешнему миру

```
nat deny_incoming yes
```

31.2.3. Окончательная конфигурация системы

В то время как `ppp` уже настроен, некоторые изменения всё ещё необходимо внести в /etc/rc.conf.

Посмотрев файл с начала до конца, убедитесь, что строка `hostname=` установлена:

```
hostname="foo.example.com"
```

Если интернет-провайдер предоставил статический IP-адрес и имя, используйте это имя в качестве имени хоста.

Проверьте переменную `network_interfaces`. Чтобы настроить систему для дозвона к интернет-провайдеру по требованию, убедитесь, что устройство tun0 добавлено в список, в противном случае удалите его.

```
network_interfaces="lo0 tun0"
ifconfig_tun0=
```

Переменная `ifconfig_tun0` должна быть пустой, а файл с именем /etc/start_if.tun0 должен быть создан. Этот файл должен содержать строку:

```
ppp -auto mysystem
```

Этот скрипт выполняется во время настройки сети, запуская демон ppp в автоматическом режиме. Если данный компьютер выступает в качестве шлюза, рекомендуется включить опцию `-alias`. Дополнительные сведения см. на соответствующей странице Справочника.

Убедитесь, что программа маршрутизатора отключена, проверив, что в /etc/rc.conf следующая строка установлена в `NO`:

```
router_enable="NO"
```

Важно, чтобы демон `routed` не запускался, так как `routed` имеет тенденцию удалять записи по умолчанию из таблицы маршрутизации , созданные `ppp`.

Вероятно, стоит убедиться, что строка `sendmail_flags` не содержит опцию `-q`, иначе `sendmail` будет периодически пытаться выполнить сетевой запрос, что может привести к нежелательному дозвону. Можно попробовать:

```
sendmail_flags="-bd"
```

Недостаток в том, что `sendmail` вынужден повторно проверять очередь почты каждый раз при установке ppp-соединения. Для автоматизации этого процесса добавьте `!bg` в файл ppp.linkup:

```
1     provider:
2       delete ALL
3       add 0 0 HISADDR
4       !bg sendmail -bd -q30m
```

Альтернативный вариант — настроить "dfilter" для блокировки SMTP-трафика. Дополнительные сведения см. в примерных файлах.

31.2.4. Использование `ppp`

Осталось только перезагрузить машину. После перезагрузки введите:

```
# ppp
```

затем `dial provider` для запуска сеанса PPP или, чтобы настроить `ppp` для автоматического установления сеансов при наличии исходящего трафика и отсутствии файла start_if.tun0, введите:

```
# ppp -auto provider
```

Возможно общаться с программой `ppp`, пока она работает в фоновом режиме, но только если настроен соответствующий диагностический порт. Для этого добавьте следующую строку в конфигурацию:

```
set server /var/run/ppp-tun%d DiagnosticPassword 0177
```

Это заставит PPP прослушивать указанный сокет домена UNIX®, запрашивая у клиентов указанный пароль перед предоставлением доступа. Символ `%d` в имени заменяется на номер устройства tun, который используется.

После настройки сокета программа pppctl(8) может использоваться в скриптах для управления работающим приложением.

31.2.5. Настройка сервисов для входящих звонков

Входящие соединения по модему предоставляет хорошее описание настройки услуг дозвона с использованием getty(8).

Альтернативой `getty` является порт comms/mgetty+sendfax, более интеллектуальная версия `getty`, разработанная для работы с коммутируемыми линиями.

Преимущество использования `mgetty` заключается в том, что он активно *общается* с модемами, то есть если порт выключен в /etc/ttys, то модем не будет отвечать на звонки.

Поздние версии `mgetty` (начиная с 0.99beta) также поддерживают автоматическое определение PPP-потоков, что позволяет клиентам получать доступ к серверу без использования скриптов.

См. http://mgetty.greenie.net/doc/mgetty_toc.html для получения дополнительной информации о `mgetty`.

По умолчанию порт comms/mgetty+sendfax поставляется с включённой опцией `AUTO_PPP`, которая позволяет `mgetty` обнаруживать фазу LCP PPP-соединений и автоматически запускать оболочку ppp. Однако, поскольку стандартная последовательность ввода логина и пароля не выполняется, необходимо аутентифицировать пользователей с помощью PAP или CHAP.

Этот раздел предполагает, что пользователь успешно скомпилировал и установил порт comms/mgetty+sendfax в своей системе.

Убедитесь, что в файле /usr/local/etc/mgetty+sendfax/login.config присутствует следующее:

```
/AutoPPP/ -     - /etc/ppp/ppp-pap-dialup
```

Это указывает `mgetty` запускать ppp-pap-dialup для обнаруженных PPP-соединений.

Создайте исполняемый файл с именем /etc/ppp/ppp-pap-dialup, содержащий следующее:

```
#!/bin/sh
exec /usr/sbin/ppp -direct pap$IDENT
```

Для каждой включённой коммутируемой линии в /etc/ttys создайте соответствующую запись в /etc/ppp/ppp.conf. Это будет прекрасно сосуществовать с определениями, созданными ранее.

```
pap:
  enable pap
  set ifaddr 203.14.100.1 203.14.100.20-203.14.100.40
  enable proxy
```

Каждый пользователь, входящий в систему этим методом, должен иметь имя пользователя и пароль в /etc/ppp/ppp.secret. Или можно добавить следующую опцию для аутентификации пользователей через PAP из /etc/passwd.

```
enable passwdauth
```

Для назначения статического IP-адреса некоторым пользователям укажите этот адрес в качестве третьего аргумента в /etc/ppp/ppp.secret. Примеры можно найти в /usr/share/examples/ppp/ppp.secret.sample.

31.3. Устранение неполадок PPP-подключений

В этом разделе рассматриваются некоторые проблемы, которые могут возникнуть при использовании PPP через модемное соединение. Некоторые интернет-провайдеры выводят приглашение `ssword`, тогда как другие используют `password`. Если скрипт `ppp` не настроен соответствующим образом, попытка входа завершится неудачей. Наиболее распространённый способ отладки соединений `ppp` — это ручное подключение, как описано в данном разделе.

31.3.1. Проверьте файлы устройств

При использовании собственного ядра убедитесь, что в конфигурационном файле ядра присутствует следующая строка:

```
device   uart
```

Устройство uart уже включено в ядро `GENERIC`, поэтому в этом случае дополнительные действия не требуются. Просто проверьте вывод `dmesg` на наличие модема с помощью:

```
# dmesg | grep uart
```

Это должно вывести некоторую важную информацию об устройствах uart. Это COM-порты, которые нам нужны. Если модем работает как стандартный последовательный порт, он должен быть указан как uart1 или COM2. Если это так, пересборка ядра не требуется. При сопоставлении, если модем находится на uart1, устройство модема будет /dev/cuau1.

31.3.2. Подключение вручную

Подключение к Интернету с ручным управлением `ppp` выполняется быстро, просто и является отличным способом для отладки соединения или получения информации о том, как ISP взаимодействует с клиентами `ppp`. Давайте запустим PPP из командной строки. Обратите внимание, что во всех наших примерах мы будем использовать *example* в качестве имени хоста машины, на которой работает PPP. Для запуска `ppp`:

```
# ppp
```

```
ppp ON example> set device /dev/cuau1
```

Эта вторая команда устанавливает модемное устройство в cuau1.

```
ppp ON example> set speed 115200
```

Это устанавливает скорость соединения на 115 200 Кбит/с.

```
ppp ON example> enable dns
```

Это указывает `ppp` настроить резолвер и добавить строки nameserver в /etc/resolv.conf. Если `ppp` не может определить имя хоста, его можно задать вручную позже.

```
ppp ON example> term
```

Это переключает в "терминальный" режим для ручного управления модемом.

```
deflink: Entering terminal mode on /dev/cuau1
type '~h' for help
```

```
at
OK
atdt123456789
```

Используйте `at` для инициализации модема, затем используйте `atdt` и номер провайдера для начала процесса дозвона.

```
CONNECT
```

Подтверждение подключения: если у нас возникнут проблемы с соединением, не связанные с оборудованием, здесь мы попытаемся их устранить.

```
ISP Login:myusername
```

На этом приглашении введите имя пользователя, предоставленное ISP.

```
ISP Pass:mypassword
```

На этом запросе введите пароль, предоставленный интернет-провайдером. Как и при входе в FreeBSD, пароль не будет отображаться.

```
Shell or PPP:ppp
```

В зависимости от провайдера, это приглашение может не появляться. Если оно появилось, то запрашивается выбор между использованием оболочки провайдера или запуском ppp. В данном примере был выбран ppp для установки подключения к Интернету.

```
Ppp ON example>
```

Обратите внимание, что в этом примере первая буква p написана заглавной. Это означает, что мы успешно подключились к интернет-провайдеру.

```
Ppp ON example>
```

Мы успешно прошли аутентификацию у нашего провайдера и ожидаем назначенный IP-адрес.

```
PPP ON example>
```

Мы согласовали IP-адрес и успешно установили соединение.

```
PPP ON example>add default HISADDR
```

Вот мы добавляем наш маршрут по умолчанию, это необходимо сделать до того, как мы сможем взаимодействовать с внешним миром, так как на данный момент единственное установленное соединение — с удалённой стороной. Если это не удаётся из-за существующих маршрутов, поставьте восклицательный знак ! перед командой add. По-другому, можно установить маршрут до установки соединения, и тогда будет согласован новый маршрут соответствующим образом.

Если всё прошло хорошо, у нас теперь должно быть активное подключение к Интернету, которое можно перевести в фоновый режим с помощью CTRL + z. Если PPP возвращается к ppp, соединение было потеряно. Это полезно знать, так как показывает статус подключения. Заглавные буквы P обозначают подключение к ISP, а строчные p показывают, что соединение потеряно.

31.3.3. Отладка

Если соединение не может быть установлено, отключите аппаратный поток CTS/RTS с помощью `set ctsrts off`. Это в основном происходит при подключении к некоторым терминальным серверам с поддержкой PPP, где PPP зависает при попытке записи данных в линию связи и ожидает сигнала Clear To Send (CTS), который может никогда не поступить. При использовании этой опции включите `set accmap`, так как может потребоваться обойти аппаратную зависимость от передачи определённых символов от конца к концу, чаще всего XON/XOFF. Дополнительную информацию об этой опции и её использовании см. в ppp(8).

Старый модем может потребовать `set parity even`. По умолчанию чётность отключена, но она используется для проверки ошибок с значительным увеличением трафика на старых

модемах.

PPP может не вернуться в командный режим, что обычно является ошибкой согласования, когда ISP ожидает начала процесса согласования. В этом случае использование `~p` заставит ppp начать отправку информации о конфигурации.

Если приглашение на вход никогда не появляется, скорее всего, требуется аутентификация PAP или CHAP. Чтобы использовать PAP или CHAP, добавьте следующие параметры в PPP перед переходом в терминальный режим:

```
ppp ON example> set authname myusername
```

Где *myusername* следует заменить на имя пользователя, предоставленное провайдером интернет-услуг.

```
ppp ON example> set authkey mypassword
```

Где *mypassword* следует заменить на пароль, назначенный провайдером.

Если соединение установлено, но не удаётся найти доменное имя, попробуйте выполнить `ping(8)` для IP-адреса. Если наблюдается 100% потерь пакетов, вероятно, маршрут по умолчанию не назначен. Убедитесь, что во время подключения был установлен параметр `add default HISADDR`. Если удаётся установить соединение с удалённым IP-адресом, возможно, в файл /etc/resolv.conf не добавлен адрес резолвера. Этот файл должен выглядеть следующим образом:

```
domain example.com
nameserver x.x.x.x
nameserver y.y.y.y
```

Где *x.x.x.x* и *y.y.y.y* следует заменить на IP-адреса DNS-серверов провайдера.

Для настройки syslog(3) для ведения журнала PPP-подключения убедитесь, что следующая строка присутствует в /etc/syslog.conf:

```
!ppp
*.*     /var/log/ppp.log
```

31.4. Использование PPP через Ethernet (PPPoE)

Этот раздел описывает, как настроить PPP через Ethernet (PPPoE).

Вот пример рабочего ppp.conf:

```
default:
```

```
  set log Phase tun command # add more detailed logging when needed
  set ifaddr 10.0.0.1/0 10.0.0.2/0

name_of_service_provider:
  set device PPPoE:xl1 # replace xl1 with the Ethernet device
  set authname THELOGINNAME
  set authkey THEPASSWORD
  set dial
  set login
  add default HISADDR
```

Как `root`, выполните:

```
# ppp -ddial name_of_service_provider
```

Добавьте следующее в /etc/rc.conf:

```
ppp_enable="YES"
ppp_mode="ddial"
ppp_nat="YES"   # when needing to enable nat for the local network, otherwise NO
ppp_profile="name_of_service_provider"
```

31.4.1. Использование тега сервиса PPPoE

Иногда для установки соединения может потребоваться использовать тег сервиса. Теги сервиса используются для различения разных серверов PPPoE, подключённых к определённой сети.

Любая необходимая информация о теге сервиса должна быть указана в документации, предоставляемой интернет-провайдером (ISP).

В крайнем случае можно попробовать установить пакет или порт net/rr-pppoe. Однако учтите, что это может перепрограммировать ваш модем и сделать его бесполезным, поэтому хорошо подумайте перед этим. Просто установите программу, поставляемую с модемом. Затем откройте меню **System** в программе. Имя профиля должно быть указано там. Обычно это *ISP*.

Имя профиля (тег сервиса) будет использоваться в записи конфигурации PPPoE в файле ppp.conf как часть провайдера для `set device`. Подробности смотрите в ppp(8). Это должно выглядеть так:

```
set device PPPoE:xl1:ISP
```

Не забудьте изменить *xl1* на соответствующее имя устройства вашей Ethernet-карты.

Не забудьте изменить *ISP* на профиль, определённый вами ранее.

Для получения дополнительной информации обратитесь к более дешёвому широкополосному доступу с FreeBSD через DSL от Renaud Waldura.

31.4.2. PPPoE с модемом 3Com® HomeConnect™ ADSL Modem Dual Link

Этот модем не соответствует спецификации PPPoE, определённой в RFC 2516.

Чтобы FreeBSD мог взаимодействовать с этим устройством, необходимо установить sysctl. Это можно сделать автоматически при загрузке, изменив файл /etc/sysctl.conf:

```
net.graph.nonstandard_pppoe=1
```

или может быть выполнено сразу с помощью команды:

```
# sysctl net.graph.nonstandard_pppoe=1
```

К сожалению, поскольку это общесистемная настройка, невозможно одновременно работать с обычным клиентом или сервером PPPoE и модемом 3Com® HomeConnect™ ADSL.

31.5. Использование PPP через ATM (PPPoA)

Следующее описание объясняет, как настроить PPP через ATM (PPPoA). PPPoA — популярный выбор среди европейских провайдеров DSL.

31.5.1. Использование mpd

Приложение mpd можно использовать для подключения к различным сервисам, в частности к PPTP-сервисам. Его можно установить с помощью пакета net/mpd5 или порта. Многие ADSL-модемы требуют создания PPTP-туннеля между модемом и компьютером.

После установки настройте mpd в соответствии с параметрами провайдера. Порт предоставляет набор примеров конфигурационных файлов, которые хорошо документированы в /usr/local/etc/mpd/. Полное руководство по настройке mpd доступно в формате HTML в /usr/ports/shared/doc/mpd/. Вот пример конфигурации для подключения к ADSL-сервису с помощью mpd. Конфигурация разделена на два файла, первый — mpd.conf:

Этот пример mpd.conf работает только с mpd 4.x.

```
default:
    load adsl

adsl:
    new -i ng0 adsl adsl
    set bundle authname username ①
    set bundle password password ②
    set bundle disable multilink
```

```
    set link no pap acfcomp protocomp
    set link disable chap
    set link accept chap
    set link keep-alive 30 10

    set ipcp no vjcomp
    set ipcp ranges 0.0.0.0/0 0.0.0.0/0

    set iface route default
    set iface disable on-demand
    set iface enable proxy-arp
    set iface idle 0

    open
```

① Имя пользователя для аутентификации у вашего провайдера.

② Пароль, используемый для аутентификации у вашего провайдера.

Информация о соединении или соединениях, которые необходимо установить, находится в файле mpd.links. Пример файла mpd.links, соответствующий приведённому выше примеру, представлен ниже:

```
adsl:
    set link type pptp
    set pptp mode active
    set pptp enable originate outcall
    set pptp self 10.0.0.1 ①
    set pptp peer 10.0.0.138 ②
```

① IP-адрес компьютера FreeBSD, на котором работает mpd.

② IP-адрес модема ADSL. По умолчанию для Alcatel SpeedTouch™ Home используется `10.0.0.138`.

Возможно легко инициализировать соединение, выполнив следующую команду от имени `root`:

```
# mpd -b adsl
```

Для просмотра состояния соединения:

```
% ifconfig ng0
ng0: flags=88d1<UP,POINTOPOINT,RUNNING,NOARP,SIMPLEX,MULTICAST> mtu 1500
     inet 216.136.204.117 --> 204.152.186.171 netmask 0xffffffff
```

Использование `mpd` является рекомендуемым способом подключения к услуге ADSL в

FreeBSD.

31.5.2. Использование pptpclient

Также возможно использовать FreeBSD для подключения к другим сервисам PPPoA с помощью net/pptpclient.

Для подключения к DSL-сервису с помощью net/pptpclient установите порт или пакет, затем отредактируйте файл /etc/ppp/ppp.conf. Пример раздела ppp.conf приведён ниже. Дополнительные сведения о параметрах ppp.conf можно найти в ppp(8).

```
adsl:
 set log phase chat lcp ipcp ccp tun command
 set timeout 0
 enable dns
 set authname username ①
 set authkey password ②
 set ifaddr 0 0
 add default HISADDR
```

① Имя пользователя для провайдера DSL.

② Пароль для вашей учётной записи.

Поскольку пароль учётной записи добавляется в ppp.conf в открытом виде, убедитесь, что никто не может прочитать содержимое этого файла:

```
# chown root:wheel /etc/ppp/ppp.conf
# chmod 600 /etc/ppp/ppp.conf
```

Это откроет туннель для PPP-сессии к DSL-маршрутизатору. Ethernet DSL-модемы имеют предварительно настроенный LAN IP-адрес для подключения. В случае Alcatel SpeedTouch™ Home этот адрес 10.0.0.138. В документации маршрутизатора должен быть указан используемый устройством адрес. Чтобы открыть туннель и начать PPP-сессию:

```
# pptp address adsl
```

Если в конец этой команды добавить амперсанд ("&"), pptp вернёт приглашение командной строки.

Будет создано виртуальное туннельное устройство tun для взаимодействия между процессами pptp и ppp. Как только вернется приглашение или процесс pptp подтвердит соединение, проверьте туннель:

```
% ifconfig tun0
tun0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1500
```

```
        inet 216.136.204.21 --> 204.152.186.171 netmask 0xffffff00
    Opened by PID 918
```

Если соединение не удалось, проверьте настройки маршрутизатора, который обычно доступен через веб-браузер. Также изучите вывод `pptp` и содержимое файла журнала /var/log/ppp.log для поиска подсказок.

Глава 32. Электронная почта

32.1. Обзор

"Электронная почта", более известная как email, является одним из самых распространённых средств связи в наши дни. Эта глава даёт базовое введение в настройку почтового сервера на FreeBSD, а также основы отправки и получения электронной почты с использованием FreeBSD. Для полного ознакомления с темой обратитесь к книгам, перечисленным в разделе Библиография.

Эта глава охватывает:

- Какие программные компоненты участвуют в отправке и получении электронной почты.
- Как настроить DragonFly Mail Agent.
- Где расположены основные файлы конфигурации Sendmail в FreeBSD.
- Разница между удалёнными и локальными почтовыми ящиками.
- Как установить и настроить альтернативный агент передачи почты (MTA), заменяющий DragonFly Mail Agent или Sendmail.
- Как устранить распространённые проблемы с почтовым сервером.
- Как настроить Sendmail только для отправки почты.
- Как настроить аутентификацию SMTP для повышения безопасности в Sendmail.
- Как установить и использовать почтовый клиент, например mutt, для отправки и получения электронной почты.
- Как загружать почту с удалённого POP или IMAP сервера.
- Как автоматически применять фильтры и правила к входящей почте.

32.2. Компоненты почты

В процессе обмена электронной почтой участвуют пять основных компонентов: почтовый пользовательский агент (MUA), агент пересылки почты (MTA), почтовый хост, удалённый или локальный почтовый ящик и DNS. В этом разделе представлен обзор этих компонентов.

Почтовый клиент (MUA)

Почтовый клиент (Mail User Agent — MUA) — это приложение, используемое для создания, отправки и получения электронных писем. Это может быть программа командной строки, например встроенная утилита `mail` или стороннее приложение из коллекции портов, такое как alpine, elm или mutt. В коллекции портов также доступны десятки графических программ, включая Claws Mail, Evolution и Thunderbird. Некоторые организации предоставляют веб-почту, доступную через браузер. Дополнительную информацию об установке и использовании MUA в FreeBSD можно найти в разделе Почтовые клиенты.

Почтовый агент пересылки (MTA)

Агент пересылки почты (MTA) отвечает за получение входящей почты и отправку исходящей. Начиная с версии FreeBSD 14.0, MTA по умолчанию — DragonFly Mail Agent (dma(8)); в более ранних версиях используется sendmail(8). Другие MTA, такие как Exim, Postfix и qmail, могут быть установлены для замены MTA по умолчанию.

Почтовый сервер и почтовые ящики

Почтовый сервер — это сервер, отвечающий за доставку и получение почты для хоста или сети. Почтовый сервер собирает всю почту, отправленную в домен, и сохраняет её либо в формате `mbox` по умолчанию, либо в альтернативном формате Maildir, в зависимости от конфигурации. После сохранения почты её можно прочитать локально с помощью почтового клиента или удалённо получить с использованием протоколов, таких как POP или IMAP. Если почта читается локально, устанавливать сервер POP или IMAP не требуется.

Система доменных имён (DNS)

Система доменных имён (DNS) и её демон named(8) играют важную роль в доставке почты. Чтобы доставить почту от одного узла к другому, MTA ищет удалённый узел в DNS, чтобы определить, какой хост будет принимать почту для адресата. Этот процесс также происходит, когда почта отправляется с удалённого хоста на MTA.

32.3. Почтовый агент DragonFly (DMA)

Почтовый агент DragonFly (DMA — DragonFly Mail Agent) — это почтовый агент, используемый в FreeBSD по умолчанию начиная с версии 14.0. dma(8) представляет собой небольшой почтовый транспортный агент (MTA), предназначенный для домашнего и офисного использования. Он принимает почту от локально установленных почтовых пользовательских агентов и доставляет её либо локально, либо удалённо. Удалённая доставка включает в себя такие функции, как поддержка TLS/SSL и аутентификация SMTP.

dma(8) не предназначен в качестве замены полноценным, крупным MTA, таким как sendmail(8) или postfix(1). Следовательно, dma(8) не прослушивает порт 25 для входящих соединений.

32.3.1. Настройка почтового агента DragonFly

DMA поставляется с конфигурацией по умолчанию, которая подходит для многих вариантов установки. Пользовательские настройки определяются в /etc/dma/dma.conf, а аутентификация SMTP настраивается в /etc/dma/auth.conf.

Использование DMA для маршрутизации исходящей почты через Gmail (пример STARTTLS:SMTP)

Этот пример /etc/dma/dma.conf можно использовать для отправки почты через SMTP-серверы Google.

```
SMARTHOST smtp.gmail.com
PORT 587
```

```
AUTHPATH /etc/dma/auth.conf
SECURETRANSFER
STARTTLS
MASQUERADE username@gmail.com
```

Аутентификацию можно настроить одной строкой в /etc/dma/auth.conf:

```
username@gmail.com|smtp.gmail.com:password
```

Если у вас включена двухфакторная аутентификация, потребуется сгенерировать пароль для приложения, так как обычный пароль для входа будет отклонён. Дополнительную информацию о паролях для приложений смотрите в документации Google.

Выполните следующую команду для проверки конфигурации:

```
% echo this is a test | mail -v -s testing-email username@gmail.com
```

Использование DMA для маршрутизации исходящей почты через Fastmail (пример SSL/TLS)

Этот пример /etc/dma/dma.conf можно использовать для отправки почты через SMTP-серверы Fastmail.

```
SMARTHOST smtp.fastmail.com
PORT 465
AUTHPATH /etc/dma/auth.conf
SECURETRANSFER
MAILNAME example.server.com
```

Аутентификацию можно настроить одной строкой в /etc/dma/auth.conf:

```
username@fastmail.com|smtp.fastmail.com:password
```

Выполните следующую команду для проверки конфигурации:

```
% echo this is a test | mail -v -s testing-email username@fastmail.com
```

Использование DMA для маршрутизации исходящей почты через пользовательский почтовый хост

Этот пример /etc/dma/dma.conf можно использовать для отправки почты через пользовательский почтовый сервер.

```
SMARTHOST mail.example.org
PORT 587
AUTHPATH /etc/dma/auth.conf
SECURETRANSFER
STARTTLS
```

Аутентификацию можно настроить одной строкой в /etc/dma/auth.conf:

```
username@example.org|mail.example.org:password
```

Выполните следующую команду для проверки конфигурации:

```
% echo this is a test | mail -v -s testing-email username@example.org
```

32.4. Sendmail

Sendmail - это старейший и многофункциональный агент пересылки почты (MTA) с долгой историей в UNIX® и UNIX-подобных системах. Он входил в базовую систему FreeBSD вплоть до версии 13, предоставляя надёжные возможности транспортировки почты, широкие варианты настройки и поддержку сложной маршрутизации и фильтрации.

32.4.1. Файлы конфигурации

Конфигурационные файлы Sendmail находятся в /etc/mail/.

/etc/mail/access

Этот файл базы данных доступа определяет, какие хосты или IP-адреса имеют доступ к локальному почтовому серверу и какой тип доступа им предоставлен. Хосты, указанные как `OK` (что является опцией по умолчанию), могут отправлять почту на этот хост при условии, что конечный пункт назначения почты — локальная машина. Хосты, указанные как `REJECT`, отклоняются для всех почтовых соединений. Хосты, указанные как `RELAY`, могут отправлять почту для любого адресата через этот почтовый сервер. Хосты, указанные как `ERROR`, получат возврат своей почты с указанной почтовой ошибкой. Если хост указан как `SKIP`, Sendmail прервет текущий поиск для этой записи без принятия или отклонения почты. Сообщения хостов, указанных как `QUARANTINE`, будут задержаны и хосты получат указанный текст в качестве причины задержки.

Примеры использования этих параметров для адресов IPv4 и IPv6 можно найти в образце конфигурации FreeBSD, /etc/mail/access.sample:

Для настройки базы данных доступа используйте формат, приведённый в примере, чтобы внести записи в /etc/mail/access, но не ставьте символ комментария (`#`) перед записями. Создайте запись для каждого хоста или сети, для которых нужно настроить доступ. Отправители почты, соответствующие левой части таблицы, будут обработаны согласно действию, указанному в правой части таблицы.

При каждом обновлении этого файла обновите его базу данных и перезапустите Sendmail:

```
# makemap hash /etc/mail/access < /etc/mail/access
# service sendmail restart
```

/etc/mail/aliases

Этот файл базы данных содержит список виртуальных почтовых ящиков, которые преобразуются в пользователей, файлы, программы или другие псевдонимы. Вот несколько записей, иллюстрирующих формат файла:

```
root: localuser
ftp-bugs: joe,eric,paul
bit.bucket:  /dev/null
procmail: "|/usr/local/bin/procmail"
```

Имя почтового ящика слева от двоеточия раскрывается в целевые адреса справа. Первая запись раскрывает почтовый ящик `root` в почтовый ящик `localuser`, который затем ищется в базе данных /etc/mail/aliases. Если совпадение не найдено, сообщение доставляется в `localuser`. Вторая запись показывает почтовый список. Письма для `ftp-bugs` раскрываются в три локальных почтовых ящика: `joe`, `eric` и `paul`. Удалённый почтовый ящик может быть указан как *user@example.com*. Третья запись демонстрирует, как записывать почту в файл, в данном случае /dev/null. Последняя запись показывает, как отправить почту в программу /usr/local/bin/procmail через UNIX®-канал. Дополнительную информацию о формате этого файла можно найти в aliases(5).

Всякий раз, когда этот файл обновляется, выполните `newaliases` для обновления и инициализации базы данных псевдонимов.

/etc/mail/sendmail.cf

Это основной конфигурационный файл для Sendmail. Он определяет общее поведение Sendmail, включая все — от перезаписи email-адресов до вывода сообщений об отказе для удалённых почтовых серверов. Соответственно, этот конфигурационный файл довольно сложен. К счастью, для стандартных почтовых серверов его редко требуется изменять.

Основной конфигурационный файл Sendmail может быть создан из макросов m4(1), определяющих функции и поведение Sendmail. Подробности можно найти в /usr/src/contrib/sendmail/cf/README.

Всякий раз, когда в этот файл вносятся изменения, Sendmail необходимо перезапустить, чтобы изменения вступили в силу.

/etc/mail/virtusertable

Этот файл базы данных сопоставляет почтовые адреса виртуальных доменов и пользователей с реальными почтовыми ящиками. Эти почтовые ящики могут быть локальными, удалёнными, алиасами, определёнными в /etc/mail/aliases, или файлами. Это позволяет размещать несколько виртуальных доменов на одной машине.

FreeBSD предоставляет пример файла конфигурации в /etc/mail/virtusertable.sample, который дополнительно демонстрирует его формат. В следующем примере показано, как создавать пользовательские записи с использованием этого формата:

```
root@example.com                root
postmaster@example.com          postmaster@noc.example.net
@example.com                    joe
```

Этот файл обрабатывается по принципу первого совпадения. Когда электронный адрес соответствует адресу слева, он сопоставляется с локальным почтовым ящиком, указанным справа. Формат первой записи в этом примере сопоставляет конкретный электронный адрес с локальным почтовым ящиком, тогда как формат второй записи сопоставляет конкретный электронный адрес с удалённым почтовым ящиком. Наконец, любой электронный адрес из `example.com`, который не совпал ни с одной из предыдущих записей, будет соответствовать последнему сопоставлению и отправлен в локальный почтовый ящик `joe`. При создании пользовательских записей используйте этот формат и добавляйте их в /etc/mail/virtusertable. При каждом редактировании этого файла обновите его базу данных и перезапустите Sendmail:

```
# makemap hash /etc/mail/virtusertable < /etc/mail/virtusertable
# service sendmail restart
```

/etc/mail/relay-domains

В стандартной установке FreeBSD Sendmail настроен так, чтобы отправлять почту только с хоста, на котором он работает. Например, если доступен POP-сервер, пользователи смогут проверять почту с удалённых устройств, но не смогут отправлять письма извне. Обычно через некоторое время после попытки отправки придёт письмо от `MAILER-DAEMON` с сообщением `5.7 Relaying Denied`.

Наиболее простое решение — добавить полное доменное имя (FQDN) провайдера в /etc/mail/relay-domains. Если требуется добавить несколько адресов, укажите их по одному на строку:

```
your.isp.example.com
other.isp.example.net
users-isp.example.org
www.example.org
```

После создания или редактирования этого файла перезапустите Sendmail с помощью команды `service sendmail restart`.

Теперь любая почта, отправленная через систему с любого хоста из этого списка, при условии, что у пользователя есть учётная запись в системе, будет доставлена успешно. Это позволяет пользователям отправлять почту с системы удалённо, не открывая систему для ретрансляции спама из Интернета.

32.5. Изменение агента передачи почты

Начиная с версии FreeBSD 14.0, dma(8) является MTA по умолчанию, а до версии 14.0 MTA по умолчанию — sendmail(8). Однако системный администратор может изменить MTA системы. Широкий выбор альтернативных MTA доступен в категории `mail` коллекции портов FreeBSD.

Если исходящая почтовая служба по умолчанию отключена, важно заменить её альтернативной системой доставки почты. В противном случае системные функции, такие как periodic(8), не смогут отправлять свои результаты по электронной почте. Многие части системы предполагают наличие работоспособного MTA. Если приложения продолжат использовать стандартные исполняемые файлы для отправки почты после их отключения, письма могут попасть в неактивную очередь и никогда не быть доставлены.

32.5.1. Замена Sendmail на другой MTA

Чтобы полностью отключить sendmail(8), выполните следующие команды:

```
# sysrc sendmail_enable="NO"
# sysrc sendmail_submit_enable="NO"
# sysrc sendmail_outbound_enable="NO"
# sysrc sendmail_msp_queue_enable="NO"
```

Чтобы отключить только службу входящей почты sendmail(8), выполните следующую команду:

```
# sysrc sendmail_enable="NO"
```

Затем остановите службу sendmail(8):

```
# service sendmail onestop
```

Для корректной работы некоторых программ может потребоваться дополнительная настройка, так как sendmail(8) настолько распространён, что многие приложения предполагают его наличие и предварительную конфигурацию. Проверьте файл /etc/periodic.conf и убедитесь, что указанные ниже параметры установлены в значение `NO`. Если файл не существует, создайте его и добавьте следующие записи:

```
daily_clean_hoststat_enable="NO"
daily_status_mail_rejects_enable="NO"
daily_status_include_submit_mailq="NO"
daily_submit_queuerun="NO"
```

Следующий шаг — установка другого MTA, в данном примере будет использоваться dma(8). Как упоминалось выше, dma(8) является MTA по умолчанию в FreeBSD начиная с версии 14.0. Следовательно, установка из портов требуется только при использовании более ранней версии.

Для установки выполните следующую команду:

```
# pkg install dma
```

Выполните настройку, как указано в разделе Настройка почтового агента DragonFly (DMA).

Затем измените все записи в файле /etc/mail/mailer.conf на dma(8):

```
# Execute the "real" sendmail program, named /usr/libexec/sendmail/sendmail
#
# If dma(8) is installed, an example mailer.conf that uses dma(8) instead can
# be found in /usr/share/examples/dma
#
sendmail        /usr/local/libexec/dma
mailq           /usr/local/libexec/dma
newaliases      /usr/local/libexec/dma
```

При использовании версии dma(8), включённой в базовую систему, пути изменятся на /usr/libexec/dma.

Чтобы обеспечить сброс всего, что находится в очереди, при загрузке или перед завершением работы, выполните следующую команду:

```
# sysrc dma_flushq_enable="YES"
```

После завершения настройки рекомендуется перезагрузить систему. Перезагрузка позволяет убедиться, что система корректно настроена для автоматического запуска нового MTA при загрузке.

32.5.2. Замена почтового агента DragonFly (DMA) на другой MTA

Как упоминалось выше, начиная с FreeBSD версии 14.0, MTA по умолчанию — DMA. В этом примере будет использоваться mail/postfix в качестве альтернативного MTA.

Перед установкой пакета mail/postfix требуется дополнительная настройка. Проверьте файл /etc/periodic.conf и убедитесь, что указанные значения установлены в `NO`. Если файл не существует, создайте его со следующими записями:

```
daily_clean_hoststat_enable="NO"
daily_status_mail_rejects_enable="NO"
daily_status_include_submit_mailq="NO"
```

```
daily_submit_queuerun="NO"
```

Затем установите пакет mail/postfix:

```
# pkg install postfix
```

Чтобы запустить пакет mail/postfix при загрузке системы, выполните следующую команду:

```
# sysrc postfix_enable="YES"
```

Хорошей практикой является прочтение сообщения после установки приложения. Оно содержит полезную информацию о настройках и т.д.

Если postfix **ещё не** активирован в /usr/local/etc/mail/mailer.conf, выполните следующие команды:

```
mv /usr/local/etc/mail/mailer.conf /usr/local/etc/mail/mailer.conf.old
install -d /usr/local/etc/mail
install -m 0644 /usr/local/share/postfix/mailer.conf.postfix
/usr/local/etc/mail/mailer.conf
```

При использовании SASL убедитесь, что postfix имеет доступ на чтение файла sasldb. Это достигается добавлением postfix в группу mail и установкой прав на чтение для группы mail для файлов /usr/local/etc/sasldb* (это должно быть настроено по умолчанию при новых установках).

После завершения настройки рекомендуется перезагрузить систему. Перезагрузка позволяет убедиться, что система корректно настроена для автоматического запуска нового MTA при загрузке.

32.6. Почтовые клиенты

MUA — это приложение, используемое для отправки и получения электронной почты. По мере того как электронная почта «развивается» и становится сложнее, MUA становятся всё более мощными и предоставляют пользователям больше функциональности и гибкости. В категории `mail` коллекции портов FreeBSD содержится множество MUA. Среди них есть графические клиенты электронной почты, такие как Evolution или Balsa, а также консольные клиенты, такие как mutt или alpine.

32.6.1. mail

mail(1) — это почтовый клиент (MUA), установленный в FreeBSD по умолчанию. Это консольный MUA, предоставляющий базовые функции для отправки и получения текстовых электронных писем. Он имеет ограниченную поддержку вложений и может работать только с локальными почтовыми ящиками.

Хотя mail(1) изначально не поддерживает взаимодействие с серверами POP или IMAP, эти почтовые ящики могут быть загружены в локальный `mbox` с помощью таких приложений, как fetchmail или getmail.

Для отправки и получения электронной почты запустите mail(1):

```
% mail
```

Содержимое почтового ящика пользователя в /var/mail автоматически читается утилитой mail(1). Если почтовый ящик пуст, программа завершает работу с сообщением о том, что почта не найдена. Если письма есть, запускается интерфейс приложения и отображается список сообщений.

Сообщения автоматически нумеруются, как видно в следующем примере:

```
Mail version 8.1 6/6/93.  Type ? for help.
"/var/mail/username": 3 messages 3 new
>N  1 root@localhost        Mon Mar  8 14:05  14/510   "test"
 N  2 root@localhost        Mon Mar  8 14:05  14/509   "user account"
 N  3 root@localhost        Mon Mar  8 14:05  14/509   "sample"
```

Сообщения теперь можно читать, набрав `t` и затем номер сообщения.

Этот пример читает первое письмо:

```
& t 1
Message 1:
From root@localhost  Mon Mar  8 14:05:52 2004
X-Original-To: username@localhost
Delivered-To: username@localhost
To: username@localhost
Subject: test
Date: Mon,  8 Mar 2004 14:05:52 +0200 (SAST)
From: root@localhost (Charlie Root)

This is a test message, please reply if you receive it.
```

Как видно в этом примере, сообщение будет отображено с полными заголовками.

Чтобы снова отобразить список сообщений, нажмите `h`.

Если требуется ответить на письмо, нажмите либо `R`, либо `r` mail(1). `R` указывает mail(1) ответить только отправителю письма, а `r` — всем остальным получателям сообщения. Эти команды могут быть дополнены номером письма, на которое нужно ответить. После ввода ответа конец сообщения должен быть обозначен отдельной строкой с `.`.

Пример можно увидеть ниже:

```
& R 1
To: root@localhost
Subject: Re: test

Thank you, I did get your email.
.
EOT
```

Чтобы отправить новое письмо, нажмите `m`, затем введите адрес электронной почты получателя. Несколько получателей можно указать, разделяя каждый адрес разделителем `,`. Затем можно ввести тему сообщения, за которым следует содержимое сообщения. Конец сообщения указывается отдельной строкой с символом `.`.

```
& mail root@localhost
Subject: I mastered mail

Now I can send and receive email using mail ... :)
.
EOT
```

При использовании mail(1) нажмите `?`, чтобы в любой момент отобразить справку. Дополнительную информацию об использовании mail(1) можно найти в mail(1).

mail(1) не был предназначен для работы с вложениями и поэтому обрабатывает их плохо. Более современные почтовые клиенты (MUA) обрабатывают вложения более разумным способом.

32.6.2. Mutt

Mutt - это мощный MUA с множеством функций, включая:

- Возможность группировки цепочек сообщений.
- Поддержку PGP для цифровой подписи и шифрования электронной почты.
- Поддержку MIME.
- Поддержку Maildir.
- Большую гибкость в настройке.

Обратитесь к ссылке http://www.mutt.org для получения дополнительной информации о Mutt.

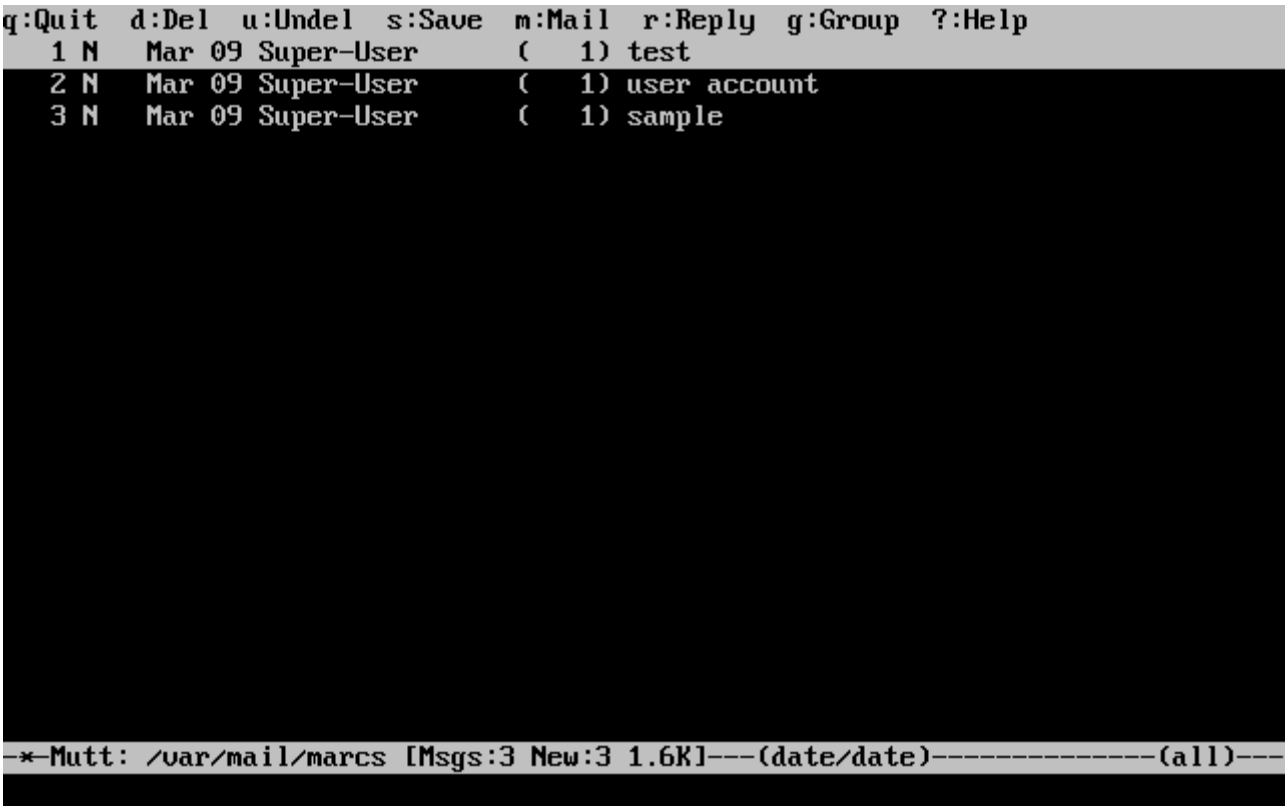
Стоит упомянуть форк Mutt под названием NeoMutt, который добавляет новые возможности. Подробнее можно узнать по ссылке сайт NeoMutt. Если выбран NeoMutt, замените следующие примеры команд с `mutt` на `neomutt`.

Mutt может быть установлен с использованием порта mail/mutt. После установки порта Mutt

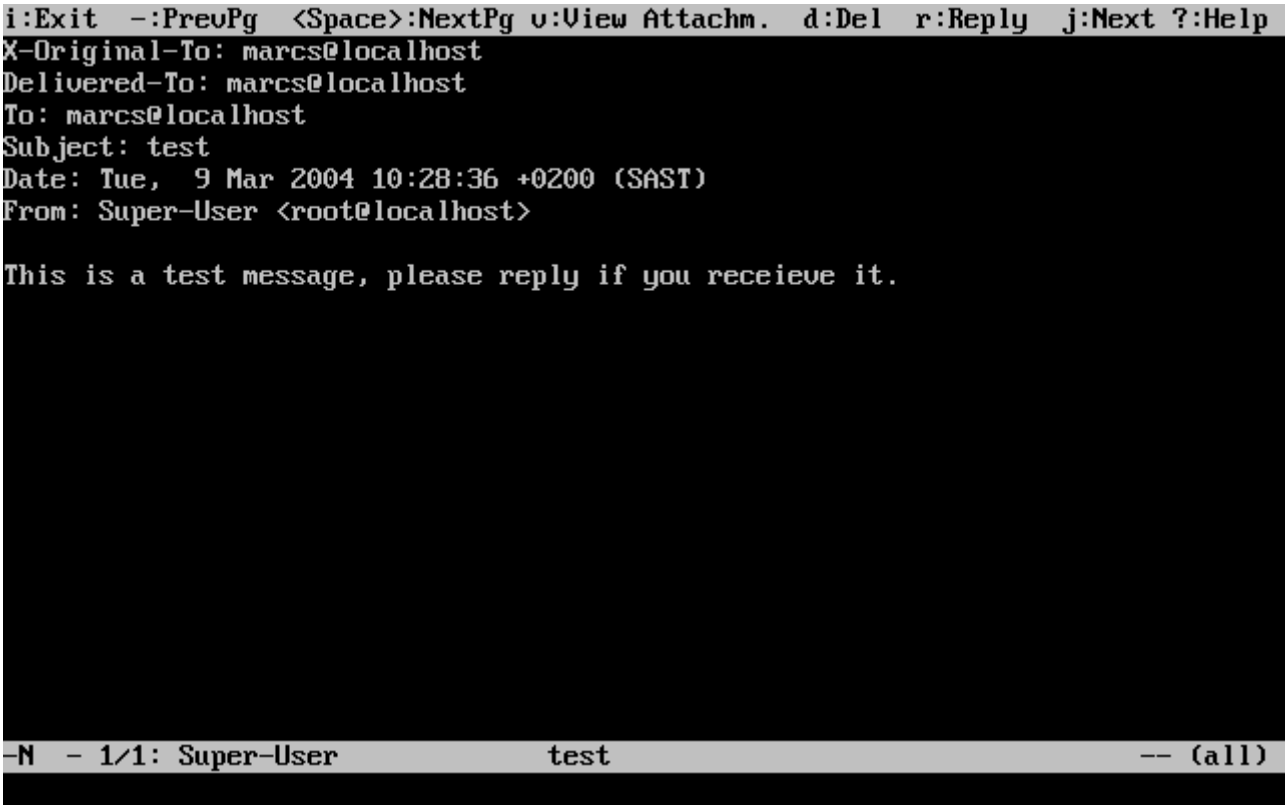
можно запустить, выполнив следующую команду:

```
% mutt
```

Mutt автоматически прочитает и отобразит содержимое почтового ящика пользователя в /var/mail. Если письма не найдены, Mutt будет ожидать команд от пользователя. В примере ниже показан список сообщений в Mutt:



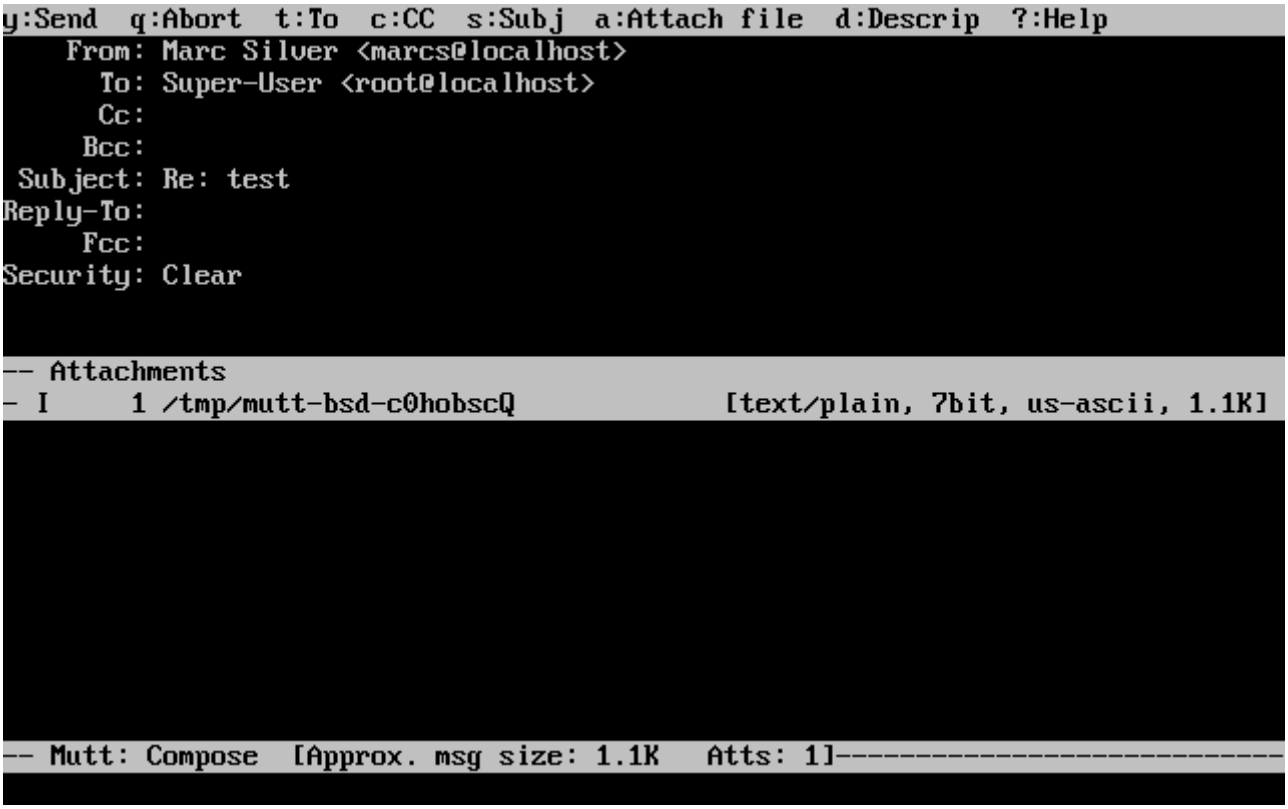
Чтобы прочитать письмо, выберите его с помощью клавиш курсора и нажмите `Enter`. Пример отображения письма в Mutt показан ниже:



Аналогично mail(1), Mutt позволяет отвечать как только отправителю сообщения, так и всем получателям. Чтобы ответить только отправителю письма, нажмите `r`. Для отправки группового ответа исходному отправителю, а также всем получателям сообщения нажмите `g`.

По умолчанию Mutt использует редактор vi(1) для создания и ответа на письма. Каждый пользователь может изменить это, создав или отредактировав файл .muttrc в своём домашнем каталоге и установив переменную `editor`, либо задав переменную окружения `EDITOR`. Дополнительную информацию о настройке Mutt можно найти на http://www.mutt.org/.

Чтобы создать новое письмо, нажмите `m`. После ввода корректной темы Mutt запустит vi(1) для написания письма. Завершив составление письма, сохраните изменения и выйдите из `vi`. Mutt продолжит работу, отобразив итоговый экран с информацией о письме, готовом к отправке. Для отправки письма нажмите `y`. Пример итогового экрана показан ниже:



Mutt содержит обширную справку, доступ к которой можно получить из большинства меню, нажав `?`. В верхней строке также отображаются сочетания клавиш, где это уместно.

32.6.3. alpine

alpine ориентирован на начинающего пользователя, но также включает некоторые расширенные возможности.

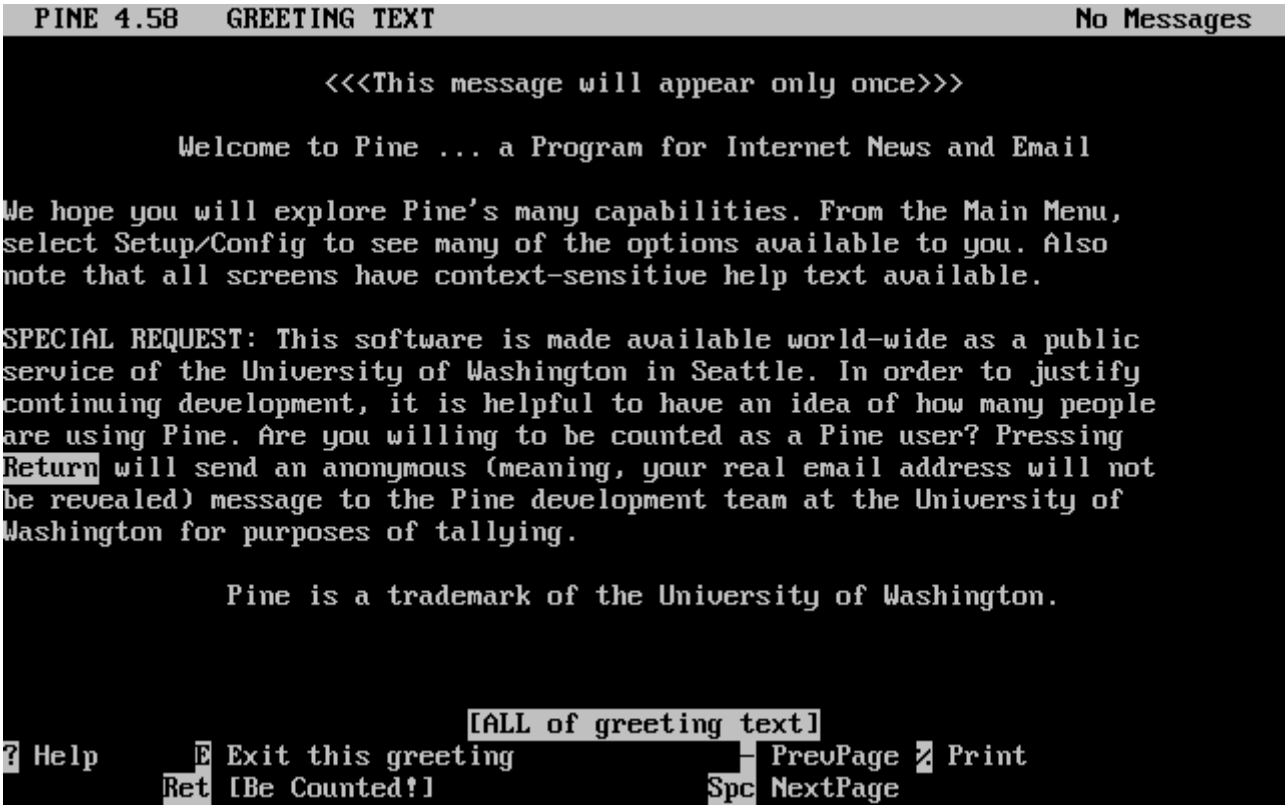
В прошлом в alpine было обнаружено несколько уязвимостей, позволяющих удалённым злоумышленникам выполнять произвольный код от имени пользователей локальной системы путём отправки специально подготовленного электронного письма. Хотя *известные* проблемы были исправлены, код alpine написан в небезопасном стиле, и Ответственный за безопасность FreeBSD считает, что, скорее всего, существуют другие необнаруженные уязвимости. Пользователи устанавливают alpine на свой страх и риск.

Текущую версию alpine можно установить с помощью порта mail/alpine. После установки порта alpine можно запустить, выполнив следующую команду:

```
% alpine
```

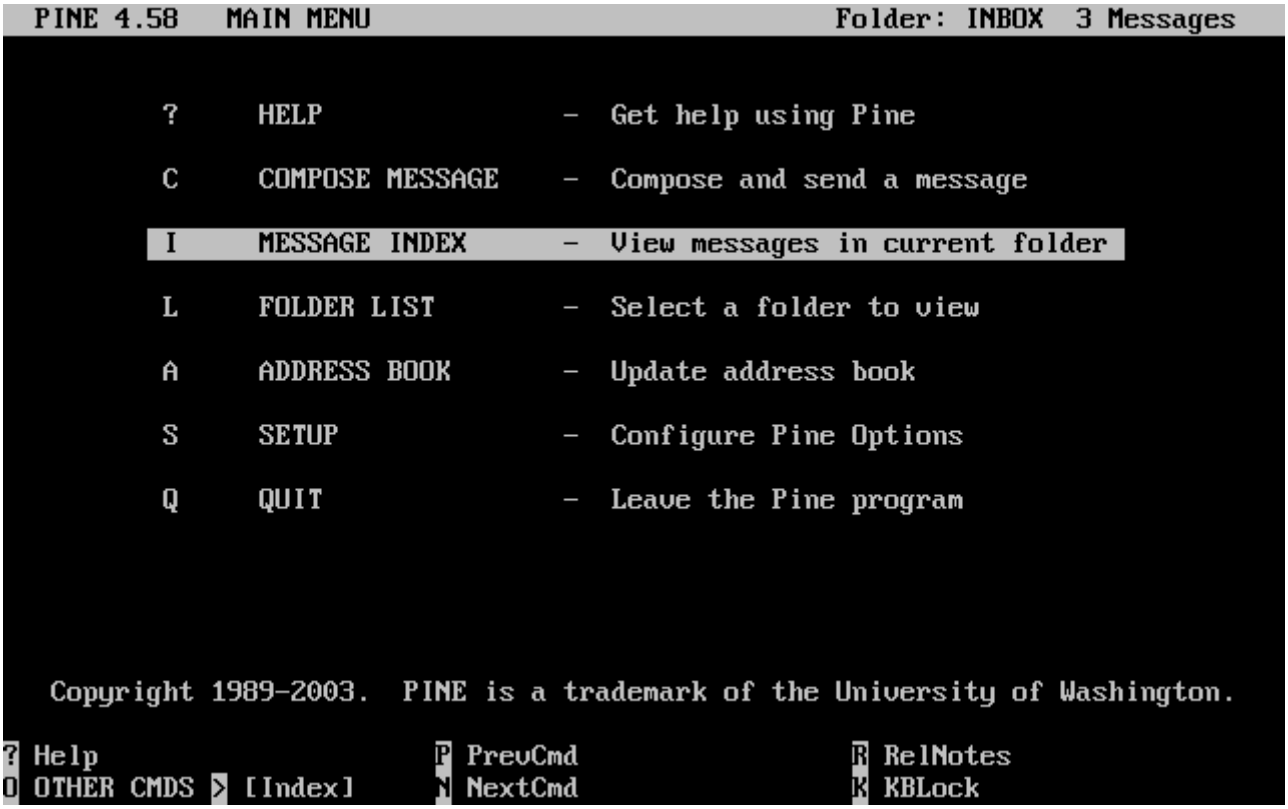
При первом запуске alpine отображает приветственную страницу с кратким введением, а также просьбу от команды разработчиков alpine отправить анонимное электронное сообщение, позволяющее им оценить количество пользователей их клиента. Чтобы отправить это анонимное сообщение, нажмите `Enter`. Или нажмите `E` для выхода из приветствия без отправки анонимного сообщения. Пример приветственной страницы

показан ниже:

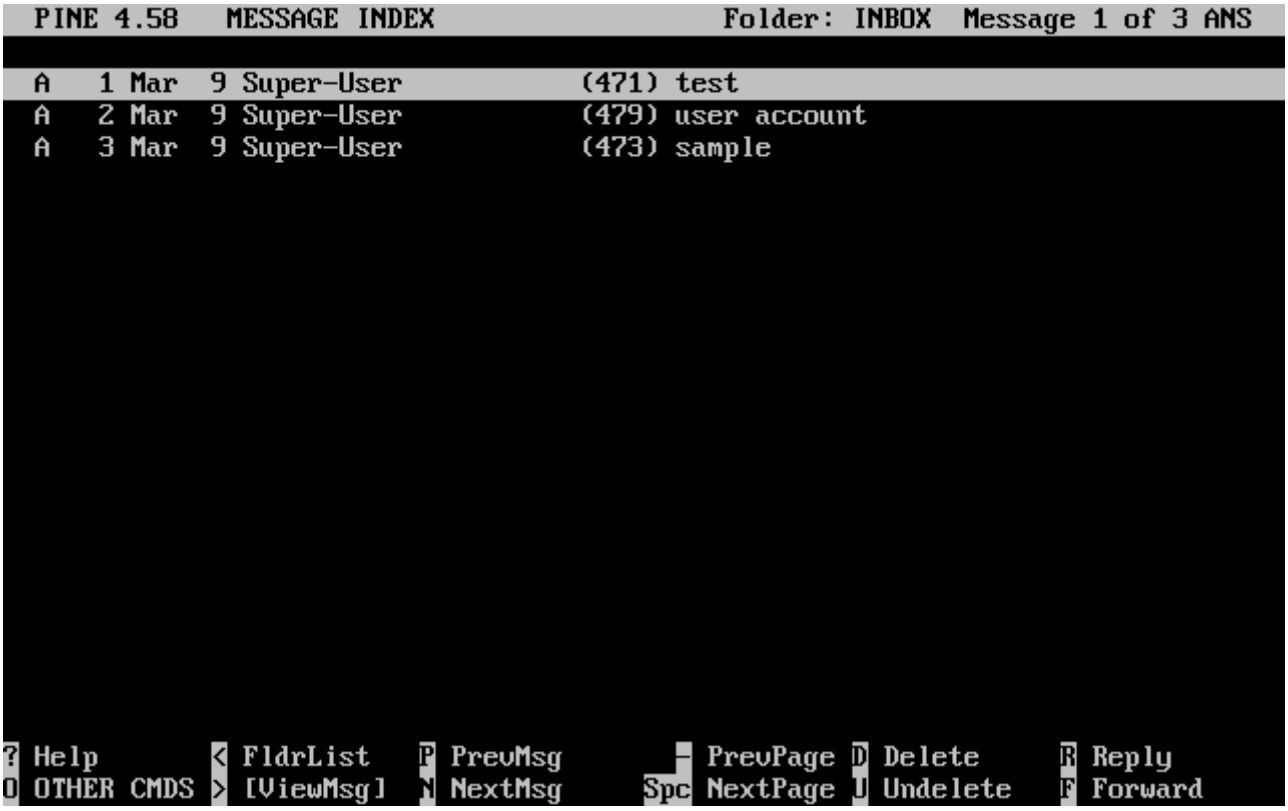


Затем отображается главное меню, в котором можно перемещаться с помощью клавиш-стрелок. Это меню предоставляет быстрый доступ к созданию новых писем, просмотру почтовых каталогов и управлению записями адресной книги. Под главным меню отображаются соответствующие сочетания клавиш для выполнения функций, связанных с текущей задачей.

По умолчанию alpine открывает папку inbox. Для просмотра списка сообщений нажмите `I` или выберите пункт MESSAGE INDEX, как показано ниже:

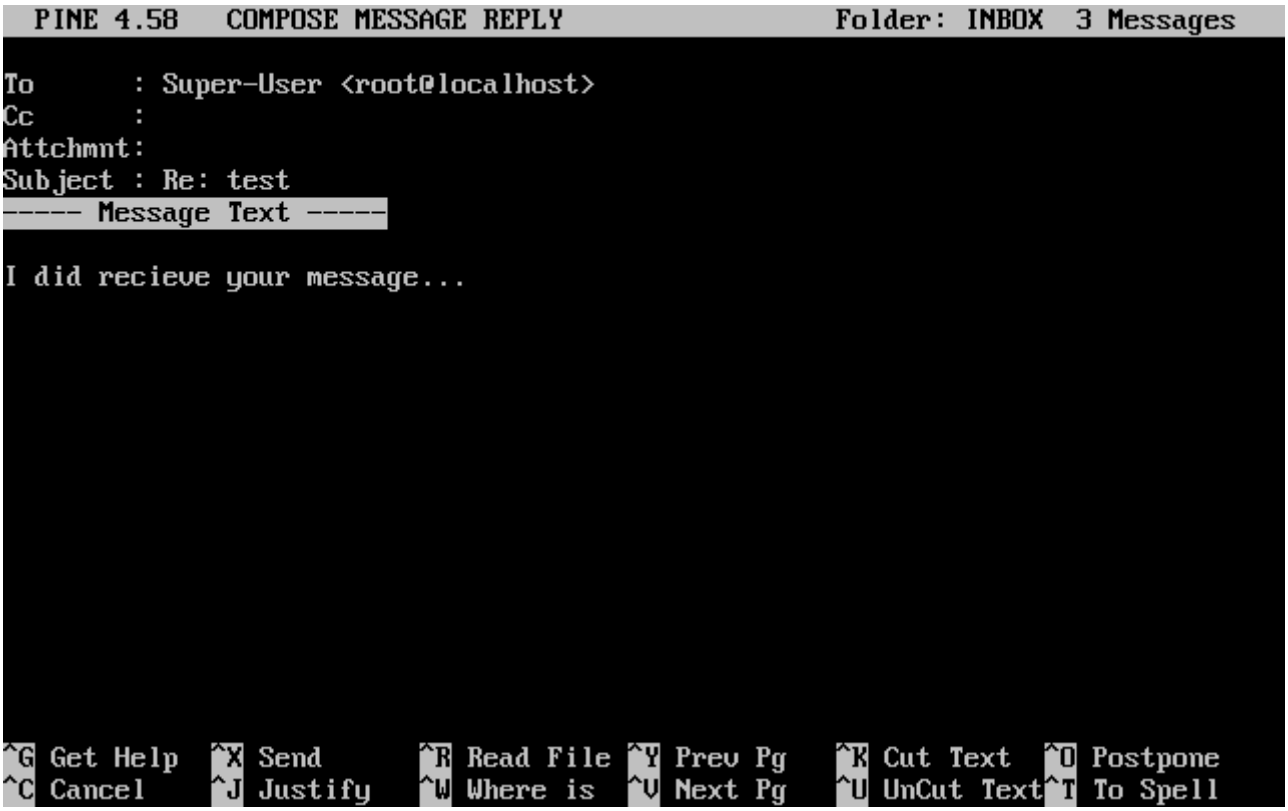


Список сообщений отображает сообщения в текущем каталоге, и по ним можно перемещаться с помощью клавиш-стрелок. Выделенные сообщения можно прочитать, нажав `Enter`.



На приведённом ниже снимке экрана отображается пример сообщения в alpine. Контекстные сочетания клавиш показаны в нижней части экрана. Пример одного из сочетаний — `r`, которое указывает почтовому клиенту ответить на текущее отображаемое сообщение.

Для ответа на письмо в alpine запускается редактор pico, который устанавливается по умолчанию вместе с alpine. pico упрощает навигацию по сообщению и более удобен для начинающих пользователей, чем vi(1) или mail(1). После завершения ответа сообщение можно отправить, нажав `Ctrl` + `X`. alpine запросит подтверждение перед отправкой сообщения.



alpine можно настроить с помощью пункта SETUP в главном меню.

32.7. Сложные темы

Этот раздел охватывает более сложные темы, такие как настройка почты и организация почтовой системы для целого домена.

32.7.1. Базовая конфигурация

Без какой-либо настройки можно отправлять электронную почту на внешние хосты, если настроен /etc/resolv.conf или сеть имеет доступ к настроенному DNS-серверу. Чтобы почта доставлялась на MTA FreeBSD, выполните одно из следующих действий:

- Запустите DNS-сервер для домена.
- Получайте почту напрямую на машину, используя её полное доменное имя.

Для того чтобы почта доставлялась напрямую на хост, у него должен быть постоянный статический IP-адрес, а не динамический. Если система находится за межсетевым экраном, он должен быть настроен для пропуска SMTP-трафика. Чтобы принимать почту напрямую на хосте, необходимо настроить одно из следующих двух условий:

- Убедитесь, что MX-запись с наименьшим номером в DNS указывает на статический IP-адрес хоста.
- Убедитесь, что в DNS для хоста нет записи MX.

Любой из вышеперечисленных вариантов позволит получать почту напрямую на хост.

Попробуйте следующее:

```
# hostname
```

Вывод должен быть похож на следующий:

```
example.FreeBSD.org
```

```
# host example.FreeBSD.org
```

Вывод должен быть похож на следующий:

```
example.FreeBSD.org has address 204.216.27.XX
```

В этом примере письма, отправленные напрямую на yourlogin@example.FreeBSD.org, должны работать без проблем, при условии что полнофункциональный MTA корректно работает на `example.FreeBSD.org`. Обратите внимание, что dma(8) не слушает порт 25 для входящих соединений и не может быть использован в данном сценарии.

Для этого примера:

```
# host example.FreeBSD.org
```

Вывод должен быть похож на следующий:

```
example.FreeBSD.org has address 204.216.27.XX
example.FreeBSD.org mail is handled (pri=10) by nevdull.FreeBSD.org
```

Вся почта, отправленная на `example.FreeBSD.org`, будет собираться на `nevdull` под тем же именем пользователя вместо прямой отправки на ваш хост.

Указанная выше информация обрабатывается DNS-сервером. DNS-запись, содержащая информацию о маршрутизации почты, называется записью почтового обменника (MX-запись). Если MX-запись отсутствует, почта будет доставлена напрямую на хост по его IP-адресу.

Запись MX для `freefall.FreeBSD.org` когда-то выглядела так:

```
freefall        MX  30  mail.crl.net
freefall        MX  40  agora.rdrop.com
freefall        MX  10  freefall.FreeBSD.org
freefall        MX  20  who.cdrom.com
```

`freefall` имел множество MX-записей. Наименьший номер MX указывает на хост, который принимает почту напрямую, если он доступен. Если он недоступен по какой-либо причине, следующий хост с меньшим номером временно примет сообщения и передаст их, когда хост с меньшим номером станет доступен.

Альтернативные MX-серверы должны иметь отдельные интернет-подключения для максимальной эффективности. Ваш провайдер может предоставить такую услугу.

32.7.2. Почта для домена

Если MTA настраивается для сети, то любая почта, отправленная на хосты в его домене, должна перенаправляться на этот MTA, чтобы пользователи могли получать свою почту на главном почтовом сервере.

Для максимального удобства учётная запись с тем же *именем пользователя* должна существовать как на MTA, так и на системе с MUA. Для создания учётных записей используйте adduser(8).

Помимо добавления локальных пользователей на хост, существуют альтернативные методы, известные как виртуальные пользователи. Такие программы, как Cyrus и Dovecot, могут быть интегрированы в MTA для управления пользователями, хранения почты, а также предоставления

> доступа через POP3 и IMAP.

Почтовый сервер (MTA) должен быть назначенным почтовым обменником для каждой рабочей станции в сети. Это делается в конфигурации DNS с помощью записи MX:

```
example.FreeBSD.org A   204.216.27.XX       ; Workstation
            MX  10 nevdull.FreeBSD.org  ; Mailhost
```

Это перенаправит почту для рабочей станции на MTA, независимо от того, куда указывает запись A. Почта отправляется на MX-хост.

Это необходимо настроить на DNS-сервере. Если в сети не запущен собственный DNS-сервер, обратитесь к интернет или DNS-провайдеру.

Ниже приведён пример виртуального хостинга электронной почты.

Рассмотрим клиента с доменом `customer1.org`, где вся почта для `customer1.org` должна отправляться на `mail.myhost.com`.

Запись DNS должна выглядеть следующим образом:

```
customer1.org       MX  10  mail.myhost.com
```

Для обработки только электронной почты домена `customer1.org` запись типа `A` *не* требуется. Однако команда `ping` для `customer1.org` не будет работать, если для этого домена не существует записи `A`.

Укажите MTA, для каких доменов и/или имён хостов следует принимать почту. Для Sendmail подойдет любой из следующих вариантов:

- Добавьте хосты в /etc/mail/local-host-names, если используется `FEATURE(use_cw_file)`.
- Добавьте строку `Cwyour.host.com` в файл /etc/sendmail.cf.

32.7.3. Настройка только для отправки

Существует множество случаев, когда требуется отправлять почту только через релейный сервер (SMTP-ретранслятор). Некоторые примеры:

- Компьютер представляет собой настольную машину, которой необходимо использовать программы, такие как mail(1), с использованием почтового ретранслятора интернет-провайдера.
- Компьютер является сервером, который не обрабатывает почту локально, а передаёт всю почту релейному серверу для обработки.

Хотя любая MTA способна выполнить эту конкретную задачу, правильная настройка полнофункциональной MTA только для обработки пересылки почты может быть сложной. Такие программы, как Sendmail и Postfix, являются избыточными для такого использования.

Кроме того, стандартное соглашение об услуге доступа в Интернет может запрещать запуск «почтового сервера».

Самый простой способ удовлетворить эти потребности — использовать MTA dma(8), входящий в состав базовой системы. Для систем версии до 13.2 необходимо установить его из портов.

В дополнение к dma(8), для достижения того же результата можно использовать стороннее программное обеспечение, например mail/ssmtp.

```
# cd /usr/ports/mail/ssmtp
# make install replace clean
```

После установки пакет mail/ssmtp можно настроить с помощью файла /usr/local/etc/ssmtp/ssmtp.conf:

```
root=yourrealemail@example.com
mailhub=mail.example.com
rewriteDomain=example.com
hostname=_HOSTNAME_
```

Используйте настоящий адрес электронной почты для `root`. Введите исходящий почтовый релейный сервер вашего интернет-провайдера вместо `mail.example.com`. Некоторые провайдеры называют это "исходящий почтовый сервер" или "SMTP-сервер".

Убедитесь, что Sendmail отключён, включая службу исходящей почты.

mail/ssmtp предоставляет дополнительные настройки. Для получения дополнительной информации обратитесь к примерам в /usr/local/etc/ssmtp или к руководству ssmtp.

Настройка ssmtp таким образом позволяет любому программному обеспечению на компьютере, которому требуется отправлять почту, работать корректно, не нарушая политику использования провайдера и не позволяя компьютеру быть захваченным для рассылки спама.

32.7.4. Аутентификация SMTP в Sendmail

Настройка аутентификации SMTP на MTA предоставляет ряд преимуществ. Аутентификация SMTP добавляет уровень безопасности в Sendmail и позволяет мобильным пользователям, меняющим хосты, использовать один и тот же MTA без необходимости перенастраивать параметры почтового клиента каждый раз.

Установите пакет security/cyrus-sasl2 из Коллекции портов. Этот порт поддерживает ряд опций на этапе компиляции. Для метода аутентификации SMTP, демонстрируемого в данном примере, убедитесь, что `LOGIN` не отключен.

После установки пакета security/cyrus-sasl2 отредактируйте файл /usr/local/lib/sasl2/Sendmail.conf или создайте его, если он не существует, и добавьте

следующую строку:

```
pwcheck_method: saslauthd
```

Далее установите security/cyrus-sasl2-saslauthd и выполните следующую команду:

```
# sysrc saslauthd_enable="YES"
```

Наконец, запустите демон saslauthd:

```
# service saslauthd start
```

Этот демон служит посредником для Sendmail при аутентификации в базе данных FreeBSD passwd(5). Это избавляет от необходимости создавать новый набор имён пользователей и паролей для каждого пользователя, которому требуется аутентификация SMTP, и позволяет сохранять одинаковые пароли для входа в систему и для почты.

Далее отредактируйте файл /etc/make.conf и добавьте следующие строки:

```
SENDMAIL_CFLAGS=-I/usr/local/include/sasl -DSASL
SENDMAIL_LDADD=/usr/local/lib/libsasl2.so
```

Эти строки предоставляют Sendmail правильные параметры конфигурации для связи с cyrus-sasl2 во время компиляции. Убедитесь, что cyrus-sasl2 установлен перед перекомпиляцией Sendmail.

Перекомпилируйте Sendmail, выполнив следующие команды:

```
# cd /usr/src/lib/libsmutil
# make cleandir && make obj && make
# cd /usr/src/lib/libsm
# make cleandir && make obj && make
# cd /usr/src/usr.sbin/sendmail
# make cleandir && make obj && make && make install
```

Эта компиляция не должна вызывать проблем, если /usr/src не подвергался значительным изменениям и необходимые динамические библиотеки доступны.

После компиляции и переустановки Sendmail отредактируйте файл /etc/mail/freebsd.mc или локальный файл .mc. Многие администраторы предпочитают использовать вывод команды hostname(1) в качестве имени файла .mc для обеспечения уникальности.

Добавьте следующие строки:

```
dnl set SASL options
TRUST_AUTH_MECH(`GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN')dnl
define(`confAUTH_MECHANISMS', `GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN')dnl
```

Эти параметры настраивают различные методы, доступные Sendmail для аутентификации пользователей. Чтобы использовать метод, отличный от pwcheck, обратитесь к документации Sendmail.

Наконец, выполните make(1) в каталоге /etc/mail. Это приведёт к обработке нового файла .mc и созданию файла .cf с именем freebsd.cf или именем, использованным для локального файла .mc.

Затем выполните `make install restart`, что скопирует файл в sendmail.cf и корректно перезапустит Sendmail.

Для получения дополнительной информации об этом процессе обратитесь к /etc/mail/Makefile.

Для проверки конфигурации используйте MUA для отправки тестового сообщения. Для более детального анализа установите уровень журналирования (`LogLevel`) Sendmail на `13` и следите за файлом /var/log/maillog на предмет ошибок.

Для получения дополнительной информации обратитесь к аутентификации SMTP.

Глава 33. Сетевые серверы

33.1. Обзор

В этой главе рассматриваются некоторые из наиболее часто используемых сетевых служб в системах UNIX®. Сюда входит установка, настройка, тестирование и поддержка различных типов сетевых служб. В этой главе приведены примеры конфигурационных файлов для справки.

К концу этой главы читатели будут знать:

- Как управлять демоном inetd.
- Как настроить Network File System (NFS).
- Как настроить сервер сетевой информации (NIS) для централизации и совместного использования учётных записей пользователей.
- Как настроить FreeBSD в качестве сервера или клиента LDAP
- Как настроить автоматические параметры сети с использованием DHCP.
- Как настроить сервер доменных имён (DNS).
- Как настроить веб-сервер Apache HTTP.
- Как настроить сервер протокола передачи файлов (FTP).
- Как настроить файловый и печатный сервер для клиентов Windows® с использованием Samba.
- Как синхронизировать время и дату, а также настроить сервер времени с использованием протокола Network Time Protocol (NTP).
- Как настроить iSCSI.

Эта глава предполагает базовые знания о:

- Скриптах /etc/rc.
- Сетевой терминологии.
- Установке дополнительного стороннего программного обеспечения (Установка приложений: Пакеты и Порты).

33.2. Суперсервер inetd

Демон inetd(8) иногда называют суперсервером, потому что он управляет соединениями для многих служб. Вместо запуска множества приложений, достаточно запустить только службу inetd. Когда поступает соединение для службы, управляемой inetd, он определяет, какому программе предназначено соединение, создаёт процесс для этой программы и делегирует программе сокет. Использование inetd для служб, которые не используются интенсивно, может снизить нагрузку на систему по сравнению с запуском каждого демона отдельно в автономном режиме.

Прежде всего, inetd используется для запуска других демонов, но несколько простых протоколов обрабатываются внутри него, таких как chargen, auth, time, echo, discard и daytime.

Этот раздел охватывает основы настройки inetd.

33.2.1. Файл конфигурации

Настройка inetd выполняется путем редактирования /etc/inetd.conf. Каждая строка этого файла конфигурации представляет приложение, которое может быть запущено inetd. По умолчанию каждая строка начинается с комментария (#), что означает, что inetd не ожидает подключений для каких-либо приложений. Чтобы настроить inetd на ожидание подключений для приложения, удалите # в начале соответствующей строки.

После сохранения изменений настройте inetd для запуска при загрузке системы, отредактировав /etc/rc.conf:

```
inetd_enable="YES"
```

Чтобы запустить inetd сейчас, чтобы он начал прослушивать настроенную службу, введите:

```
# service inetd start
```

После запуска inetd необходимо уведомлять его о каждом изменении в файле /etc/inetd.conf:

Пример 41. Перезагрузка конфигурационного файла inetd

```
# service inetd reload
```

Обычно запись по умолчанию для приложения не требует редактирования, кроме удаления #. В некоторых ситуациях может быть целесообразно изменить запись по умолчанию.

В качестве примера, это стандартная запись для ftpd(8) по IPv4:

```
ftp     stream  tcp     nowait  root    /usr/libexec/ftpd       ftpd -l
```

Семь столбцов в записи следующие:

```
service-name
socket-type
protocol
{wait|nowait}[/max-child[/max-connections-per-ip-per-minute[/max-child-per-ip]]]
user[:group][/login-class]
server-program
```

```
server-program-arguments
```

где:

service-name

Имя службы демона для запуска. Оно должно соответствовать службе, указанной в /etc/services. Это определяет, на каком порту inetd ожидает входящие соединения для этой службы. При использовании пользовательской службы она сначала должна быть добавлена в /etc/services.

socket-type

Либо `stream`, `dgram`, `raw`, или `seqpacket`. Используйте `stream` для TCP-соединений и `dgram` для UDP-сервисов.

protocol

Используйте одно из следующих названий протоколов:

Имя протокола	Объяснение
tcp или tcp4	TCP IPv4
udp или udp4	UDP IPv4
tcp6	TCP IPv6
udp6	UDP IPv6
tcp46	Как TCP IPv4, так и IPv6
udp46	Как UDP IPv4, так и IPv6

{wait|nowait}[/max-child[/max-connections-per-ip-per-minute[/max-child-per-ip]]]

В этом поле необходимо указать `wait` или `nowait`. Параметры `max-child`, `max-connections-per-ip-per-minute` и `max-child-per-ip` являются необязательными.

`wait|nowait` указывает, способна ли служба обрабатывать свой собственный сокет. Типы сокетов `dgram` должны использовать `wait`, в то время как для демонов `stream`, которые обычно многопоточные, следует использовать `nowait`. `wait` обычно передаёт несколько сокетов одному демону, тогда как `nowait` создаёт дочерний демон для каждого нового сокета.

Максимальное количество дочерних демонов, которые может породить inetd, задается параметром `max-child`. Например, чтобы ограничить десять экземпляров демона, укажите `/10` после `nowait`. Указание `/0` позволяет создавать неограниченное количество дочерних процессов.

`max-connections-per-ip-per-minute` ограничивает количество соединений с любого конкретного IP-адреса в минуту. Как только лимит достигнут, последующие соединения с этого IP-адреса будут отбрасываться до конца минуты. Например, значение `/10` ограничивает любой конкретный IP-адрес десятью попытками соединения в минуту. `max-child-per-ip` ограничивает количество дочерних процессов, которые могут быть запущены от имени любого отдельного IP-адреса в любой момент времени. Эти опции

позволяют ограничить чрезмерное потребление ресурсов и помогают предотвратить атаки типа "Отказ в обслуживании".

Пример можно увидеть в настройках по умолчанию для fingerd(8):

```
finger stream  tcp     nowait/3/10 nobody /usr/libexec/fingerd fingerd -k -s
```

user

Имя пользователя, от имени которого будет работать демон. Демоны обычно работают от имени `root`, `daemon` или `nobody`.

server-program

Полный путь к демону. Если демон является службой, предоставляемой inetd внутренне, используйте `internal`.

server-program-arguments

Используется для указания любых аргументов командной строки, передаваемых демону при его запуске. Если демон является внутренней службой, используйте `internal`.

33.2.2. Параметры командной строки

Как и большинство серверных демонов, inetd имеет ряд опций, которые можно использовать для изменения его поведения. По умолчанию inetd запускается с параметрами `-wW -C 60`. Эти опции включают TCP wrappers для всех сервисов, включая внутренние, и предотвращают запросы любого IP-адреса к любому сервису чаще 60 раз в минуту.

Для изменения параметров по умолчанию, передаваемых inetd, добавьте запись `inetd_flags` в файл /etc/rc.conf. Если inetd уже запущен, перезапустите его командой `service inetd restart`.

Доступные варианты ограничения скорости:

-c maximum

Укажите максимальное количество одновременных вызовов каждой службы по умолчанию, где по умолчанию значение не ограничено. Может быть переопределено для каждой службы отдельно с помощью параметра `max-child` в /etc/inetd.conf.

-C rate

Укажите максимальное количество вызовов службы с одного IP-адреса в минуту по умолчанию. Это значение может быть переопределено для отдельной службы с помощью параметра `max-connections-per-ip-per-minute` в файле /etc/inetd.conf.

-R rate

Укажите максимальное количество вызовов службы в течение одной минуты, где значение по умолчанию — `256`. Значение `0` позволяет неограниченное количество.

-s maximum

Укажите максимальное количество раз, которое служба может быть вызвана с одного IP-адреса одновременно, по умолчанию значение не ограничено. Может быть переопределено для каждой службы отдельно с помощью параметра `max-child-per-ip` в /etc/inetd.conf.

Доступны дополнительные параметры. Полный список параметров смотрите в inetd(8).

33.2.3. Безопасность

Многие демоны, которыми может управлять inetd, не обладают достаточной защитой. Некоторые демоны, такие как fingerd, могут предоставлять информацию, полезную для злоумышленника. Включайте только необходимые службы и отслеживайте систему на предмет чрезмерных попыток подключения. Параметры `max-connections-per-ip-per-minute`, `max-child` и `max-child-per-ip` могут быть использованы для ограничения подобных атак.

По умолчанию TCP wrappers включены. Дополнительную информацию о наложении TCP-ограничений на различные демоны, запускаемые через inetd, можно найти в hosts_access(5).

33.3. Сетевая файловая система (NFS — Network File System)

FreeBSD поддерживает Network File System (NFS), что позволяет серверу делиться каталогами и файлами с клиентами по сети. С помощью NFS пользователи и программы могут обращаться к файлам на удалённых системах так, как если бы они хранились локально.

NFS имеет множество практических применений. Некоторые из наиболее распространённых вариантов использования включают:

- Данные, которые в противном случае дублировались бы на каждом клиенте, могут храниться в одном месте и быть доступными для клиентов в сети.
- Несколько клиентов могут нуждаться в доступе к каталогу /usr/ports/distfiles. Общий доступ к этому каталогу позволяет быстро получить исходные файлы без необходимости загрузки их на каждый клиент.
- На крупных сетях часто удобнее настроить центральный NFS-сервер, на котором хранятся все домашние каталоги пользователей. Пользователи могут входить в систему с любого клиента в сети и получать доступ к своим домашним каталогам.
- Управление экспортом NFS упрощено. Например, существует только одна файловая система, в которой необходимо настраивать политики безопасности или резервного копирования.
- Съемные устройства хранения данных могут использоваться другими компьютерами в сети. Это уменьшает количество устройств в сети и обеспечивает централизованное управление их безопасностью. Часто бывает удобнее устанавливать программное обеспечение на несколько компьютеров с централизованного носителя для установки.

NFS состоит из сервера и одного или нескольких клиентов. Клиент удалённо получает доступ к данным, хранящимся на машине сервера. Для корректной работы необходимо настроить и запустить несколько процессов.

Эти демоны должны быть запущены на сервере:

Демон	Описание
nfsd	Демон NFS, обслуживающий запросы от клиентов NFS.
mountd	Демон монтирования NFS, который выполняет запросы, полученные от nfsd.
rpcbind	Этот демон позволяет клиентам NFS определять, какой порт использует сервер NFS.

Запуск nfsiod(8) на клиенте может повысить производительность, но не является обязательным.

33.3.1. Настройка сервера

Файловые системы, которые сервер NFS будет предоставлять в общий доступ, указаны в /etc/exports. Каждая строка в этом файле определяет файловую систему для экспорта, клиентов, которые имеют доступ к этой файловой системе, и любые параметры доступа. При добавлении записей в этот файл каждая экспортируемая файловая система, её свойства и разрешённые хосты должны быть указаны в одной строке. Если в записи не указаны клиенты, то любой клиент в сети может подключить эту файловую систему.

Следующие записи в /etc/exports демонстрируют, как экспортировать файловые системы. Примеры могут быть изменены в соответствии с файловыми системами и именами клиентов в сети читателя. В этом файле можно использовать множество опций, но здесь упомянуты лишь некоторые. Полный список опций смотрите в exports(5).

В этом примере показано, как экспортировать /media на три хоста с именами *alpha*, *bravo* и *charlie*:

```
/media -ro alpha bravo charlie
```

Флаг `-ro` делает файловую систему доступной только для чтения, предотвращая внесение клиентами изменений в экспортированную файловую систему. В этом примере предполагается, что имена хостов находятся либо в DNS, либо в /etc/hosts. Обратитесь к hosts(5), если в сети нет DNS-сервера.

Следующий пример экспортирует /home трём клиентам по IP-адресу. Это может быть полезно для сетей без DNS или записей в /etc/hosts. Флаг `-alldirs` позволяет подкаталогам быть точками монтирования. Другими словами, он не будет автоматически монтировать подкаталоги, но разрешит клиенту монтировать необходимые каталоги по мере надобности.

```
/usr/home  -alldirs  10.0.0.2 10.0.0.3 10.0.0.4
```

Следующий пример экспортирует /a, чтобы два клиента из разных доменов могли получить доступ к этой файловой системе. Параметр `-maproot=root` позволяет пользователю `root` на удалённой системе записывать данные в экспортированную файловую систему как `root`. Если параметр `-maproot=root` не указан, пользователь `root` на клиенте будет отображён на учётную запись `nobody` на сервере и будет ограничен правами доступа, определёнными для `nobody`.

```
/a  -maproot=root  host.example.com box.example.org
```

Клиент может быть указан только один раз для каждой файловой системы. Например, если /usr представляет собой одну файловую систему, следующие записи будут недопустимыми, так как обе указывают на один и тот же узел:

```
# Invalid when /usr is one file system
/usr/src   client
/usr/ports client
```

Правильный формат для данной ситуации — использовать одну запись:

```
/usr/src /usr/ports  client
```

Ниже приведён пример корректного списка экспорта, где /usr и /exports являются локальными файловыми системами:

```
# Export src and ports to client01 and client02, but only
# client01 has root privileges on it
/usr/src /usr/ports -maproot=root    client01
/usr/src /usr/ports               client02
# The client machines have root and can mount anywhere
# on /exports. Anyone in the world can mount /exports/obj read-only
/exports -alldirs -maproot=root      client01 client02
/exports/obj -ro
```

Чтобы включить процессы, необходимые для работы сервера NFS при загрузке, добавьте следующие параметры в /etc/rc.conf:

```
rpcbind_enable="YES"
nfs_server_enable="YES"
mountd_enable="YES"
```

Сервер можно запустить, выполнив следующую команду:

```
# service nfsd start
```

Всякий раз, когда запускается сервер NFS, также автоматически запускается mountd. Однако mountd читает /etc/exports только при запуске. Чтобы последующие изменения в /etc/exports вступили в силу немедленно, заставьте mountd перечитать его:

```
# service mountd reload
```

Обратитесь к zfs-share(8) для описания экспорта наборов данных ZFS через NFS с использованием свойства ZFS `sharenfs` вместо файла exports(5).

Обратитесь к nfsv4(4) для описания настройки NFS версии 4.

33.3.2. Настройка клиента

Чтобы включить клиенты NFS, установите эту опцию в файле /etc/rc.conf каждого клиента:

```
nfs_client_enable="YES"
```

Затем выполните эту команду на каждом клиенте NFS:

```
# service nfsclient start
```

Клиент теперь имеет всё необходимое для монтирования удалённой файловой системы. В этих примерах имя сервера — `server`, а имя клиента — `client`. Чтобы смонтировать /home с сервера `server` в точку монтирования /mnt на клиенте `client`:

```
# mount server:/home /mnt
```

Файлы и каталоги в /home теперь будут доступны на `client`, в каталоге /mnt.

Для монтирования удалённой файловой системы при каждой загрузке клиента добавьте её в /etc/fstab:

```
server:/home    /mnt    nfs rw  0   0
```

Обратитесь к fstab(5) для описания всех доступных опций.

33.3.3. Блокировка

Некоторые приложения требуют блокировки файлов для корректной работы. Чтобы включить блокировку, выполните следующую команду как на клиенте, так и на сервере:

```
# sysrc rpc_lockd_enable="YES"
```

Затем запустите службу rpc.lockd(8):

```
# service lockd start
```

Если блокировка не требуется на сервере, клиент NFS можно настроить для локальной блокировки, добавив параметр `-L` при выполнении команды mount. Дополнительные сведения см. в mount_nfs(8).

33.3.4. Автоматизация монтирования с помощью autofs(5)

Автомонтирование autofs(5) поддерживается начиная с FreeBSD 10.1-RELEASE. Для использования функциональности автомонтирования в более старых версиях FreeBSD используйте amd(8). В этой главе описывается только автомонтирование autofs(5).

Утилита autofs(5) — это общее название для нескольких компонентов, которые вместе позволяют автоматически монтировать удалённые и локальные файловые системы при обращении к файлу или каталогу внутри этих файловых систем. Она состоит из компонента ядра autofs(5) и нескольких пользовательских приложений: automount(8), automountd(8) и autounmountd(8). Она служит альтернативой для amd(8) из предыдущих выпусков FreeBSD. amd по-прежнему предоставляется для обратной совместимости, так как эти утилиты используют разные форматы карт; формат, используемый autofs, совпадает с форматом других автомонтировщиков SVR4, таких как в Solaris, MacOS X и Linux.

Виртуальная файловая система autofs(5) монтируется на указанные точки монтирования с помощью automount(8), который обычно запускается во время загрузки.

Всякий раз, когда процесс пытается получить доступ к файлу в точке монтирования autofs(5), ядро уведомляет демон automountd(8) и приостанавливает вызвавший процесс. Демон automountd(8) обрабатывает запросы ядра, находя соответствующую карту и монтируя файловую систему в соответствии с ней, после чего сигнализирует ядру о разблокировке процесса. Демон autounmountd(8) автоматически размонтирует автомонтируемые файловые системы по истечении некоторого времени, если они больше не используются.

Основной файл конфигурации autofs — это /etc/auto_master. Он связывает отдельные карты с корневыми точками монтирования. Для объяснения синтаксиса auto_master и карт обратитесь к auto_master(5).

Существует специальная карта автомонтирования, смонтированная в /net. При обращении к файлу в этом каталоге, autofs(5) ищет соответствующую удалённую точку монтирования и автоматически монтирует её. Например, попытка доступа к файлу в /net/foobar/usr приведёт к тому, что automountd(8) смонтирует экспорт /usr с хоста `foobar`.

Пример 42. Подключение экспорта с помощью autofs(5)

В этом примере `showmount -e` показывает экспортированные файловые системы, которые могут быть подключены с NFS-сервера `foobar`:

```
% showmount -e foobar
Exports list on foobar:
/usr                               10.10.10.0
/a                                 10.10.10.0
% cd /net/foobar/usr
```

Результат выполнения `showmount` показывает, что /usr экспортируется. При переходе в каталог /host/foobar/usr, automountd(8) перехватывает запрос и пытается разрешить имя хоста `foobar`. В случае успеха automountd(8) автоматически монтирует исходный экспорт.

Чтобы включить autofs(5) при загрузке, добавьте следующую строку в /etc/rc.conf:

```
autofs_enable="YES"
```

Затем autofs(5) может быть запущен выполнением:

```
# service automount start
# service automountd start
# service autounmountd start
```

Формат карты autofs(5) такой же, как и в других операционных системах. Информация об этом формате из других источников может быть полезной, например, из документации Mac OS X.

Обратитесь к справочным страницам automount(8), automountd(8), autounmountd(8) и auto_master(5) для получения дополнительной информации.

33.4. Сетевая информационная система (NIS)

Сетевая информационная система (NIS — Network Information System) предназначена для централизованного администрирования UNIX®-подобных систем, таких как Solaris™, HP-UX, AIX®, Linux, NetBSD, OpenBSD и FreeBSD. Изначально NIS была известна как Yellow Pages, но название было изменено из-за проблем с товарными знаками. Именно поэтому команды NIS начинаются с `yp`.

NIS — это клиент-серверная система на основе удалённых вызовов процедур (RPC), которая позволяет группе машин в домене NIS использовать общий набор конфигурационных файлов. Это позволяет системному администратору настраивать клиентские системы NIS с минимальным объёмом конфигурационных данных, а также добавлять, удалять или изменять конфигурационные данные из единого места.

FreeBSD использует вторую версию протокола NIS.

33.4.1. Термины и процессы NIS

Таблица 28.1 обобщает термины и важные процессы, используемые NIS:

Таблица 47. Терминология NIS

Термин	Описание
Имя домена NIS	Серверы и клиенты NIS используют общее имя домена NIS. Как правило, это имя не связано с DNS.
rpcbind(8)	Эта служба включает RPC и должна работать для запуска сервера NIS или работы в качестве клиента NIS.
ypbind(8)	Эта служба связывает клиент NIS с его сервером NIS. Она принимает имя домена NIS и использует RPC для подключения к серверу. Это основа клиент-серверного взаимодействия в среде NIS. Если эта служба не запущена на клиентской машине, она не сможет получить доступ к серверу NIS.
ypserv(8)	Это процесс для сервера NIS. Если эта служба перестанет работать, сервер больше не сможет отвечать на запросы NIS, поэтому, надеюсь, есть подчиненный сервер, который возьмет на себя управление. Некоторые клиенты, не относящиеся к FreeBSD, не будут пытаться переподключиться с использованием подчиненного сервера, и процесс ypbind, возможно, потребуется перезапустить на этих клиентах.
rpc.yppasswdd(8)	Этот процесс работает только на основных серверах NIS. Этот демон позволяет клиентам NIS изменять свои пароли в NIS. Если этот демон не запущен, пользователям придется входить на главный сервер NIS и изменять пароли там.

33.4.2. Типы машин

В среде NIS существует три типа хостов:

- Основной сервер NIS

  Этот сервер выступает в роли центрального хранилища информации о конфигурации хостов и содержит авторитетные копии файлов, используемых всеми клиентами NIS.

Файлы passwd, group и другие, используемые клиентами NIS, хранятся на главном сервере. Хотя возможно, чтобы одна машина была основным сервером NIS для нескольких доменов NIS, такая конфигурация не рассматривается в этой главе, так как предполагается относительно небольшая среда NIS.

- Подчиненные серверы NIS

  Подчинённые серверы NIS хранят копии файлов данных NIS главного сервера для обеспечения избыточности. Подчинённые серверы также помогают распределить нагрузку основного сервера, так как клиенты NIS всегда подключаются к NIS серверу, который отвечает первым.

- Клиенты NIS

  Клиенты NIS проходят аутентификацию на сервере NIS при входе в систему.

Информация из многих файлов может быть совместно использована с помощью NIS. Файлы master.passwd, group и hosts часто распространяются через NIS. Когда процессу на клиенте требуется информация, которая обычно находится в этих файлах локально, он отправляет запрос к связанному с ним NIS-серверу.

33.4.3. Планирование и подготовка

В этом разделе описывается пример среды NIS, состоящей из 15 машин FreeBSD без централизованной точки администрирования. На каждой машине есть свои файлы /etc/passwd и /etc/master.passwd. Эти файлы синхронизируются между собой только вручную. В настоящее время, когда в лабораторию добавляется новый пользователь, этот процесс необходимо повторять на всех 15 машинах.

Конфигурация лаборатории будет следующей:

Имя машины	IP-адрес	Роль машины
`ellington`	`10.0.0.2`	Основной сервер NIS
`coltrane`	`10.0.0.3`	Подчиненный сервер NIS
`basie`	`10.0.0.4`	Факультетская рабочая станция
`bird`	`10.0.0.5`	Клиентская машина
`cli[1-11]`	`10.0.0.[6-17]`	Другие клиентские машины

Если это первый раз, когда разрабатывается схема NIS, её следует тщательно спланировать заранее. Независимо от размера сети, в процессе планирования необходимо принять несколько решений.

Выбор имени домена NIS

Когда клиент рассылает широковещательные запросы на получение информации, он включает имя домена NIS, к которому принадлежит. Таким образом, несколько серверов в одной сети могут определить, какой сервер должен отвечать на конкретный запрос.

Думайте о доменном имени NIS как об имени для группы хостов.

Некоторые организации предпочитают использовать своё доменное имя интернета в качестве имени домена NIS. Это не рекомендуется, так как может вызвать путаницу при попытках отладки сетевых проблем. Имя домена NIS должно быть уникальным в пределах сети, и полезно, если оно описывает группу машин, которую представляет. Например, художественный отдел компании Acme Inc. может находиться в домене NIS "acme-art". В этом примере будет использоваться имя домена `test-domain`.

Однако некоторые операционные системы, отличные от FreeBSD, требуют, чтобы имя домена NIS совпадало с именем интернет-домена. Если одна или несколько машин в сети имеют это ограничение, *необходимо* использовать имя интернет-домена в качестве имени домена NIS.

Требования к физическому серверу

Есть несколько моментов, которые следует учитывать при выборе машины для использования в качестве сервера NIS. Поскольку клиенты NIS зависят от доступности сервера, следует выбрать машину, которая не перезагружается часто. Идеально, чтобы сервер NIS был отдельной машиной, единственной целью которой является быть сервером NIS. Если сеть не сильно загружена, допустимо разместить сервер NIS на машине, где выполняются другие службы. Однако, если сервер NIS станет недоступен, это негативно скажется на всех клиентах NIS.

33.4.4. Настройка основного сервера NIS

Канонические копии всех NIS-файлов хранятся на основном сервере. Базы данных, используемые для хранения информации, называются NIS-картами. В FreeBSD эти карты хранятся в /var/yp/[domainname], где [domainname] — это имя NIS-домена. Поскольку поддерживается несколько доменов, возможно наличие нескольких каталогов, по одному для каждого домена. Каждый домен будет иметь свой независимый набор карт.

Основные и подчинённые серверы NIS обрабатывают все запросы NIS через ypserv(8). Этот демон отвечает за приём входящих запросов от клиентов NIS, преобразование запрошенного домена и имени карты в путь к соответствующему файлу базы данных и передачу данных из базы обратно клиенту.

Настройка основного NIS-сервера может быть относительно простой, в зависимости от потребностей окружения. Поскольку FreeBSD предоставляет встроенную поддержку NIS, её достаточно включить, добавив следующие строки в /etc/rc.conf:

```
nisdomainname="test-domain" ①
nis_server_enable="YES"     ②
nis_yppasswdd_enable="YES"  ③
```

① Эта строка устанавливает имя домена NIS в `test-domain`.

② Это автоматизирует запуск процессов сервера NIS при загрузке системы.

③ Это включает демон rpc.yppasswdd(8), позволяющий пользователям изменять свой NIS-

пароль с клиентской машины.

В многосерверном домене, где серверные машины также являются клиентами NIS, необходимо соблюдать осторожность. Обычно рекомендуется принудительно заставлять серверы привязываться к самим себе, а не разрешать им рассылать запросы на привязку и потенциально привязываться друг к другу. Могут возникнуть странные режимы сбоев, если один сервер выйдет из строя, а другие будут зависеть от него. В конечном итоге все клиенты превысят время ожидания и попытаются привязаться к другим серверам, но задержка может быть значительной, а режим сбоя сохранится, поскольку серверы могут снова привязаться друг к другу.

Сервер, который также является клиентом, может быть принудительно привязан к определённому серверу путём добавления следующих строк в /etc/rc.conf:

```
nis_client_enable="YES"             ①
nis_client_flags="-S test-domain,server"    ②
```

① Это позволяет также запускать клиентские приложения.

② Эта строка устанавливает имя домена NIS в `test-domain` и привязывает к себе.

После сохранения изменений введите `/etc/netstart`, чтобы перезапустить сеть и применить значения, указанные в /etc/rc.conf. Перед инициализацией карт NIS запустите ypserv(8):

```
# service ypserv start
```

Инициализация карт NIS

NIS-карты создаются из конфигурационных файлов в /etc на NIS-мастере, за исключением одного: /etc/master.passwd. Это сделано для предотвращения распространения паролей на все серверы в NIS-домене. Поэтому перед инициализацией NIS-карт необходимо настроить основные файлы паролей:

```
# cp /etc/master.passwd /var/yp/master.passwd
# cd /var/yp
# vi master.passwd
```

Рекомендуется удалить все записи системных учётных записей, а также любые пользовательские учётные записи, которые не нужно распространять на клиенты NIS, такие как `root` и другие административные учётные записи.

Убедитесь, что файл /var/yp/master.passwd не доступен для чтения группе или всем, установив его права доступа на `600`.

После завершения этой задачи инициализируйте карты NIS. FreeBSD включает скрипт ypinit(8) для этого. При создании карт для главного сервера укажите `-m` и задайте имя домена NIS:

```
ellington# ypinit -m test-domain
Server Type: MASTER Domain: test-domain
Creating an YP server will require that you answer a few questions.
Questions will all be asked at the beginning of the procedure.
Do you want this procedure to quit on non-fatal errors? [y/n: n] n
Ok, please remember to go back and redo manually whatever fails.
If not, something might not work.
At this point, we have to construct a list of this domains YP servers.
rod.darktech.org is already known as master server.
Please continue to add any slave servers, one per line. When you are
done with the list, type a <control D>.
master server   :  ellington
next host to add:  coltrane
next host to add:  ^D
The current list of NIS servers looks like this:
ellington
coltrane
Is this correct?  [y/n: y] y

[..output from map generation..]

NIS Map update completed.
ellington has been setup as an YP master server without any errors.
```

Это создаст файл /var/yp/Makefile на основе /var/yp/Makefile.dist. По умолчанию этот файл предполагает, что в окружении есть единственный NIS-сервер только с клиентами FreeBSD. Поскольку у `test-domain` есть подчиненный сервер, отредактируйте эту строку в /var/yp/Makefile, чтобы она начиналась с комментария (`#`):

```
NOPUSH = "True"
```

Добавление новых пользователей

Каждый раз при создании нового пользователя учётная запись должна быть добавлена на основной NIS-сервер, а NIS-карты должны быть перестроены. До этого новый пользователь не сможет войти в систему нигде, кроме главного NIS-сервера. Например, чтобы добавить нового пользователя `jsmith` в домен `test-domain`, выполните следующие команды на основном сервере:

```
# pw useradd jsmith
# cd /var/yp
# make test-domain
```

Пользователь также может быть добавлен с помощью `adduser jsmith` вместо `pw useradd smith`.

33.4.5. Настройка подчиненного сервера NIS

Для настройки подчиненного сервера NIS войдите на подчиненный сервер и отредактируйте /etc/rc.conf, как для основного сервера. Не генерируйте карты NIS, так как они уже существуют на основном сервере. При запуске `ypinit` на подчиненном сервере используйте `-s` (для подчиненного) вместо `-m` (для основного). Эта опция требует указания имени основного сервера NIS в дополнение к имени домена, как показано в этом примере:

```
coltrane# ypinit -s ellington test-domain

Server Type: SLAVE Domain: test-domain Master: ellington

Creating an YP server will require that you answer a few questions.
Questions will all be asked at the beginning of the procedure.

Do you want this procedure to quit on non-fatal errors? [y/n: n]  n

Ok, please remember to go back and redo manually whatever fails.
If not, something might not work.
There will be no further questions. The remainder of the procedure
should take a few minutes, to copy the databases from ellington.
Transferring netgroup...
ypxfr: Exiting: Map successfully transferred
Transferring netgroup.byuser...
ypxfr: Exiting: Map successfully transferred
Transferring netgroup.byhost...
ypxfr: Exiting: Map successfully transferred
Transferring master.passwd.byuid...
ypxfr: Exiting: Map successfully transferred
Transferring passwd.byuid...
ypxfr: Exiting: Map successfully transferred
Transferring passwd.byname...
ypxfr: Exiting: Map successfully transferred
Transferring group.bygid...
ypxfr: Exiting: Map successfully transferred
Transferring group.byname...
ypxfr: Exiting: Map successfully transferred
Transferring services.byname...
ypxfr: Exiting: Map successfully transferred
Transferring rpc.bynumber...
ypxfr: Exiting: Map successfully transferred
Transferring rpc.byname...
ypxfr: Exiting: Map successfully transferred
Transferring protocols.byname...
ypxfr: Exiting: Map successfully transferred
Transferring master.passwd.byname...
ypxfr: Exiting: Map successfully transferred
Transferring networks.byname...
ypxfr: Exiting: Map successfully transferred
Transferring networks.byaddr...
```

```
ypxfr: Exiting: Map successfully transferred
Transferring netid.byname...
ypxfr: Exiting: Map successfully transferred
Transferring hosts.byaddr...
ypxfr: Exiting: Map successfully transferred
Transferring protocols.bynumber...
ypxfr: Exiting: Map successfully transferred
Transferring ypservers...
ypxfr: Exiting: Map successfully transferred
Transferring hosts.byname...
ypxfr: Exiting: Map successfully transferred

coltrane has been setup as an YP slave server without any errors.
Remember to update map ypservers on ellington.
```

Это создаст каталог на подчиненном сервере с именем /var/yp/test-domain, который содержит копии карт основного сервера NIS. Добавление этих записей в /etc/crontab на каждом подчиненном сервере заставит их синхронизировать свои карты с картами на основном сервере:

```
20      *       *       *       *       root    /usr/libexec/ypxfr passwd.byname
21      *       *       *       *       root    /usr/libexec/ypxfr passwd.byuid
```

Эти записи не являются обязательными, поскольку основной сервер автоматически пытается передать любые изменения карт своим подчинённым серверам. Однако, поскольку клиенты могут зависеть от подчинённого сервера для предоставления корректной информации о паролях, рекомендуется принудительно выполнять частые обновления карт паролей. Это особенно важно в загруженных сетях, где обновления карт могут не всегда завершаться.

Для завершения настройки выполните `/etc/netstart` на подчинённом сервере, чтобы запустить службы NIS.

33.4.6. Настройка клиента NIS

Клиент NIS связывается с сервером NIS с помощью ypbind(8). Этот демон рассылает RPC-запросы в локальной сети. Эти запросы указывают доменное имя, настроенное на клиенте. Если NIS-сервер в том же домене получает один из таких запросов, он отвечает, и ypbind записывает адрес сервера. Если доступно несколько серверов, клиент будет использовать адрес первого ответившего сервера и направлять все свои NIS-запросы к нему. Клиент автоматически отправляет ping-запросы серверу через регулярные промежутки времени, чтобы убедиться, что он всё ещё доступен. Если ответ не получен в разумные сроки, ypbind пометит домен как несвязанный и снова начнёт рассылку запросов в надежде найти другой сервер.

Для настройки машины FreeBSD в качестве клиента NIS:

1. Отредактируйте файл /etc/rc.conf и добавьте следующие строки, чтобы установить имя домена NIS и запустить ypbind(8) при старте сети:

```
nisdomainname="test-domain"
nis_client_enable="YES"
```

2. Для импорта всех возможных записей паролей с сервера NIS, используйте `vipw`, чтобы удалить все учётные записи пользователей, кроме одной, из /etc/master.passwd. При удалении учётных записей учитывайте, что хотя бы одна локальная учётная запись должна остаться, и эта учётная запись должна быть членом группы `wheel`. Если возникнут проблемы с NIS, эту локальную учётную запись можно использовать для удалённого входа, получения прав суперпользователя и устранения проблемы. Перед сохранением изменений добавьте следующую строку в конец файла:

```
+::::::::::
```

   Эта строка настраивает клиент для предоставления любому пользователю с действительной учётной записью в картах паролей NIS-сервера учётной записи на клиенте. Существует множество способов настройки NIS-клиента путём изменения этой строки. Один из методов описан в Использование групп сети. Для более подробного ознакомления обратитесь к книге `Managing NFS and NIS`, опубликованной O'Reilly Media.

3. Для импорта всех возможных записей групп с сервера NIS добавьте следующую строку в /etc/group:

```
+:*::
```

Для немедленного запуска клиента NIS выполните следующие команды от имени суперпользователя:

```
# /etc/netstart
# service ypbind start
```

После выполнения этих шагов, выполнение команды `ypcat passwd` на клиенте должно отобразить карту passwd сервера.

33.4.7. Безопасность NIS

Поскольку RPC — это широковещательный сервис, любая система, запускающая ypbind в том же домене, может получить содержимое NIS-карт. Чтобы предотвратить несанкционированные операции, ypserv(8) поддерживает функцию под названием

"securenets", которая может использоваться для ограничения доступа к определённому набору хостов. По умолчанию эта информация хранится в /var/yp/securenets, если только ypserv(8) не запущен с ключом `-p` и альтернативным путём. Этот файл содержит записи, состоящие из спецификации сети и сетевой маски, разделённых пробелами. Строки, начинающиеся с `"#"`, считаются комментариями. Пример файла securenets может выглядеть так:

```
# allow connections from local host -- mandatory
127.0.0.1     255.255.255.255
# allow connections from any host
# on the 192.168.128.0 network
192.168.128.0 255.255.255.0
# allow connections from any host
# between 10.0.0.0 to 10.0.15.255
# this includes the machines in the testlab
10.0.0.0      255.255.240.0
```

Если ypserv(8) получает запрос от адреса, соответствующего одному из этих правил, он обработает запрос как обычно. Если адрес не соответствует ни одному правилу, запрос будет проигнорирован и в журнал будет записано предупреждение. Если файл securenets не существует, `ypserv` разрешит соединения с любого хоста.

TCP Wrapper — это альтернативный механизм контроля доступа вместо securenets. Хотя оба механизма контроля доступа добавляют некоторый уровень безопасности, они оба уязвимы к атакам "подмены IP". Весь трафик, связанный с NIS, должен блокироваться на межсетевом экране.

Серверы, использующие securenets, могут не обслуживать легитимных клиентов NIS с устаревшими реализациями TCP/IP. Некоторые из этих реализаций устанавливают все биты хоста в ноль при выполнении широковещательных запросов или не учитывают маску подсети при вычислении широковещательного адреса. Хотя некоторые из этих проблем можно устранить, изменив конфигурацию клиента, другие проблемы могут потребовать вывода из эксплуатации этих клиентских систем или отказа от securenets.

Использование TCP Wrapper увеличивает задержку сервера NIS. Дополнительная задержка может быть достаточно длительной, чтобы вызвать таймауты в клиентских программах, особенно в загруженных сетях с медленными серверами NIS. Если один или несколько клиентов страдают от задержек, преобразуйте этих клиентов в подчинённые серверы NIS и заставьте их привязываться к самим себе.

Запрет доступа некоторым пользователям

В этом примере система `basie` является рабочей станцией преподавателя в домене NIS. Файл passwd на главном сервере NIS содержит учётные записи как преподавателей, так и студентов. В этом разделе показано, как разрешить вход преподавателей в эту систему, запретив вход студентам.

Чтобы предотвратить вход определённых пользователей в систему, даже если они присутствуют в базе данных NIS, используйте `vipw` для добавления `-имя_пользователя` с

правильным количеством двоеточий в конце файла /etc/master.passwd на клиенте, где *имя_пользователя* — это имя пользователя, которому запрещен вход. Строка с заблокированным пользователем должна находиться перед строкой +, которая разрешает вход пользователям NIS. В этом примере пользователю bill запрещен вход на basie:

```
basie# cat /etc/master.passwd
root:[password]:0:0::0:0:The super-user:/root:/bin/csh
toor:[password]:0:0::0:0:The other super-user:/root:/bin/sh
daemon:*:1:1::0:0:Owner of many system processes:/root:/usr/sbin/nologin
operator:*:2:5::0:0:System &:/:/usr/sbin/nologin
bin:*:3:7::0:0:Binaries Commands and Source,,,:/:/usr/sbin/nologin
tty:*:4:65533::0:0:Tty Sandbox:/:/usr/sbin/nologin
kmem:*:5:65533::0:0:KMem Sandbox:/:/usr/sbin/nologin
games:*:7:13::0:0:Games pseudo-user:/usr/games:/usr/sbin/nologin
news:*:8:8::0:0:News Subsystem:/:/usr/sbin/nologin
man:*:9:9::0:0:Mister Man Pages:/usr/share/man:/usr/sbin/nologin
bind:*:53:53::0:0:Bind Sandbox:/:/usr/sbin/nologin
uucp:*:66:66::0:0:UUCP pseudo-user:/var/spool/uucppublic:/usr/libexec/uucp/uucico
xten:*:67:67::0:0:X-10 daemon:/usr/local/xten:/usr/sbin/nologin
pop:*:68:6::0:0:Post Office Owner:/nonexistent:/usr/sbin/nologin
nobody:*:65534:65534::0:0:Unprivileged user:/nonexistent:/usr/sbin/nologin
-bill:::::::::
+:::::::::

basie#
```

33.4.8. Использование сетевых групп

Запрет указанным пользователям возможности входа в отдельные системы становится неэффективным и не масштабируемым в крупных сетях и быстро лишает NIS основного преимущества: *централизованного* администрирования.

Сетевые группы были разработаны для управления большими и сложными сетями с сотнями пользователей и машин. Их использование аналогично группам в UNIX®, с той основной разницей, что отсутствует числовой идентификатор и есть возможность определять сетевую группу, включая как учётные записи пользователей, так и другие сетевые группы.

Для дополнения примера, используемого в этой главе, домен NIS будет увеличен за счет пользователей и систем, показанным в таблицах 28.2 и 28.3:

Таблица 48. Дополнительные пользователи

Имена пользователей	Описание
alpha, beta	Сотрудники IT-отдела
charlie, delta	Стажеры IT-отдела
echo, foxtrott, golf, ...	Сотрудники

Имена пользователей	Описание
able, baker, ...	Интерны

Таблица 49. Дополнительные Системы

Имена машин	Описание
war, death, famine, pollution	Только сотрудники IT имеют право входить на эти серверы.
pride, greed, envy, wrath, lust, sloth	Все сотрудники IT-отдела имеют право входить на эти серверы.
one, two, three, four, ...	Обычные рабочие станции, используемые сотрудниками.
trashcan	Очень старая машина без каких-либо важных данных. Даже интернам разрешено использовать эту систему.

При использовании сетевых групп для настройки этого сценария каждый пользователь назначается в одну или несколько сетевых групп, а затем вход разрешается или запрещается для всех членов сетевой группы. При добавлении новой машины необходимо определить ограничения входа для всех сетевых групп. Когда добавляется новый пользователь, его учётная запись должна быть добавлена в одну или несколько сетевых групп. Если настройка NIS выполнена тщательно, для предоставления или запрета доступа к машинам потребуется изменить только один центральный файл конфигурации.

Первым шагом является инициализация NIS `netgroup` карты. В FreeBSD эта карта не создаётся по умолчанию. На главном сервере NIS используйте редактор для создания карты с именем /var/yp/netgroup.

Этот пример создаёт четыре сетевые группы для представления сотрудников IT, стажеров IT, сотрудников и интернов:

```
IT_EMP  (,alpha,test-domain)    (,beta,test-domain)
IT_APP  (,charlie,test-domain)  (,delta,test-domain)
USERS   (,echo,test-domain)     (,foxtrott,test-domain) \
        (,golf,test-domain)
INTERNS (,able,test-domain)     (,baker,test-domain)
```

Каждая запись настраивает сетевую группу. Первый столбец в записи — это название сетевой группы. Каждый набор скобок представляет либо группу из одного или нескольких пользователей, либо имя другой сетевой группы. При указании пользователя три поля, разделённые запятыми, внутри каждой группы означают:

1. Имя хоста(ов), на котором другие поля, представляющие пользователя, действительны. Если имя хоста не указано, запись действительна на всех хостах.
2. Имя учётной записи, принадлежащей этой сетевой группе.
3. NIS-домен для учётной записи. Учётные записи могут быть импортированы из других

NIS-доменов в сетевую группу.

Если группа содержит нескольких пользователей, разделяйте каждого пользователя пробелом. Кроме того, каждое поле может содержать символы подстановки. Подробности см. в netgroup(5).

Имена сетевых групп длиннее 8 символов не должны использоваться. Имена чувствительны к регистру, и использование заглавных букв для имён сетевых групп — это простой способ отличить имена пользователей, машин и сетевых групп.

Некоторые клиенты NIS, не относящиеся к FreeBSD, не могут обрабатывать сетевые группы, содержащие более 15 записей. Это ограничение можно обойти, создав несколько подгрупп с 15 или менее пользователями и настоящую сетевую группу, состоящую из этих подгрупп, как показано в этом примере:

```
BIGGRP1  (,joe1,domain)  (,joe2,domain)  (,joe3,domain) [...]
BIGGRP2  (,joe16,domain)  (,joe17,domain) [...]
BIGGRP3  (,joe31,domain)  (,joe32,domain)
BIGGROUP  BIGGRP1 BIGGRP2 BIGGRP3
```

Повторите этот процесс, если в одной сетевой группе существует более 225 (15 умножить на 15) пользователей.

Для активации и распространения новой NIS-карты:

```
ellington# cd /var/yp
ellington# make
```

Это создаст три карты NIS netgroup, netgroup.byhost и netgroup.byuser. Используйте опцию ключа карты в ypcat(1), чтобы проверить доступность новых карт NIS:

```
ellington% ypcat -k netgroup
ellington% ypcat -k netgroup.byhost
ellington% ypcat -k netgroup.byuser
```

Вывод первой команды должен напоминать содержимое файла /var/yp/netgroup. Вторая команда выводит результат только в случае создания специфичных для хоста групп сетей. Третья команда используется для получения списка групп сетей для пользователя.

Для настройки клиента используйте vipw(8), чтобы указать имя сетевой группы. Например, на сервере с именем `war` замените эту строку:

```
+:::::::::
```

строкой

```
+@IT_EMP:::::::::
```

Указывает, что только пользователи, определённые в сетевой группе `IT_EMP`, будут импортированы в базу данных паролей этой системы, и только этим пользователям разрешён вход в систему.

Эта конфигурация также применяется к функции `~` оболочки и всем процедурам, которые преобразуют между именами пользователей и числовыми идентификаторами пользователей. Другими словами, `cd ~user` не будет работать, `ls -l` покажет числовой ID вместо имени пользователя, а `find . -user joe -print` завершится с сообщением `No such user`. Чтобы исправить это, импортируйте все записи пользователей, не разрешая им вход на серверы. Это можно достичь, добавив дополнительную строку:

```
+:::::::::/usr/sbin/nologin
```

Эта строка настраивает клиент на импорт всех записей, но с заменой оболочки в этих записях на /usr/sbin/nologin.

Убедитесь, что дополнительная строка добавлена *после* `+@IT_EMP:::::::::`. В противном случае у всех пользовательских учётных записей, импортированных из NIS, будет указана оболочка входа /usr/sbin/nologin, и никто не сможет войти в систему.

Для настройки менее важных серверов замените старые `+:::::::::` на серверах следующими строками:

```
+@IT_EMP:::::::::
+@IT_APP:::::::::
+:::::::::/usr/sbin/nologin
```

Соответствующие строки для рабочих станций будут:

```
+@IT_EMP:::::::::
+@USERS:::::::::
+:::::::::/usr/sbin/nologin
```

NIS поддерживает создание netgroups из других netgroups, что может быть полезно при изменении политики доступа пользователей. Одна из возможностей — создание ролевых netgroups. Например, можно создать netgroup с именем `BIGSRV` для определения ограничений входа на важные серверы, другую netgroup `SMALLSRV` для менее важных серверов и третью netgroup `USERBOX` для рабочих станций. Каждая из этих netgroups содержит netgroups, которым разрешено входить на эти машины. Новые записи для карты NIS `netgroup` будут выглядеть так:

```
BIGSRV    IT_EMP  IT_APP
SMALLSRV  IT_EMP  IT_APP  ITINTERN
```

```
USERBOX   IT_EMP  ITINTERN USERS
```

Этот метод определения ограничений входа работает достаточно хорошо, когда можно определить группы машин с одинаковыми ограничениями. К сожалению, это скорее исключение, чем правило. В большинстве случаев требуется возможность определять ограничения входа для каждой машины отдельно.

Определения машинозависимых сетевых групп — ещё один способ справиться с изменениями политики. В этом сценарии файл /etc/master.passwd на каждой системе содержит две строки, начинающиеся с "+". Первая строка добавляет сетевую группу с учётными записями, которым разрешён вход на эту машину, а вторая строка добавляет все остальные учётные записи с оболочкой /usr/sbin/nologin. Рекомендуется использовать имя сетевой группы в версии "ВСЕ-ЗАГЛАВНЫЕ", соответствующее имени хоста:

```
+@BOXNAME:::::::::
+::::::::::/usr/sbin/nologin
```

После выполнения этой задачи на всех машинах больше не требуется изменять локальные версии файла /etc/master.passwd. Все дальнейшие изменения можно выполнять, редактируя карту NIS. Вот пример возможной карты `netgroup` для данного сценария:

```
# Define groups of users first
IT_EMP    (,alpha,test-domain)    (,beta,test-domain)
IT_APP    (,charlie,test-domain)  (,delta,test-domain)
DEPT1     (,echo,test-domain)     (,foxtrott,test-domain)
DEPT2     (,golf,test-domain)     (,hotel,test-domain)
DEPT3     (,india,test-domain)    (,juliet,test-domain)
ITINTERN  (,kilo,test-domain)     (,lima,test-domain)
D_INTERNS (,able,test-domain)     (,baker,test-domain)
#
# Now, define some groups based on roles
USERS     DEPT1   DEPT2     DEPT3
BIGSRV    IT_EMP  IT_APP
SMALLSRV  IT_EMP  IT_APP    ITINTERN
USERBOX   IT_EMP  ITINTERN  USERS
#
# And a groups for a special tasks
# Allow echo and golf to access our anti-virus-machine
SECURITY  IT_EMP  (,echo,test-domain)  (,golf,test-domain)
#
# machine-based netgroups
# Our main servers
WAR       BIGSRV
FAMINE    BIGSRV
# User india needs access to this server
POLLUTION  BIGSRV  (,india,test-domain)
#
# This one is really important and needs more access restrictions
```

```
DEATH     IT_EMP
#
# The anti-virus-machine mentioned above
ONE       SECURITY
#
# Restrict a machine to a single user
TWO       (,hotel,test-domain)
# [...more groups to follow]
```

Не всегда целесообразно использовать сетевые группы, привязанные к машинам. При развертывании нескольких десятков или сотен систем можно использовать ролевые сетевые группы вместо машинных, чтобы размер карты NIS оставался в разумных пределах.

33.4.9. Форматы паролей

NIS требует, чтобы все хосты в домене NIS использовали одинаковый формат шифрования паролей. Если у пользователей возникают проблемы с аутентификацией на клиенте NIS, это может быть связано с разным форматом паролей. В гетерогенной сети формат должен поддерживаться всеми операционными системами, где DES является минимальным общим стандартом.

Чтобы проверить, какой формат использует сервер или клиент, посмотрите на этот раздел в /etc/login.conf:

```
default:\
    :passwd_format=des:\
    [Further entries elided]
```

В этом примере система использует формат DES для хеширования паролей. Другие возможные значения включают `blf` для Blowfish, `md5` для MD5, `sha256` и `sha512` для SHA-256 и SHA-512 соответственно. Для получения дополнительной информации и актуального списка доступных вариантов на вашей системе обратитесь к crypt(3).

Если формат на хосте необходимо изменить, чтобы он соответствовал формату, используемому в домене NIS, базу данных возможностей входа необходимо перестроить после сохранения изменений:

```
# cap_mkdb /etc/login.conf
```

Формат паролей для существующих учётных записей не будет обновлён, пока каждый пользователь не изменит свой пароль *после* перестроения базы данных возможностей входа.

33.5. Протокол LDAP

Протокол LDAP (Lightweight Directory Access Protocol) — это протокол уровня приложений,

используемый для доступа, изменения и аутентификации объектов с помощью распределённой службы каталогов. Его можно сравнить с телефонной книгой или архивом, который хранит несколько уровней иерархической однородной информации. Он применяется в сетях Active Directory и OpenLDAP, позволяя пользователям получать доступ к различным уровням внутренней информации с использованием одной учётной записи. Например, аутентификация электронной почты, получение контактных данных сотрудников и аутентификация на внутренних веб-сайтах могут осуществляться с помощью одной учётной записи в базе данных LDAP-сервера.

В этом разделе представлено краткое руководство по настройке сервера LDAP в системе FreeBSD. Предполагается, что администратор уже имеет продуманный план, включающий тип хранимой информации, её назначение, перечень пользователей с доступом к этой информации и способы защиты от несанкционированного доступа.

33.5.1. Терминология и структура LDAP

LDAP использует несколько терминов, которые следует понять перед началом настройки. Все записи каталога состоят из группы *атрибутов*. Каждый из этих наборов атрибутов содержит уникальный идентификатор, известный как *Отличительное имя* (DN — Distinguished Name), который обычно строится из нескольких других атрибутов, таких как общее имя или *Относительное отличительное имя* (RDN — Relative Distinguished Name). Подобно тому, как каталоги имеют абсолютные и относительные пути, можно рассматривать DN как абсолютный путь, а RDN — как относительный путь.

Пример записи LDAP выглядит следующим образом. В этом примере выполняется поиск записи для указанной учётной записи пользователя (`uid`), организационного подразделения (`ou`) и организации (`o`):

```
% ldapsearch -xb "uid=trhodes,ou=users,o=example.com"
# extended LDIF
#
# LDAPv3
# base <uid=trhodes,ou=users,o=example.com> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# trhodes, users, example.com
dn: uid=trhodes,ou=users,o=example.com
mail: trhodes@example.com
cn: Tom Rhodes
uid: trhodes
telephoneNumber: (123) 456-7890

# search result
search: 2
result: 0 Success

# numResponses: 2
```

```
# numEntries: 1
```

Этот пример записи показывает значения атрибутов `dn`, `mail`, `cn`, `uid` и `telephoneNumber`. Атрибут `cn` является RDN.

Дополнительная информация о LDAP и его терминологии доступна по адресу http://www.openldap.org/doc/admin24/intro.html.

33.5.2. Настройка сервера LDAP

FreeBSD не предоставляет встроенный LDAP-сервер. Начните настройку с установки пакета net/openldap-server или порта:

```
# pkg install openldap-server
```

В пакете включен большой набор параметров по умолчанию. Их можно просмотреть, выполнив команду `pkg info openldap-server`. Если их недостаточно (например, требуется поддержка SQL), рекомендуется перекомпилировать порт с использованием соответствующего фреймворка.

Установка создаёт каталог /var/db/openldap-data для хранения данных. Необходимо создать каталог для хранения сертификатов:

```
# mkdir /usr/local/etc/openldap/private
```

Следующий этап — настройка Центра Сертификации. Следующие команды должны быть выполнены из каталога /usr/local/etc/openldap/private. Это важно, так как права доступа к файлам должны быть строгими, и пользователи не должны иметь доступ к этим файлам. Более подробную информацию о сертификатах и их параметрах можно найти в OpenSSL. Чтобы создать Центр Сертификации, начните с этой команды и следуйте инструкциям:

```
# openssl req -days 365 -nodes -new -x509 -keyout ca.key -out ../ca.crt
```

Записи для запросов могут быть любыми, *за исключением* `Common Name`. Эта запись должна *отличаться* от имени хоста системы. Если это будет самоподписанный сертификат, добавьте к имени хоста префикс `CA` — как Центр Сертификации.

Следующая задача — создать запрос на подпись сертификата и закрытый ключ. Введите эту команду и следуйте инструкциям:

```
# openssl req -days 365 -nodes -new -keyout server.key -out server.csr
```

В процессе генерации сертификата обязательно правильно укажите атрибут `Common Name`. Запрос на подпись сертификата (Certificate Signing Request) должен быть подписан Центром сертификации, чтобы использоваться в качестве действительного сертификата:

```
# openssl x509 -req -days 365 -in server.csr -out ../server.crt -CA ../ca.crt -CAkey
ca.key -CAcreateserial
```

Заключительная часть процесса генерации сертификатов — создание и подписание клиентских сертификатов:

```
# openssl req -days 365 -nodes -new -keyout client.key -out client.csr
# openssl x509 -req -days 3650 -in client.csr -out ../client.crt -CA ../ca.crt -CAkey
ca.key
```

Помните, что нужно использовать тот же атрибут `Common Name` при запросе. По завершении убедитесь, что в результате выполнения команд было создано в общей сложности восемь (8) новых файлов.

Демон, запускающий сервер OpenLDAP, называется slapd. Его настройка выполняется через файл slapd.ldif: старый файл slapd.conf больше не используется в OpenLDAP.

Есть примеры конфигурации для slapd.ldif доступны, и также их можно найти в /usr/local/etc/openldap/slapd.ldif.sample. Документация параметров в slapd-config(5). Каждый раздел slapd.ldif, как и все другие наборы атрибутов LDAP, однозначно идентифицируется через DN. Убедитесь, что между строкой `dn:` и желаемым концом раздела нет пустых строк. В следующем примере TLS будет использоваться для настройки безопасного канала. Первый раздел представляет глобальную конфигурацию:

```
#
# See slapd-config(5) for details on configuration options.
# This file should NOT be world readable.
#
dn: cn=config
objectClass: olcGlobal
cn: config
#
#
# Define global ACLs to disable default read access.
#
olcArgsFile: /var/run/openldap/slapd.args
olcPidFile: /var/run/openldap/slapd.pid
olcTLSCertificateFile: /usr/local/etc/openldap/server.crt
olcTLSCertificateKeyFile: /usr/local/etc/openldap/private/server.key
olcTLSCACertificateFile: /usr/local/etc/openldap/ca.crt
#olcTLSCipherSuite: HIGH
olcTLSProtocolMin: 3.1
olcTLSVerifyClient: never
```

Здесь необходимо указать файлы Центра сертификации, сертификата сервера и закрытого ключа сервера. Рекомендуется позволить клиентам выбирать алгоритм шифрования и опустить опцию `olcTLSCipherSuite` (несовместимо с TLS-клиентами, кроме openssl). Опция

`olcTLSProtocolMin` позволяет серверу требовать минимальный уровень безопасности: это рекомендуется. Хотя проверка обязательна для сервера, для клиента она не требуется: `olcTLSVerifyClient: never`.

Второй раздел посвящен серверным модулям и может быть настроен следующим образом:

```
#
# Load dynamic backend modules:
#
dn: cn=module,cn=config
objectClass: olcModuleList
cn: module
olcModulepath:  /usr/local/libexec/openldap
olcModuleload:  back_mdb.la
#olcModuleload: back_bdb.la
#olcModuleload: back_hdb.la
#olcModuleload: back_ldap.la
#olcModuleload: back_passwd.la
#olcModuleload: back_shell.la
```

Третий раздел посвящён загрузке необходимых схем `ldif` для использования базами данных: они являются важными.

```
dn: cn=schema,cn=config
objectClass: olcSchemaConfig
cn: schema

include: file:///usr/local/etc/openldap/schema/core.ldif
include: file:///usr/local/etc/openldap/schema/cosine.ldif
include: file:///usr/local/etc/openldap/schema/inetorgperson.ldif
include: file:///usr/local/etc/openldap/schema/nis.ldif
```

Далее, раздел конфигурации фронтенда (уровня взаимодействия с клиентами):

```
# Frontend settings
#
dn: olcDatabase={-1}frontend,cn=config
objectClass: olcDatabaseConfig
objectClass: olcFrontendConfig
olcDatabase: {-1}frontend
olcAccess: to * by * read
#
# Sample global access control policy:
#   Root DSE: allow anyone to read it
#   Subschema (sub)entry DSE: allow anyone to read it
#   Other DSEs:
#       Allow self write access
#       Allow authenticated users read access
```

```
#       Allow anonymous users to authenticate
#
#olcAccess: to dn.base="" by * read
#olcAccess: to dn.base="cn=Subschema" by * read
#olcAccess: to *
#   by self write
#   by users read
#   by anonymous auth
#
# if no access controls are present, the default policy
# allows anyone and everyone to read anything but restricts
# updates to rootdn.  (e.g., "access to * by * read")
#
# rootdn can always read and write EVERYTHING!
#
olcPasswordHash: {SSHA}
# {SSHA} is already the default for olcPasswordHash
```

Еще один раздел посвящен *бэкенду конфигурации* — единственному способу последующего доступа к конфигурации сервера OpenLDAP, который доступен только глобальному суперпользователю.

```
dn: olcDatabase={0}config,cn=config
objectClass: olcDatabaseConfig
olcDatabase: {0}config
olcAccess: to * by * none
olcRootPW: {SSHA}iae+lrQZILpiUdf16Z9KmDmSwT77Dj4U
```

Имя администратора по умолчанию — `cn=config`. Введите slappasswd в оболочке, выберите пароль и используйте его хеш в `olcRootPW`. Если этот параметр не указан сейчас, до импорта slapd.ldif, никто не сможет впоследствии изменить раздел *глобальной конфигурации*.

Последний раздел посвящен бэкенду базы данных (уровню хранения данных):

```
#######################################################################
# LMDB database definitions
#######################################################################
#
dn: olcDatabase=mdb,cn=config
objectClass: olcDatabaseConfig
objectClass: olcMdbConfig
olcDatabase: mdb
olcDbMaxSize: 1073741824
olcSuffix: dc=domain,dc=example
olcRootDN: cn=mdbadmin,dc=domain,dc=example
# Cleartext passwords, especially for the rootdn, should
# be avoided.  See slappasswd(8) and slapd-config(5) for details.
# Use of strong authentication encouraged.
```

```
olcRootPW: {SSHA}X2wHvIWDk6G76CQyCMS1vDCvtICWgn0+
# The database directory MUST exist prior to running slapd AND
# should only be accessible by the slapd and slap tools.
# Mode 700 recommended.
olcDbDirectory: /var/db/openldap-data
# Indices to maintain
olcDbIndex: objectClass eq
```

Эта база данных содержит *фактическое содержимое* каталога LDAP. Доступны типы, отличные от `mdb`. Суперпользователь (не путать с глобальным) настраивается здесь: (возможно, пользовательское) имя пользователя в `olcRootDN` и хэш пароля в `olcRootPW`; slappasswd можно использовать, как и раньше.

Этот репозиторий содержит четыре примера файла slapd.ldif. Для преобразования существующего slapd.conf в slapd.ldif обратитесь к этой странице (обратите внимание, что это может добавить некоторые бесполезные опции).

После завершения настройки файл slapd.ldif должен быть скопирован в пустой каталог. Рекомендуется создать его следующим образом:

```
# mkdir /usr/local/etc/openldap/slapd.d/
```

Импорт базы данных конфигурации:

```
# /usr/local/sbin/slapadd -n0 -F /usr/local/etc/openldap/slapd.d/ -l
/usr/local/etc/openldap/slapd.ldif
```

Запустите демон slapd:

```
# /usr/local/libexec/slapd -F /usr/local/etc/openldap/slapd.d/
```

Опция `-d` может использоваться для отладки, как указано в slapd(8). Чтобы проверить, что сервер запущен и работает:

```
# ldapsearch -x -b '' -s base '(objectclass=*)' namingContexts
# extended LDIF
#
# LDAPv3
# base <> with scope baseObject
# filter: (objectclass=*)
# requesting: namingContexts
#

#
dn:
namingContexts: dc=domain,dc=example
```

```
# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

Сервер по-прежнему должен быть доверенным. Если это никогда не делалось ранее, следуйте этим инструкциям. Установите пакет или порт OpenSSL:

```
# pkg install openssl
```

Из каталога, где находится ca.crt (в данном примере, /usr/local/etc/openldap), выполните:

```
# c_rehash .
```

И сертификат центра сертификации, и сертификат сервера теперь правильно распознаются в своих соответствующих ролях. Чтобы проверить это, выполните следующую команду из каталога, где находится server.crt:

```
# openssl verify -verbose -CApath . server.crt
```

Если slapd был запущен, перезапустите его. Как указано в /usr/local/etc/rc.d/slapd, для корректного запуска slapd при загрузке следующие строки должны быть добавлены в /etc/rc.conf:

```
slapd_enable="YES"
slapd_flags='-h "ldapi://%2fvar%2frun%2fopenldap%2fldapi/ ldap://0.0.0.0/"'
slapd_sockets="/var/run/openldap/ldapi"
slapd_cn_config="YES"
```

slapd не предоставляет отладку при загрузке. Для этой цели проверьте /var/log/debug.log, dmesg -a и /var/log/messages.

Следующий пример добавляет группу `team` и пользователя `john` в базу данных LDAP `domain.example`, которая пока пуста. Сначала создайте файл domain.ldif:

```
# cat domain.ldif
dn: dc=domain,dc=example
objectClass: dcObject
objectClass: organization
o: domain.example
dc: domain
```

```
dn: ou=groups,dc=domain,dc=example
objectClass: top
objectClass: organizationalunit
ou: groups

dn: ou=users,dc=domain,dc=example
objectClass: top
objectClass: organizationalunit
ou: users

dn: cn=team,ou=groups,dc=domain,dc=example
objectClass: top
objectClass: posixGroup
cn: team
gidNumber: 10001

dn: uid=john,ou=users,dc=domain,dc=example
objectClass: top
objectClass: account
objectClass: posixAccount
objectClass: shadowAccount
cn: John McUser
uid: john
uidNumber: 10001
gidNumber: 10001
homeDirectory: /home/john/
loginShell: /usr/bin/bash
userPassword: secret
```

См. документацию OpenLDAP для получения более подробной информации. Используйте slappasswd для замены пароля в открытом тексте `secret` на хеш в `userPassword`. Путь, указанный как `loginShell`, должен существовать во всех системах, где `john` имеет право входить. Наконец, используйте администратора `mdb` для изменения базы данных:

```
# ldapadd -W -D "cn=mdbadmin,dc=domain,dc=example" -f domain.ldif
```

Изменения в разделе *глобальной конфигурации* могут выполняться только глобальным суперпользователем. Например, предположим, что изначально была указана опция `olcTLSCipherSuite: HIGH:MEDIUM:SSLv3`, которую теперь необходимо удалить. Сначала создайте файл, содержащий следующее:

```
# cat global_mod
dn: cn=config
changetype: modify
delete: olcTLSCipherSuite
```

Затем примените изменения:

```
# ldapmodify -f global_mod -x -D "cn=config" -W
```

При запросе введите пароль, выбранный в разделе *бэкенда конфигурации*. Имя пользователя не требуется: здесь `cn=config` представляет DN раздела базы данных, который нужно изменить. Или используйте `ldapmodify` для удаления отдельной строки базы данных или `ldapdelete` для удаления всей записи.

Если что-то пойдет не так или если глобальный суперпользователь не сможет получить доступ к бэкенду конфигурации, можно удалить и перезаписать всю конфигурацию:

```
# rm -rf /usr/local/etc/openldap/slapd.d/
```

slapd.ldif затем можно отредактировать и снова импортировать. Пожалуйста, следуйте этой процедуре только в том случае, если нет другого доступного решения.

Это конфигурация только сервера. На той же машине также может быть размещен LDAP-клиент с собственной отдельной конфигурацией.

33.6. Протокол динамической конфигурации узла (DHCP)

Протокол динамической конфигурации узла (DHCP —Dynamic Host Configuration Protocol) позволяет системе подключаться к сети для получения необходимой адресной информации для общения в этой сети. FreeBSD включает версию `dhclient` от OpenBSD, которая используется клиентом для получения адресной информации. FreeBSD не устанавливает сервер DHCP, но несколько серверов доступны в коллекции портов FreeBSD. Протокол DHCP полностью описан в RFC 2131. Информационные ресурсы также доступны на isc.org/downloads/dhcp/.

Этот раздел описывает, как использовать встроенный DHCP-клиент. Затем он описывает, как установить и настроить DHCP-сервер.

В FreeBSD устройство bpf(4) необходимо как для сервера DHCP, так и для клиента DHCP. Это устройство включено в ядро GENERIC, которое устанавливается с FreeBSD. Пользователям, предпочитающим создавать собственное ядро, необходимо оставить это устройство, если используется DHCP.

Следует отметить, что bpf также позволяет привилегированным пользователям запускать анализаторы сетевых пакетов в этой системе.

33.6.1. Настройка клиента DHCP

Поддержка DHCP-клиента включена в установщик FreeBSD, что позволяет легко настроить новую систему для автоматического получения сетевой адресации от существующего DHCP-сервера. Примеры настройки сети можно найти в разделе Учётные записи, Часовая зона,

Службы и Защита.

Когда `dhclient` выполняется на клиентской машине, он начинает транслировать запросы на получение конфигурационной информации. По умолчанию эти запросы используют UDP-порт 68. Сервер отвечает на UDP-порту 67, предоставляя клиенту IP-адрес и другую соответствующую сетевую информацию, такую как маска подсети, шлюз по умолчанию и адреса DNS-серверов. Эта информация предоставляется в форме "аренды" DHCP и действительна в течение настраиваемого времени. Это позволяет автоматически повторно использовать устаревшие IP-адреса для клиентов, которые больше не подключены к сети. Клиенты DHCP могут получить от сервера большое количество информации. Полный список можно найти в dhcp-options(5).

По умолчанию, при загрузке системы FreeBSD её DHCP-клиент работает в фоновом режиме или *асинхронно*. Другие скрипты запуска продолжают выполняться, пока завершается процесс DHCP, что ускоряет загрузку системы.

DHCP в фоновом режиме работает хорошо, когда сервер DHCP быстро отвечает на запросы клиента. Однако на некоторых системах выполнение DHCP может занять много времени. Если сетевые службы пытаются запуститься до того, как DHCP назначит информацию о сетевой адресации, они завершатся с ошибкой. Использование DHCP в *синхронном* режиме предотвращает эту проблему, приостанавливая запуск до завершения настройки DHCP.

Эта строка в /etc/rc.conf используется для настройки фонового или асинхронного режима:

```
ifconfig_fxp0="DHCP"
```

Эта строка может уже существовать, если система была настроена на использование DHCP во время установки. Замените `fxp0`, указанный в этих примерах, на имя интерфейса, который нужно настроить динамически, как описано в Настройка сетевых интерфейсов.

Для настройки системы на использование синхронного режима с приостановкой во время запуска до завершения DHCP используйте “SYNCDHCP”:

```
ifconfig_fxp0="SYNCDHCP"
```

Есть ещё несколько опций клиента. Подробности смотрите в rc.conf(5), выполнив поиск по `dhclient`.

Клиент DHCP использует следующие файлы:

- /etc/dhclient.conf

  Файл конфигурации, используемый `dhclient`. Обычно этот файл содержит только комментарии, так как значения по умолчанию подходят для большинства клиентов. Этот конфигурационный файл описан в dhclient.conf(5).

- /sbin/dhclient

Дополнительную информацию о самой команде можно найти в dhclient(8).

- /sbin/dhclient-script

  Специфичный для FreeBSD скрипт конфигурации DHCP-клиента. Он описан в dhclient-script(8), но для правильной работы не требует изменений со стороны пользователя.

- /var/db/dhclient.leases.interface

  Клиент DHCP сохраняет базу данных действительных аренд в этом файле, который записывается как журнал и описывается в dhclient.leases(5).

33.6.2. Установка и настройка сервера DHCP

В этом разделе показано, как настроить систему FreeBSD в качестве DHCP-сервера с использованием реализации DHCP-сервера от Консорциума Интернет-систем (ISC —Internet Systems Consortium). Эту реализацию и её документацию можно установить с помощью пакета net/isc-dhcp44-server или порта.

Установка пакета net/isc-dhcp44-server включает образец файла конфигурации. Скопируйте /usr/local/etc/dhcpd.conf.example в /usr/local/etc/dhcpd.conf и внесите необходимые изменения в этот новый файл.

Файл конфигурации состоит из объявлений для подсетей и хостов, которые определяют информацию, предоставляемую клиентам DHCP. Например, следующие строки настраивают следующее:

```
option domain-name "example.org";①
option domain-name-servers ns1.example.org;②
option subnet-mask 255.255.255.0;③

default-lease-time 600;④
max-lease-time 72400;⑤
ddns-update-style none;⑥

subnet 10.254.239.0 netmask 255.255.255.224 {
  range 10.254.239.10 10.254.239.20;⑦
  option routers rtr-239-0-1.example.org, rtr-239-0-2.example.org;⑧
}

host fantasia {
  hardware ethernet 08:00:07:26:c0:a5;⑨
  fixed-address fantasia.fugue.com;⑩
}
```

① Этот параметр задаёт домен поиска по умолчанию, который будет предоставляться клиентам. Дополнительную информацию можно найти в resolv.conf(5).

② Эта опция определяет разделённый запятыми список DNS-серверов, которые должен использовать клиент. Они могут быть указаны по их Полным Доменным Именам (FQDN),

как показано в примере, или по их IP-адресам.

③ Маска подсети, которая будет предоставлена клиентам.

④ Время истечения аренды по умолчанию в секундах. Клиент может быть настроен для переопределения этого значения.

⑤ Максимально допустимая продолжительность аренды в секундах. Если клиент запросит аренду на более длительный срок, аренда всё равно будет выдана, но будет действительна только в течение `max-lease-time`.

⑥ Значение по умолчанию `none` отключает динамические обновления DNS. Изменение этого параметра на `interim` настраивает DHCP-сервер на обновление DNS-сервера при каждой выдаче аренды, чтобы DNS-сервер знал, какие IP-адреса связаны с какими компьютерами в сети. Не изменяйте значение по умолчанию, если DNS-сервер не настроен для поддержки динамического DNS.

⑦ Эта строка создаёт пул доступных IP-адресов, зарезервированных для выделения клиентам DHCP. Диапазон адресов должен быть действительным для сети или подсети, указанной в предыдущей строке.

⑧ Объявляет шлюз по умолчанию, действительный для сети или подсети, указанной перед открывающей скобкой `{`.

⑨ Указывает аппаратный MAC-адрес клиента, чтобы DHCP-сервер мог распознать клиента при его запросе.

⑩ Указывает, что данный узел всегда должен получать один и тот же IP-адрес. Использование имени узла корректно, так как DHCP-сервер разрешит имя узла перед возвратом информации об аренде.

Этот файл конфигурации поддерживает гораздо больше опций. Подробности и примеры смотрите в dhcpd.conf(5), который устанавливается вместе с сервером.

После завершения настройки dhcpd.conf включите DHCP-сервер в /etc/rc.conf:

```
dhcpd_enable="YES"
dhcpd_ifaces="dc0"
```

Замените `dc0` на интерфейс (или интерфейсы, разделённые пробелами), на котором DHCP-сервер должен ожидать запросы от DHCP-клиентов.

Запустите сервер, выполнив следующую команду:

```
# service isc-dhcpd start
```

Любые последующие изменения конфигурации сервера потребуют остановки и последующего запуска службы dhcpd с помощью service(8).

Сервер DHCP использует следующие файлы. Обратите внимание, что страницы руководства устанавливаются вместе с серверным программным обеспечением.

- /usr/local/sbin/dhcpd

  Дополнительную информацию о сервере dhcpd можно найти в dhcpd(8).

- /usr/local/etc/dhcpd.conf

  Файл конфигурации сервера должен содержать всю информацию, которую необходимо предоставить клиентам, а также сведения о работе сервера. Этот конфигурационный файл описан в dhcpd.conf(5).

- /var/db/dhcpd.leases

  Сервер DHCP ведёт базу данных выданных аренд в этом файле, который записывается в виде журнала. Обратитесь к dhcpd.leases(5), где приводится несколько более подробное описание.

- /usr/local/sbin/dhcrelay

  Этот демон используется в сложных средах, где один DHCP-сервер перенаправляет запрос от клиента другому DHCP-серверу в отдельной сети. Если требуется такая функциональность, установите пакет net/isc-dhcp44-relay или соответствующий порт. Установка включает dhcrelay(8), где приведены более подробные сведения.

33.7. Система доменных имён (DNS)

Система доменных имён (DNS) — это протокол, который сопоставляет доменные имена с IP-адресами и наоборот. DNS координируется в масштабах Интернета через довольно сложную систему авторитетных корневых серверов, серверов доменов верхнего уровня (TLD — Top Level Domain) и других менее масштабных серверов имен, которые хранят и кэшируют информацию об отдельных доменах. Для выполнения DNS-запросов в системе не обязательно запускать сервер имен.

Следующая таблица описывает некоторые термины, связанные с DNS:

Таблица 50. Терминология DNS

Термин	Определение
Прямая запись DNS (Forward DNS)	Сопоставление имён хостов с IP-адресами.
Зона ответственности (Origin)	Относится к домену, охватываемому определённым файлом зоны.
Резолвер (Resolver)	Системный процесс, с помощью которого машина запрашивает у сервера имён информацию о зоне.
Обратная запись DNS (Reverse DNS)	Сопоставление IP-адресов с именами хостов.
Корневая зона (Root zone)	Начало иерархии зон Интернета. Все зоны находятся под корневой зоной, аналогично тому, как все файлы в файловой системе находятся под корневым каталогом.

Термин	Определение
Зона (Zone)	Отдельный домен, поддомен или часть DNS, управляемые одной организацией.

Примеры зон:

- `.` — так обычно обозначается корневая зона в документации.
- `org.` — это домен верхнего уровня (TLD) в корневой зоне.
- `example.org.` — это зона под доменом `org.`.
- `1.168.192.in-addr.arpa` — это зона, содержащая ссылки на все IP-адреса, входящие в адресное пространство `192.168.1.*`.

Как видно, более специфичная часть имени хоста расположена слева. Например, `example.org.` более специфично, чем `org.`, а `org.` более специфично, чем корневая зона. Структура каждой части имени хоста во многом напоминает файловую систему: каталог /dev находится в корне и так далее.

33.7.1. Причины для запуска сервера имен

Серверы имён обычно бывают двух видов: авторитетные DNS-серверы и кэширующие DNS-серверы (также известные как резолвинг-серверы).

Авторитетный сервер имён необходим, когда:

- Хочется предоставлять DNS-информацию для всего мира, отвечая на запросы авторитетно.
- Домен, например `example.org`, зарегистрирован, и IP-адреса должны быть назначены именам хостов в нём.
- Блок IP-адресов требует обратных DNS-записей (IP к имени хоста).
- Резервный или вторичный сервер имен, называемый подчиненным (slave), будет отвечать на запросы.

Кэширующий сервер имён необходим, когда:

- Локальный DNS-сервер может кэшировать и отвечать быстрее, чем запрос к внешнему серверу имен.

Когда кто-нибудь запрашивает информацию о `www.FreeBSD.org`, резолвер обычно обращается к серверу имён провайдера и получает ответ. При использовании локального кэширующего DNS-сервера запрос во внешний мир выполняется только один раз этим сервером. Дополнительные запросы не будут выходить за пределы локальной сети, так как информация сохраняется в локальном кэше.

33.7.2. Конфигурация DNS-сервера

Unbound включён в базовую систему FreeBSD. По умолчанию он предоставляет разрешение DNS только для локальной машины. Хотя пакет базовой системы можно настроить для

предоставления служб разрешения за пределами локальной машины, рекомендуется удовлетворять такие требования путём установки Unbound из коллекции портов FreeBSD.

Чтобы включить Unbound, добавьте следующую строку в /etc/rc.conf:

```
local_unbound_enable="YES"
```

Любые существующие серверы имён в /etc/resolv.conf будут настроены как серверы пересылки в новой конфигурации Unbound.

Если какой-либо из перечисленных серверов имён не поддерживает DNSSEC, локальное разрешение DNS завершится неудачей. Обязательно протестируйте каждый сервер имён и удалите те, которые не прошли проверку. Следующая команда покажет дерево доверия или ошибку для сервера имен, работающего на 192.168.1.1:

```
% drill -S FreeBSD.org @192.168.1.1
```

После подтверждения поддержки DNSSEC каждым сервером имен, запустите Unbound:

```
# service local_unbound onestart
```

Это обеспечит обновление /etc/resolv.conf, чтобы запросы к доменам, защищённым DNSSEC, теперь работали. Например, выполните следующую команду для проверки дерева доверия DNSSEC FreeBSD.org:

```
% drill -S FreeBSD.org
;; Number of trusted keys: 1
;; Chasing: freebsd.org. A

DNSSEC Trust tree:
freebsd.org. (A)
|---freebsd.org. (DNSKEY keytag: 36786 alg: 8 flags: 256)
    |---freebsd.org. (DNSKEY keytag: 32659 alg: 8 flags: 257)
    |---freebsd.org. (DS keytag: 32659 digest type: 2)
        |---org. (DNSKEY keytag: 49587 alg: 7 flags: 256)
            |---org. (DNSKEY keytag: 9795 alg: 7 flags: 257)
            |---org. (DNSKEY keytag: 21366 alg: 7 flags: 257)
            |---org. (DS keytag: 21366 digest type: 1)
            |   |---. (DNSKEY keytag: 40926 alg: 8 flags: 256)
            |       |---. (DNSKEY keytag: 19036 alg: 8 flags: 257)
            |---org. (DS keytag: 21366 digest type: 2)
                |---. (DNSKEY keytag: 40926 alg: 8 flags: 256)
                    |---. (DNSKEY keytag: 19036 alg: 8 flags: 257)
;; Chase successful
```

33.7.3. Конфигурация авторитетного сервера имен

FreeBSD не предоставляет программное обеспечение авторитетного сервера имён в базовой системе. Пользователям рекомендуется устанавливать сторонние приложения, такие как dns/nsd или dns/bind918 из пакетов или портов.

33.8. Сетевое взаимодействие без настройки (mDNS/DNS-SD)

Сетевое взаимодействие без настройки (иногда называемое *Zeroconf*) — это набор технологий, упрощающих настройку сети. Главные составные части Zeroconf:

- Линк-локальная адресация, предоставляющая автоматическое назначение числовых сетевых адресов.
- Мультикаст DNS (*mDNS*), обеспечивающий автоматическое распространение и разрешение имён хостов.
- DNS-Based Service Discovery (*DNS-SD*), обеспечивающий автоматическое обнаружение экземпляров сервисов.

33.8.1. Настройка и запуск Avahi

Одной из популярных реализаций zeroconf является Avahi. Avahi можно установить и настроить с помощью следующих команд:

```
# pkg install avahi-app nss_mdns
# grep -q '^hosts:.*\<mdns\>' /etc/nsswitch.conf || sed -i "" 's/^hosts: .*/& mdns/'
/etc/nsswitch.conf
# service dbus enable
# service avahi-daemon enable
# service dbus start
# service avahi-daemon start
```

33.9. HTTP сервер Apache

Веб-сервер с открытым исходным кодом Apache является наиболее широко используемым веб-сервером. FreeBSD не устанавливает этот веб-сервер по умолчанию, но его можно установить из пакета www/apache24 или порта.

В этом разделе приведена сводка по настройке и запуску версии 2.*x* HTTP-сервера Apache на FreeBSD. Более подробная информация о Apache 2.X и его директивах конфигурации доступна по ссылке httpd.apache.org.

33.9.1. Настройка и запуск Apache

В FreeBSD основной файл конфигурации сервера Apache HTTP Server устанавливается как /usr/local/etc/apache2x/httpd.conf, где *x* обозначает номер версии. Этот текстовый файл в

формате ASCII начинается со строк комментариев, обозначенных символом #. Наиболее часто изменяемые директивы:

`ServerRoot "/usr/local"`

Указывает иерархию каталогов по умолчанию для установки Apache. Исполняемые файлы хранятся в подкаталогах bin и sbin корня сервера, а конфигурационные файлы — в подкаталоге etc/apache2x.

`ServerAdmin you@example.com`

Замените это на адрес электронной почты для получения сообщений о проблемах с сервером. Этот адрес также появляется на некоторых страницах, сгенерированных сервером, таких как документы об ошибках.

`ServerName www.example.com:80`

Позволяет администратору установить имя хоста, которое отправляется клиентам сервера. Например, можно использовать `www` вместо фактического имени хоста. Если у системы нет зарегистрированного DNS-имени, введите её IP-адрес. Если сервер будет прослушивать альтернативный порт, замените `80` на номер альтернативного порта.

`DocumentRoot "/usr/local/www/apache2_x_/data"`

Каталог, из которого будут обслуживаться документы. По умолчанию все запросы обрабатываются из этого каталога, но символические ссылки и псевдонимы могут использоваться для указания на другие расположения.

Всегда рекомендуется создать резервную копию конфигурационного файла Apache по умолчанию перед внесением изменений. После завершения настройки Apache сохраните файл и проверьте конфигурацию с помощью `apachectl`. Запуск команды `apachectl configtest` должен вернуть `Syntax OK`.

Чтобы запускать Apache при загрузке системы, добавьте следующую строку в /etc/rc.conf:

```
apache24_enable="YES"
```

Если Apache должен запускаться с нестандартными параметрами, следующую строку можно добавить в /etc/rc.conf для указания необходимых флагов:

```
apache24_flags=""
```

Если apachectl не сообщает об ошибках конфигурации, то запустите `httpd`:

```
# service apache24 start
```

Службу `httpd` можно проверить, введя `http://localhost` в веб-браузере, заменив *localhost* на полное доменное имя машины, на которой работает `httpd`. Отображаемая веб-страница по умолчанию находится в /usr/local/www/apache24/data/index.html.

Проверить конфигурацию Apache на наличие ошибок после внесения последующих изменений в конфигурацию во время работы `httpd` можно с помощью следующей команды:

```
# service apache24 configtest
```

Важно отметить, что `configtest` не является стандартом rc(8), и не следует ожидать, что он будет работать для всех стартовых скриптов.

33.9.2. Виртуальный хостинг

Виртуальный хостинг позволяет запускать несколько веб-сайтов на одном сервере Apache. Виртуальные хосты могут быть *IP-ориентированными* или *имя-ориентированными*. IP-ориентированный виртуальный хостинг использует разные IP-адреса для каждого сайта. Имя-ориентированный виртуальный хостинг использует заголовки HTTP/1.1 клиента для определения имени хоста, что позволяет сайтам использовать один и тот же IP-адрес.

Для настройки Apache с использованием виртуального хоста на основе имён добавьте блок `VirtualHost` для каждого веб-сайта. Например, для веб-сервера с именем `www.domain.tld` и виртуальным доменом `www.someotherdomain.tld` добавьте следующие записи в httpd.conf:

```
<VirtualHost *>
    ServerName www.domain.tld
    DocumentRoot /www/domain.tld
</VirtualHost>

<VirtualHost *>
    ServerName www.someotherdomain.tld
    DocumentRoot /www/someotherdomain.tld
</VirtualHost>
```

Для каждого виртуального хоста замените значения `ServerName` и `DocumentRoot` на те, которые должны использоваться.

Для получения дополнительной информации о настройке виртуальных хостов обратитесь к официальной документации Apache по адресу: http://httpd.apache.org/docs/vhosts/.

33.9.3. Модули Apache

Apache использует модули для расширения функциональности, предоставляемой базовым сервером. Обратитесь к http://httpd.apache.org/docs/current/mod/ для получения полного списка и деталей настройки доступных модулей.

В FreeBSD некоторые модули могут быть скомпилированы с портом www/apache24. Введите `make config` в /usr/ports/www/apache24, чтобы увидеть, какие модули доступны и какие включены по умолчанию. Если модуль не скомпилирован с портом, коллекция портов FreeBSD предоставляет простой способ установки многих модулей. В этом разделе описаны три наиболее часто используемых модуля.

Поддержка SSL

В прошлом поддержка SSL в Apache требовала дополнительного модуля под названием mod_ssl. Сейчас это не так, и стандартная установка Apache включает SSL в веб-сервер. Пример настройки поддержки SSL-сайтов доступен в установленном файле httpd-ssl.conf внутри каталога /usr/local/etc/apache24/extra. В этом же каталоге также находится пример файла с именем ssl.conf-sample. Рекомендуется изучить оба файла для правильной настройки защищённых сайтов в веб-сервере Apache.

После настройки SSL необходимо раскомментировать следующую строку в основном файле http.conf, чтобы активировать изменения при следующей перезагрузке или обновлении Apache:

```
#Include etc/apache24/extra/httpd-ssl.conf
```

Версии SSL два и три имеют известные уязвимости. Настоятельно рекомендуется использовать версии TLS 1.2 и 1.3 вместо старых вариантов SSL. Это можно сделать, установив следующие параметры в ssl.conf:

```
SSLProtocol all -SSLv3 -SSLv2 +TLSv1.2 +TLSv1.3
SSLProxyProtocol all -SSLv2 -SSLv3 -TLSv1 -TLSv1.1
```

Для завершения настройки SSL в веб-сервере раскомментируйте следующую строку, чтобы убедиться, что конфигурация будет загружена в Apache при перезапуске или обновлении:

```
# Secure (SSL/TLS) connections
Include etc/apache24/extra/httpd-ssl.conf
```

Следующие строки также должны быть раскомментированы в файле httpd.conf для полной поддержки SSL в Apache:

```
LoadModule authn_socache_module libexec/apache24/mod_authn_socache.so
LoadModule socache_shmcb_module libexec/apache24/mod_socache_shmcb.so
LoadModule ssl_module libexec/apache24/mod_ssl.so
```

Следующий шаг — работа с центром сертификации для установки соответствующих сертификатов в системе. Это создаст цепочку доверия для сайта и предотвратит появление предупреждений о самоподписанных сертификатах.

mod_perl

Модуль mod_perl позволяет писать модули Apache на Perl. Кроме того, встроенный в сервер постоянный интерпретатор избегает накладных расходов на запуск внешнего интерпретатора и задержек при старте Perl.

mod_perl можно установить с помощью пакета www/mod_perl2 или порта. Документация по использованию этого модуля доступна по адресу http://perl.apache.org/docs/2.0/index.html.

mod_php

PHP: Препроцессор Гипертекста (PHP) — это язык программирования общего назначения, который особенно хорошо подходит для веб-разработки. Способный встраиваться в HTML, его синтаксис основан на C, Java™ и Perl, что позволяет веб-разработчикам быстро создавать динамически генерируемые веб-страницы.

Поддержка PHP для Apache и любых других функций, написанных на этом языке, может быть добавлена путем установки соответствующего порта.

Для всех поддерживаемых версий выполните поиск в базе данных пакетов с помощью `pkg`:

```
# pkg search php
```

Будет отображён список, включающий версии и дополнительные возможности, которые они предоставляют. Компоненты полностью модульные, что означает, что функции включаются путем установки соответствующего порта. Чтобы установить PHP версии 7.4 для Apache, выполните следующую команду:

```
# pkg install mod_php74
```

Если необходимо установить какие-либо зависимости, они также будут установлены.

По умолчанию PHP не будет включен. Следующие строки необходимо добавить в конфигурационный файл Apache, расположенный в /usr/local/etc/apache24, чтобы активировать его:

```
<FilesMatch "\.php$">
    SetHandler application/x-httpd-php
</FilesMatch>
<FilesMatch "\.phps$">
    SetHandler application/x-httpd-php-source
</FilesMatch>
```

В дополнение, параметр `DirectoryIndex` в конфигурационном файле также потребуется обновить, и Apache нужно будет перезапустить или перезагрузить, чтобы изменения вступили в силу.

Поддержка многих функций PHP также может быть установлена с помощью `pkg`. Например, для установки поддержки XML или SSL установите соответствующие порты:

```
# pkg install php74-xml php74-openssl
```

Как и ранее, для вступления изменений в силу необходимо перезагрузить конфигурацию Apache, даже в случаях, когда была просто установка модуля.

Для выполнения плавного перезапуска с целью перезагрузки конфигурации выполните следующую команду:

```
# apachectl graceful
```

После завершения установки есть два способа получить установленные модули поддержки PHP и информацию о среде сборки. Первый — установить полный бинарный файл PHP и выполнить команду для получения информации:

```
# pkg install php74
```

```
# php -i | less
```

Необходимо передать вывод в постраничный просмотрщик, например, `more` или `less`, чтобы упростить восприятие большого объёма вывода.

Наконец, для внесения изменений в глобальную конфигурацию PHP существует хорошо документированный файл, установленный в /usr/local/etc/php.ini. На момент установки этот файл не будет существовать, так как есть две версии на выбор: php.ini-development и php.ini-production. Они представляют собой отправные точки, помогающие администраторам в развертывании.

Поддержка HTTP2

Поддержка протокола HTTP2 в Apache включена по умолчанию при установке порта с помощью `pkg`. Новая версия HTTP содержит множество улучшений по сравнению с предыдущей версией, включая использование одного соединения с веб-сайтом, что сокращает общее количество циклов TCP-соединений. Кроме того, данные заголовков пакетов сжимаются, а HTTP2 по умолчанию требует шифрования.

Когда Apache настроен на использование только HTTP2, веб-браузеры будут требовать безопасное, зашифрованное HTTPS-соединение. Если Apache настроен на использование обеих версий, HTTP1.1 будет считаться резервным вариантом при возникновении проблем с соединением.

Хотя это изменение требует от администраторов внесения изменений, они положительные и способствуют более безопасному Интернету для всех. Изменения требуются только для сайтов, которые в настоящее время не используют SSL и TLS.

Эта конфигурация зависит от предыдущих разделов, включая поддержку TLS. Рекомендуется выполнить эти инструкции перед продолжением с данной конфигурацией.

Начните процесс, включив модуль http2, раскомментировав строку в /usr/local/etc/apache24/httpd.conf, и замените модуль mpm_prefork на mpm_event, так как первый не поддерживает HTTP2.

```
LoadModule http2_module libexec/apache24/mod_http2.so
LoadModule mpm_event_module libexec/apache24/mod_mpm_event.so
```

Существует отдельный порт mod_http2, который доступен. Он предназначен для более быстрого получения исправлений безопасности и ошибок по сравнению с модулем, установленным через встроенный порт apache24. Он не обязателен для поддержки HTTP2, но доступен. При установке следует использовать mod_h2.so вместо mod_http2.so в конфигурации Apache.

Существует два метода реализации HTTP2 в Apache; один способ — глобально для всех сайтов и каждого VirtualHost, работающего в системе. Чтобы включить HTTP2 глобально, добавьте следующую строку под директивой ServerName:

```
Protocols h2 http/1.1
```

Для включения HTTP2 в незашифрованном виде используйте h2h2chttp/1.1 в файле httpd.conf.

Наличие здесь h2c позволит передавать незашифрованные данные HTTP2 в системе, но это не рекомендуется. Кроме того, использование здесь http/1.1 позволит системе вернуться к версии протокола HTTP1.1, если это потребуется.

Для включения HTTP2 для отдельных VirtualHosts добавьте ту же строку в директиву VirtualHost в файле httpd.conf или httpd-ssl.conf.

Перезагрузите конфигурацию с помощью команды `apachectl`reload и проверьте конфигурацию каким-либо из следующих способов после посещения одной из страниц на сервере:

```
# grep "HTTP/2.0" /var/log/httpd-access.log
```

Это должно вернуть что-то похожее на следующее:

```
192.168.1.205 - - [18/Oct/2020:18:34:36 -0400] "GET / HTTP/2.0" 304 -
192.0.2.205 - - [18/Oct/2020:19:19:57 -0400] "GET / HTTP/2.0" 304 -
192.0.0.205 - - [18/Oct/2020:19:20:52 -0400] "GET / HTTP/2.0" 304 -
192.0.2.205 - - [18/Oct/2020:19:23:10 -0400] "GET / HTTP/2.0" 304 -
```

Другой способ — использование встроенного в веб-браузер отладчика сайтов или `tcpdump`; однако использование любого из этих методов выходит за рамки данного документа.

Поддержка обратных прокси-соединений HTTP2 с использованием модуля mod_proxy_http2.so. При настройке директив ProxyPass или RewriteRules с флагом [P] следует использовать h2:// для соединения.

33.9.4. Динамические веб-сайты

В дополнение к mod_perl и mod_php, доступны другие языки для создания динамического веб-содержимого. Среди них Django и Ruby on Rails.

Django

Django — это фреймворк под лицензией BSD, разработанный для того, чтобы позволить разработчикам быстро создавать высокопроизводительные и элегантные веб-приложения. Он предоставляет объектно-реляционный преобразователь (ORM), позволяющий разрабатывать типы данных как объекты Python. Для этих объектов предоставляется богатый динамический API доступа к базе данных, без необходимости написания разработчиком кода на SQL. Также имеется расширяемая система шаблонов, чтобы логика приложения была отделена от HTML-представления.

Django зависит от mod_python и движка SQL-базы данных. В FreeBSD порт www/py-django автоматически устанавливает mod_python и поддерживает базы данных PostgreSQL, MySQL или SQLite, по умолчанию используется SQLite. Чтобы изменить движок базы данных, введите `make config` в /usr/ports/www/py-django, затем установите порт.

После установки Django приложению понадобится каталог проекта вместе с конфигурацией Apache для использования встроенного интерпретатора Python. Этот интерпретатор используется для вызова приложения при обращении к определённым URL на сайте.

Для настройки Apache для передачи запросов определённых URL веб-приложению добавьте следующее в httpd.conf, указав полный путь к каталогу проекта:

```
<Location "/">
    SetHandler python-program
    PythonPath "['/dir/to/the/django/packages/'] + sys.path"
    PythonHandler django.core.handlers.modpython
    SetEnv DJANGO_SETTINGS_MODULE mysite.settings
    PythonAutoReload On
    PythonDebug On
</Location>
```

Обратитесь к https://docs.djangoproject.com для получения дополнительной информации о том, как использовать Django.

Ruby on Rails

Ruby on Rails — это ещё один фреймворк с открытым исходным кодом для веб-разработки, предоставляющий полный стек разработки. Он оптимизирован для повышения продуктивности веб-разработчиков и позволяет быстро создавать мощные приложения. В FreeBSD его можно установить с помощью пакета www/rubygem-rails или порта.

Обратитесь к http://guides.rubyonrails.org для получения дополнительной информации о том, как использовать Ruby on Rails.

33.10. Протокол передачи файлов (FTP)

Протокол передачи файлов (FTP — File Transfer Protocol) предоставляет пользователям простой способ передачи файлов на FTP-сервер и с него. В базовую систему FreeBSD включено программное обеспечение FTP-сервера — ftpd.

В FreeBSD предусмотрено несколько файлов конфигурации для управления доступом к FTP-серверу. В этом разделе приводится их краткое описание. Подробнее о встроенном FTP-сервере можно узнать в ftpd(8).

33.10.1. Конфигурация

Самый важный этап настройки — определение учётных записей, которым будет разрешён доступ к FTP-серверу. В системе FreeBSD существует ряд системных учётных записей, которым не следует разрешать доступ по FTP. Список пользователей, которым запрещён любой доступ по FTP, можно найти в /etc/ftpusers. По умолчанию в него включены системные учётные записи. Можно добавить дополнительных пользователей, которым не следует разрешать доступ к FTP.

В некоторых случаях может быть желательно ограничить доступ некоторых пользователей, не запрещая им полностью использовать FTP. Это можно сделать, создав файл /etc/ftpchroot, как описано в ftpchroot(5). В этом файле перечислены пользователи и группы, подлежащие ограничениям доступа к FTP.

Для обеспечения анонимного доступа по FTP к серверу создайте пользователя с именем `ftp` в системе FreeBSD. Пользователи смогут входить на FTP-сервер под именем `ftp` или `anonymous`. При запросе пароля будет принят любой ввод, но по соглашению в качестве пароля следует использовать адрес электронной почты. FTP-сервер вызовет chroot(2) при входе анонимного пользователя, чтобы ограничить доступ только домашним каталогом пользователя `ftp`.

Существуют два текстовых файла, которые можно создать для отображения приветственных сообщений клиентам FTP. Содержимое файла /etc/ftpwelcome будет показано пользователям до появления запроса на вход. После успешного входа будет отображено содержимое файла /etc/ftpmotd. Обратите внимание, что путь к этому файлу указывается относительно окружения входа, поэтому для анонимных пользователей будет отображаться содержимое файла ~ftp/etc/ftpmotd.

После настройки FTP-сервера установите соответствующую переменную в /etc/rc.conf, чтобы служба запускалась при загрузке:

```
ftpd_enable="YES"
```

Чтобы запустить службу сейчас:

```
# service ftpd start
```

Протестируйте подключение к FTP-серверу, набрав:

```
% ftp localhost
```

Демон ftpd использует syslog(3) для записи сообщений. По умолчанию, демон системного журнала записывает сообщения, связанные с FTP, в /var/log/xferlog. Местоположение журнала FTP может быть изменено путём редактирования следующей строки в /etc/syslog.conf:

```
ftp.info      /var/log/xferlog
```

Имейте в виду потенциальные проблемы, связанные с запуском анонимного FTP-сервера. В частности, хорошо подумайте, прежде чем разрешать анонимным пользователям загружать файлы. Может оказаться, что FTP-сайт станет площадкой для обмена нелицензионным коммерческим программным обеспечением или даже чем-то хуже. Если загрузка файлов анонимными пользователями необходима, убедитесь в правильности настроек прав доступа, чтобы эти файлы не могли быть прочитаны другими анонимными пользователями до тех пор, пока их не проверит администратор.

33.11. Услуги файлов и печати для клиентов Microsoft® Windows® (Samba)

Samba — это популярный пакет открытого программного обеспечения, предоставляющий файловые и печатные услуги с использованием протокола SMB/CIFS. Этот протокол встроен в системы Microsoft® Windows®. Он может быть добавлен в системы, отличные от Microsoft® Windows®, путем установки клиентских библиотек Samba. Протокол позволяет клиентам получать доступ к общим данным и принтерам. Эти ресурсы могут быть отображены как локальный диск, а общие принтеры могут использоваться так, как если бы они были локальными.

На FreeBSD клиентские библиотеки Samba могут быть установлены с помощью порта или пакета net/samba416. Клиент предоставляет возможность системе FreeBSD получать доступ к общим ресурсам SMB/CIFS в сети Microsoft® Windows®.

Система FreeBSD также может быть настроена в качестве сервера Samba путем установки порта или пакета net/samba416. Это позволяет администратору создавать общие ресурсы SMB/CIFS на системе FreeBSD, к которым могут обращаться клиенты под управлением Microsoft® Windows® или использующие клиентские библиотеки Samba.

33.11.1. Конфигурация сервера

Samba настраивается в файле /usr/local/etc/smb4.conf. Этот файл должен быть создан до начала использования Samba.

Простой пример smb4.conf для общего доступа к каталогам и принтерам с клиентами Windows® в рабочей группе показан ниже. Для более сложных настроек, включающих LDAP или Active Directory, проще использовать samba-tool(8) для создания начального smb4.conf.

```
[global]
workgroup = WORKGROUP
server string = Samba Server Version %v
netbios name = ExampleMachine
wins support = Yes
security = user
passdb backend = tdbsam

# Example: share /usr/src accessible only to 'developer' user
[src]
path = /usr/src
valid users = developer
writable  = yes
browsable = yes
read only = no
guest ok = no
public = no
create mask = 0666
directory mask = 0755
```

Глобальные настройки

Настройки, описывающие сеть, добавляются в /usr/local/etc/smb4.conf:

`workgroup`

Имя рабочей группы, которая будет обслуживаться.

`netbios name`

Имя NetBIOS, под которым известен сервер Samba. По умолчанию оно совпадает с первой частью DNS-имени хоста.

`server string`

Строка, которая будет отображаться в выводе команды `net view` и некоторых других сетевых инструментов, предназначенных для отображения описательного текста о сервере.

`wins support`

Будет ли Samba выступать в качестве сервера WINS. Не следует включать поддержку WINS более чем на одном сервере в сети.

Настройки безопасности

Важнейшие настройки в /usr/local/etc/smb4.conf — это модель безопасности и формат хранения паролей. Эти параметры управляются следующими директивами:

`security`

Если клиенты используют имена пользователей, совпадающие с их именами на машине FreeBSD, следует использовать уровень безопасности пользователя. `security = user` — это политика безопасности по умолчанию, которая требует от клиентов сначала войти в систему, прежде чем они смогут получить доступ к общим ресурсам.

Обратитесь к smb.conf(5), чтобы узнать о других поддерживаемых настройках для опции `security`.

`passdb backend`

Samba поддерживает несколько различных моделей аутентификации на стороне сервера. Клиенты могут быть аутентифицированы с помощью LDAP, NIS+, SQL-базы данных или модифицированного файла паролей. Рекомендуемый метод аутентификации `tdbsam` идеально подходит для простых сетей, и мы его рассмотрим здесь. Для более крупных или сложных сетей рекомендуется `ldapsam`. `smbpasswd` был прежним методом, используемым по умолчанию, и теперь устарел.

Пользователи Samba

Пользовательские учётные записи FreeBSD должны быть сопоставлены с базой данных `SambaSAMAccount` для доступа клиентов Windows® к общему ресурсу. Сопоставьте существующие учётные записи FreeBSD с помощью pdbedit(8):

```
# pdbedit -a -u username
```

В этом разделе упомянуты только наиболее часто используемые настройки. Дополнительную информацию о доступных параметрах конфигурации можно найти на Официальном вики Samba.

33.11.2. Начало работы с Samba

Чтобы включить Samba при загрузке, добавьте следующую строку в /etc/rc.conf:

```
samba_server_enable="YES"
```

Чтобы сейчас запустить Samba:

```
# service samba_server start
Performing sanity check on Samba configuration: OK
Starting nmbd.
Starting smbd.
```

Samba состоит из трёх отдельных демонов. Оба демона nmbd и smbd запускаются параметром `samba_enable`. Если также требуется разрешение имён через winbind, укажите:

```
winbindd_enable="YES"
```

Samba можно остановить в любой момент, набрав:

```
# service samba_server stop
```

Samba — это комплексный программный комплект, функциональность которого обеспечивает широкую интеграцию с сетями Microsoft® Windows®. Для получения дополнительной информации о возможностях, выходящих за рамки базовой конфигурации, описанной здесь, обратитесь к https://www.samba.org.

33.12. Синхронизация времени с помощью NTP

Со временем часы компьютера могут отставать или спешить. Это создаёт проблемы, так как многие сетевые службы требуют, чтобы компьютеры в сети использовали одинаковое точное время. Точное время также необходимо для обеспечения согласованности временных меток файлов. Протокол сетевого времени (NTP — Network Time Protocol) — это один из способов обеспечить точность часов в сети.

FreeBSD включает ntpd(8), который можно настроить для запроса к другим серверам NTP с целью синхронизации часов на этом компьютере или для предоставления сервиса времени другим компьютерам в сети.

В этом разделе описывается, как настроить ntpd в FreeBSD. Дополнительная документация доступна в /usr/share/doc/ntp/ в формате HTML.

33.12.1. Конфигурация NTP

На FreeBSD встроенный ntpd может использоваться для синхронизации системных часов. Настройка ntpd осуществляется с помощью переменных rc.conf(5) и файла /etc/ntp.conf, как подробно описано в следующих разделах.

ntpd взаимодействует с сетевыми узлами с помощью UDP-пакетов. Любые межсетевые экраны между вашей машиной и её NTP-узлами должны быть настроены так, чтобы разрешать входящие и исходящие UDP-пакеты через порт 123.

Файл /etc/ntp.conf

ntpd читает файл /etc/ntp.conf, чтобы определить, к каким серверам NTP обращаться. Рекомендуется выбирать несколько серверов NTP на случай, если один из серверов станет недоступен или его часы окажутся ненадёжными. По мере получения ответов ntpd отдаёт предпочтение более надёжным серверам перед менее надёжными. Запрашиваемые серверы могут быть локальными в сети, предоставляться ISP или выбираться из онлайн-списка общедоступных серверов NTP. При выборе общедоступного сервера NTP следует

выбирать сервер, географически близкий к вам, и ознакомиться с его политикой использования. Ключевое слово `pool` в конфигурации выбирает один или несколько серверов из пула серверов. Доступен онлайн-список общедоступных пулов NTP, организованный по географическим регионам. Кроме того, FreeBSD предоставляет спонсируемый проектом пул `0.freebsd.pool.ntp.org`.

Пример 43. Пример /etc/ntp.conf

Вот простой пример файла ntp.conf. Его можно безопасно использовать в таком виде; он содержит рекомендуемые параметры `restrict` для работы в общедоступном сетевом подключении.

```
# Disallow ntpq control/query access.  Allow peers to be added only
# based on pool and server statements in this file.
restrict default limited kod nomodify notrap noquery nopeer
restrict source  limited kod nomodify notrap noquery

# Allow unrestricted access from localhost for queries and control.
restrict 127.0.0.1
restrict ::1

# Add a specific server.
server ntplocal.example.com iburst

# Add FreeBSD pool servers until 3-6 good servers are available.
tos minclock 3 maxclock 6
pool 0.freebsd.pool.ntp.org iburst

# Use a local leap-seconds file.
leapfile "/var/db/ntpd.leap-seconds.list"
```

Формат этого файла описан в ntp.conf(5). Приведённые ниже описания дают краткий обзор только ключевых слов, использованных в примере файла выше.

По умолчанию сервер NTP доступен для любого узла сети. Ключевое слово `restrict` управляет тем, какие системы могут обращаться к серверу. Поддерживается несколько записей `restrict`, каждая из которых уточняет ограничения, заданные в предыдущих утверждениях. Значения, указанные в примере, предоставляют локальной системе полный доступ для запросов и управления, в то время как удалённые системы могут только запрашивать время. Для получения дополнительной информации обратитесь к подразделу `Access Control Support` в ntp.conf(5).

Ключевое слово `server` указывает отдельный сервер для запросов. Файл может содержать несколько ключевых слов `server`, по одному серверу на каждой строке. Ключевое слово `pool` определяет пул серверов. ntpd добавит один или несколько серверов из этого пула по мере необходимости, чтобы достичь количества узлов, указанного с помощью значения `tos minclock`. Ключевое слово `iburst` предписывает ntpd выполнить серию из восьми быстрых обменов пакетами с сервером при первом установлении соединения, чтобы быстро

синхронизировать системное время.

Ключевое слово `leapfile` указывает расположение файла, содержащего информацию о високосных секундах. Этот файл автоматически обновляется с помощью periodic(8). Указанное расположение файла должно соответствовать значению переменной `ntp_db_leapfile` в файле /etc/rc.conf.

Записи NTP в /etc/rc.conf

Установите `ntpd_enable=YES` для запуска ntpd при загрузке. После добавления `ntpd_enable=YES` в /etc/rc.conf, ntpd можно немедленно запустить без перезагрузки системы, введя:

```
# service ntpd start
```

Для использования ntpd необходимо установить только `ntpd_enable`. При необходимости также могут быть заданы перечисленные ниже переменные rc.conf.

Установите `ntpd_sync_on_start=YES`, чтобы разрешить ntpd однократно корректировать время при запуске на любую величину. Обычно ntpd записывает сообщение об ошибке и завершает работу, если расхождение времени превышает 1000 секунд. Эта опция особенно полезна для систем без аккумуляторного резервного питания часов реального времени.

Установите `ntpd_oomprotect=YES`, чтобы защитить демон ntpd от завершения системой при попытке восстановиться после состояния нехватки памяти (OOM).

Установите в значении `ntpd_config=` ` расположение альтернативного файла ntp.conf.

Установите `ntpd_flags=` с любыми другими флагами ntpd по необходимости, но избегайте использования тех флагов, которые устанавливаются внутри файла /etc/rc.d/ntpd:

- `-p` (расположение pid-файла)
- `-c` (вместо этого установите `ntpd_config=`)

ntpd и непривилегированный пользователь `ntpd`

ntpd в FreeBSD может запускаться и работать как непривилегированный пользователь. Для этого требуется модуль политики mac_ntpd(4). Скрипт запуска /etc/rc.d/ntpd сначала проверяет конфигурацию NTP. Если возможно, он загружает модуль `mac_ntpd`, а затем запускает ntpd как непривилегированный пользователь `ntpd` (идентификатор пользователя 123). Чтобы избежать проблем с доступом к файлам и каталогам, скрипт запуска не будет автоматически запускать ntpd как `ntpd`, если конфигурация содержит любые файлозависимые параметры.

Присутствие любого из следующих параметров в `ntpd_flags` требует ручной настройки, как описано ниже, для запуска от пользователя `ntpd`:

- -f or --driftfile
- -i or --jaildir

- -k or --keyfile
- -l or --logfile
- -s or --statsdir

Наличие любого из следующих ключевых слов в ntp.conf требует ручной настройки, как описано ниже, для запуска от пользователя `ntpd`:

- crypto
- driftfile
- key
- logdir
- statsdir

Для ручной настройки ntpd для запуска от пользователя `ntpd` необходимо:

- Убедитесь, что пользователь `ntpd` имеет доступ ко всем файлам и каталогам, указанным в конфигурации.
- Обеспечьте загрузку или компиляцию модуля `mac_ntpd` в ядро. Подробности см. в mac_ntpd(4).
- Установите `ntpd_user="ntpd"` в /etc/rc.conf

33.12.2. Использование NTP с PPP-подключением

ntpd не требует постоянного подключения к Интернету для корректной работы. Однако, если PPP-соединение настроено на дозвон по требованию, следует предотвратить инициацию дозвона или поддержание соединения из-за трафика NTP. Это можно настроить с помощью директив `filter` в /etc/ppp/ppp.conf. Например:

```
set filter dial 0 deny udp src eq 123
# Prevent NTP traffic from initiating dial out
set filter dial 1 permit 0 0
set filter alive 0 deny udp src eq 123
# Prevent incoming NTP traffic from keeping the connection open
set filter alive 1 deny udp dst eq 123
# Prevent outgoing NTP traffic from keeping the connection open
set filter alive 2 permit 0/0 0/0
```

Для получения более подробной информации обратитесь к разделу `ФИЛЬТРАЦИЯ ПАКЕТОВ` в ppp(8) и примерам в /usr/share/examples/ppp/.

Некоторые интернет-провайдеры блокируют порты с низкими номерами, что мешает работе NTP, так как ответы никогда не достигают машины.

33.13. Настройка инициатора и цели iSCSI

iSCSI — это способ совместного использования хранилища по сети. В отличие от NFS, который работает на уровне файловой системы, iSCSI работает на уровне блочного устройства.

В терминологии iSCSI система, предоставляющая хранилище, называется *целью*. Хранилище может быть физическим диском, областью, представляющей несколько дисков, или частью физического диска. Например, если диск(и) отформатированы с использованием ZFS, можно создать zvol для использования в качестве хранилища iSCSI.

Клиенты, которые обращаются к хранилищу iSCSI, называются *инициаторами*. Для инициаторов хранилище, доступное через iSCSI, отображается как неформатированный диск, известный как LUN (логический номер устройства). Узлы устройств для диска появляются в /dev/, и устройство должно быть отдельно отформатировано и смонтировано.

FreeBSD предоставляет встроенную поддержку iSCSI целевой системы и инициатора на уровне ядра. В этом разделе описывается, как настроить систему FreeBSD в качестве целевой системы или инициатора.

33.13.1. Настройка цели iSCSI

Для настройки цели iSCSI создайте конфигурационный файл /etc/ctl.conf, добавьте строку в /etc/rc.conf, чтобы убедиться, что демон ctld(8) автоматически запускается при загрузке, а затем запустите демон.

Вот пример простого файла конфигурации /etc/ctl.conf. Полное описание доступных опций этого файла можно найти в ctl.conf(5).

```
portal-group pg0 {
    discovery-auth-group no-authentication
    listen 0.0.0.0
    listen [::]
}

target iqn.2012-06.com.example:target0 {
    auth-group no-authentication
    portal-group pg0

    lun 0 {
        path /data/target0-0
        size 4G
    }
}
```

Первая запись определяет группу порталов `pg0`. Группы порталов определяют, на каких сетевых адресах будет слушать демон ctld(8). Запись `discovery-auth-group no-authentication` указывает, что любой инициатор может выполнять обнаружение целей iSCSI без

аутентификации. Третья и четвёртая строки настраивают ctld(8) для прослушивания всех IPv4-адресов (`listen 0.0.0.0`) и IPv6-адресов (`listen [::]`) на стандартном порту 3260.

Нет необходимости определять группу порталов, так как существует встроенная группа порталов с именем `default`. В этом случае разница между `default` и `pg0` заключается в том, что для `default` обнаружение целей всегда запрещено, а для `pg0` — всегда разрешено.

Вторая запись определяет одну цель. У цели есть два возможных значения: машина, обслуживающая iSCSI, или именованная группа LUN. В этом примере используется второе значение, где `iqn.2012-06.com.example:target0` — это имя цели. Это имя цели подходит для тестирования. Для реального использования замените `com.example` на настоящий домен, записанный в обратном порядке. `2012-06` представляет год и месяц получения контроля над этим доменом, а `target0` может быть любым значением. В этом файле конфигурации можно определить любое количество целей.

Строка `auth-group no-authentication` разрешает всем инициаторам подключаться к указанной цели, а `portal-group pg0` делает цель доступной через группу порталов `pg0`.

Следующий раздел определяет LUN. Для инициатора каждый LUN будет виден как отдельное дисковое устройство. Для каждой цели можно определить несколько LUN. Каждый LUN идентифицируется числом, где LUN 0 является обязательным. Строка `path /data/target0-0` определяет полный путь к файлу или zvol, который используется для LUN. Этот путь должен существовать до запуска ctld(8). Вторая строка необязательна и указывает размер LUN.

Далее, чтобы убедиться, что демон ctld(8) запускается при загрузке, добавьте эту строку в /etc/rc.conf:

```
ctld_enable="YES"
```

Чтобы запустить ctld(8) сейчас, выполните следующую команду:

```
# service ctld start
```

Поскольку демон ctld(8) запускается, он читает файл /etc/ctl.conf. Если этот файл был изменён после запуска демона, используйте следующую команду, чтобы изменения вступили в силу немедленно:

```
# service ctld reload
```

Аутентификация

Предыдущий пример изначально небезопасен, так как не использует аутентификацию, предоставляя любому полный доступ ко всем целям. Чтобы потребовать имя пользователя и пароль для доступа к целям, измените конфигурацию следующим образом:

```
auth-group ag0 {
    chap username1 secretsecret
    chap username2 anothersecret
}

portal-group pg0 {
    discovery-auth-group no-authentication
    listen 0.0.0.0
    listen [::]
}

target iqn.2012-06.com.example:target0 {
    auth-group ag0
    portal-group pg0
    lun 0 {
        path /data/target0-0
        size 4G
    }
}
```

Раздел `auth-group` определяет пары имени пользователя и пароля. Инициатор, пытающийся подключиться к `iqn.2012-06.com.example:target0`, должен сначала указать определённое имя пользователя и секрет. Однако обнаружение цели по-прежнему разрешено без аутентификации. Чтобы потребовать аутентификацию при обнаружении цели, установите `discovery-auth-group` в определённое имя `auth-group` вместо `no-authentication`.

Обычно определяют один экспортируемый объект для каждого инициатора. В качестве сокращения для синтаксиса выше, имя пользователя и пароль могут быть указаны непосредственно в записи объекта:

```
target iqn.2012-06.com.example:target0 {
    portal-group pg0
    chap username1 secretsecret

    lun 0 {
        path /data/target0-0
        size 4G
    }
}
```

33.13.2. Настройка инициатора iSCSI

Описанный в этом разделе инициатор iSCSI поддерживается начиная с FreeBSD 10.0-RELEASE. Для использования инициатора iSCSI, доступного в более старых версиях, обратитесь к iscontrol(8).

Инициатору iSCSI требуется, чтобы демон iscsid(8) был запущен. Этот демон не использует

файл конфигурации. Для его автоматического запуска при загрузке добавьте следующую строку в /etc/rc.conf:

```
iscsid_enable="YES"
```

Чтобы сейчас запустить iscsid(8), выполните следующую команду:

```
# service iscsid start
```

Подключение к цели может быть выполнено с файлом конфигурации /etc/iscsi.conf или без него. В этом разделе показаны оба типа подключений.

Подключение к цели без файла конфигурации

Для подключения инициатора к одному целевому устройству укажите IP-адрес портала и имя целевого устройства:

```
# iscsictl -A -p 10.10.10.10 -t iqn.2012-06.com.example:target0
```

Для проверки успешности соединения выполните команду `iscsictl` без аргументов. Вывод должен выглядеть примерно так:

```
Target name                                    Target portal   State
iqn.2012-06.com.example:target0                10.10.10.10     Connected: da0
```

В этом примере сеанс iSCSI был успешно установлен, где /dev/da0 представляет подключённый LUN. Если цель `iqn.2012-06.com.example:target0` экспортирует более одного LUN, в соответствующем разделе вывода будет показано несколько устройств:

```
Connected: da0 da1 da2.
```

Любые ошибки будут отображены в выводе, а также в системных журналах. Например, это сообщение обычно означает, что демон iscsid(8) не запущен:

```
Target name                                    Target portal   State
iqn.2012-06.com.example:target0                10.10.10.10     Waiting for iscsid(8)
```

Следующее сообщение указывает на проблему с сетью, например, неверный IP-адрес или порт:

```
Target name                                    Target portal   State
iqn.2012-06.com.example:target0                10.10.10.11     Connection refused
```

Это сообщение означает, что указано неправильное имя цели:

```
Target name                                     Target portal   State
iqn.2012-06.com.example:target0                 10.10.10.10     Not found
```

Это сообщение означает, что цель требует аутентификации:

```
Target name                                     Target portal   State
iqn.2012-06.com.example:target0                 10.10.10.10     Authentication failed
```

Чтобы указать имя пользователя CHAP и секрет, используйте следующий синтаксис:

```
# iscsictl -A -p 10.10.10.10 -t iqn.2012-06.com.example:target0 -u user -s
secretsecret
```

Подключение к цели с использованием файла конфигурации

Для подключения с использованием файла конфигурации создайте файл /etc/iscsi.conf с содержимым, подобным этому:

```
t0 {
    TargetAddress   = 10.10.10.10
    TargetName      = iqn.2012-06.com.example:target0
    AuthMethod      = CHAP
    chapIName       = user
    chapSecret      = secretsecret
}
```

`t0` задаёт псевдоним для раздела конфигурационного файла. Он будет использоваться инициатором для указания, какую конфигурацию применять. Остальные строки определяют параметры, используемые при подключении. `TargetAddress` и `TargetName` являются обязательными, тогда как остальные параметры — опциональными. В этом примере показаны имя пользователя CHAP и секретный ключ.

Для подключения к указанной цели укажите псевдоним:

```
# iscsictl -An t0
```

Или для подключения ко всем целям, определённым в файле конфигурации, используйте:

```
# iscsictl -Aa
```

Чтобы инициатор автоматически подключался ко всем целям в /etc/iscsi.conf, добавьте

следующее в /etc/rc.conf:

```
iscsictl_enable="YES"
iscsictl_flags="-Aa"
```

Глава 34. Межсетевые экраны

34.1. Обзор

Межсетевые экраны позволяют фильтровать входящий и исходящий трафик, проходящий через систему. Межсетевой экран может использовать один или несколько наборов "правил" для проверки сетевых пакетов при их поступлении или выходе из сетевых соединений и либо пропускает трафик, либо блокирует его. Правила межсетевого экрана могут проверять одну или несколько характеристик пакетов, таких как тип протокола, адрес исходного или целевого хоста, а также исходный или целевой порт.

Межсетевые экраны могут повысить безопасность узла или сети. Они могут использоваться для выполнения одной или нескольких следующих функций:

- Защищать и изолировать приложения, сервисы и машины внутренней сети от нежелательного трафика из общедоступного Интернета.
- Ограничивать или отключать доступ с хостов внутренней сети к сервисам публичного Интернета.
- Поддерживать преобразования сетевых адресов (NAT), что позволяет внутренней сети использовать частные IP-адреса и совместно использовать одно подключение к публичному интернету с помощью одного IP-адреса или общего пула автоматически назначаемых публичных адресов.

В базовой системе FreeBSD встроено три межсетевых экрана: PF, IPFW и IPFILTER, также известный как IPF. FreeBSD также предоставляет два инструмента для управления использованием пропускной способности: altq(4) и dummynet(4). ALTQ традиционно тесно связан с PF, а dummynet — с IPFW. Каждый межсетевой экран использует правила для контроля доступа пакетов к системе FreeBSD и из неё, хотя подходы у них различаются, и каждый имеет свой собственный синтаксис правил.

FreeBSD предоставляет несколько межсетевых экранов для удовлетворения различных требований и предпочтений широкого круга пользователей. Каждый пользователь должен оценить, какой межсетевой экран лучше всего соответствует его потребностям.

Прочтите эту главу, чтобы узнать:

- Как определить правила фильтрации пакетов.
- Различия между межсетевыми экранами, встроенными в FreeBSD.
- Как использовать и настроить межсетевой экран PF.
- Как использовать и настроить межсетевой экран IPFW.
- Как использовать и настроить межсетевой экран IPFILTER.

Прежде чем читать эту главу, необходимо:

- Понимать основы FreeBSD и принципы работы Интернета.

Поскольку все межсетевые экраны основаны на проверке значений выбранных управляющих полей пакетов, создатель набора правил межсетевого экрана должен понимать, как работает TCP/IP, какие значения содержатся в управляющих полях пакетов и как эти значения используются в обычном сеансе связи. Хорошее введение можно найти в TCP/IP Primer от Daryl.

34.2. Концепции межсетевого экрана

Набор правил содержит группу правил, которые пропускают или блокируют пакеты на основе значений, содержащихся в пакете. Двунаправленный обмен пакетами между хостами составляет сеанс взаимодействия. Межсетевой экран обрабатывает как пакеты, поступающие из общедоступного Интернета, так и пакеты, генерируемые системой в ответ на них. Каждая служба TCP/IP имеет свой протокол и порт прослушивания. Пакеты, предназначенные для конкретной службы, идут из исходного адреса с использованием непривилегированного порта и направляются на конкретный порт службы на адресе назначения. Все вышеуказанные параметры могут быть использованы в качестве критериев выбора для создания правил, которые будут пропускать или блокировать службы.

Для поиска неизвестных номеров портов обратитесь к /etc/services. Или посетите https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers и выполните поиск по номеру порта, чтобы узнать его назначение.

Ознакомьтесь с этой ссылкой для номеров портов, используемых троянами.

FTP имеет два режима: активный режим и пассивный режим. Разница заключается в том, как устанавливается канал передачи данных. Пассивный режим более безопасен, так как канал передачи данных устанавливается инициатором сессии FTP. Для подробного объяснения работы FTP и различных режимов см. http://www.slacksite.com/other/ftp.html.

Набор правил межсетевого экрана может быть "исключающим" или "включающим". Исключающий межсетевой экран пропускает весь трафик, кроме трафика, соответствующего набору правил. Включающий межсетевой экран действует наоборот, пропуская только трафик, соответствующий правилам, и блокируя всё остальное.

Включающий межсетевой экран обеспечивает лучший контроль исходящего трафика, что делает его предпочтительным выбором для систем, предоставляющих услуги в публичном Интернете. Он также контролирует тип трафика, исходящего из публичного Интернета, который может получить доступ к частной сети. Весь трафик, не соответствующий правилам, блокируется и регистрируется. Включающие межсетевые экраны, как правило, безопаснее исключающих, поскольку они значительно снижают риск пропуска нежелательного трафика.

Если не указано иное, все конфигурации и примеры наборов правил в этой главе создают включающие наборы правил для межсетевого экрана.

Безопасность можно дополнительно усилить с помощью "межсетевого экрана с

отслеживанием состояния". Такой тип межсетевого экрана отслеживает активные соединения и разрешает только трафик, который соответствует существующему соединению или открывает новое, разрешённое соединение.

Фильтрация с отслеживанием состояния рассматривает трафик как двусторонний обмен пакетами, составляющими сеанс. Когда для соответствующего правила указано состояние, межсетевой экран динамически создаёт внутренние правила для каждого ожидаемого пакета, обмениваемого во время сеанса. Он обладает достаточными возможностями сопоставления, чтобы определить, является ли пакет допустимым для сеанса. Любые пакеты, которые не соответствуют шаблону сеанса, автоматически отклоняются.

Когда сеанс завершается, он удаляется из динамической таблицы состояний.

Фильтрация с сохранением состояния позволяет сосредоточиться на блокировке/пропускании новых сеансов. Если новый сеанс пропущен, все последующие его пакеты автоматически разрешаются, а любые поддельные пакеты автоматически отвергаются. Если новый сеанс заблокирован, ни один из его последующих пакетов не разрешается. Фильтрация с сохранением состояния предоставляет расширенные возможности сопоставления, способные защитить от потока различных методов атак, используемых злоумышленниками.

NAT означает *Network Address Translation* (Преобразование сетевых адресов). Функция NAT позволяет частной локальной сети за межсетевым экраном использовать один IP-адрес, назначенный провайдером, даже если этот адрес динамический. NAT даёт возможность каждому компьютеру в локальной сети выходить в Интернет без необходимости оплачивать провайдеру несколько интернет-аккаунтов или IP-адресов.

NAT автоматически преобразует частный IP-адрес LAN для каждой системы в локальной сети в единственный публичный IP-адрес, когда пакеты выходят за пределы межсетевого экрана, направляясь в публичный Интернет. Он также выполняет обратное преобразование для возвращающихся пакетов.

Согласно RFC 1918, следующие диапазоны IP-адресов зарезервированы для частных сетей, которые никогда не маршрутизируются напрямую в публичный Интернет и, следовательно, могут использоваться с NAT:

- `10.0.0.0/8`.
- `172.16.0.0/12`.
- `192.168.0.0/16`.

При работе с правилами межсетевого экрана будьте *очень осторожны*. Некоторые конфигурации *могут заблокировать администратора* на сервере. Для безопасности рекомендуется выполнять первоначальную настройку межсетевого экрана с локальной консоли, а не удалённо через ssh.

34.3. PF

Начиная с FreeBSD 5.3, портированная версия межсетевого экрана PF из OpenBSD включена в базовую систему как её неотъемлемая часть. PF — это полноценный и многофункциональный межсетевой экран с опциональной поддержкой ALTQ (Alternate Queuing), которая обеспечивает качество обслуживания (QoS).

Проект OpenBSD содержит каноническое справочное издание по PF в FAQ по PF. Питер Ханстин ведёт подробный учебник по PF на http://home.nuug.no/~peter/pf/.

При чтении FAQ по PF учитывайте, что версия PF в FreeBSD значительно отличается от оригинальной версии OpenBSD за прошедшие годы. Не все функции работают одинаково в FreeBSD и OpenBSD, и наоборот.

`Список рассылки, посвящённый FreeBSD packet filter` — хорошее место для вопросов о настройке и работе PF межсетевого экрана. Перед тем как задать вопрос, проверьте архивы списка рассылки — возможно, ответ уже был дан.

Эта часть Руководства посвящена PF в контексте FreeBSD. В ней показано, как включить PF и ALTQ, а также приведены несколько примеров создания наборов правил в системе FreeBSD.

34.3.1. Включение PF

Для использования PF необходимо сначала загрузить его модуль ядра. В этом разделе описаны записи, которые можно добавить в /etc/rc.conf для включения PF.

Начните с добавления `pf_enable=yes` в /etc/rc.conf:

```
# sysrc pf_enable=yes
```

Дополнительные параметры, описанные в pfctl(8), могут быть переданы в PF при его запуске. Добавьте или измените эту запись в /etc/rc.conf и укажите необходимые флаги между кавычками (""):

```
pf_flags=""                   # additional flags for pfctl startup
```

PF не запустится, если не сможет найти файл конфигурации набора правил. По умолчанию FreeBSD не поставляется с готовым набором правил, и файл /etc/pf.conf отсутствует. Примеры наборов правил можно найти в /usr/share/examples/pf/. Если пользовательский набор правил был сохранён в другом месте, добавьте строку в /etc/rc.conf, указывающую полный путь к файлу:

```
pf_rules="/path/to/pf.conf"
```

Поддержка журналирования для PF предоставляется pflog(4). Для включения поддержки

журналирования добавьте `pflog_enable=yes` в /etc/rc.conf:

```
# sysrc pflog_enable=yes
```

Следующие строки также могут быть добавлены для изменения расположения файла журнала по умолчанию или для указания дополнительных флагов, передаваемых в pflog(4) при его запуске:

```
pflog_logfile="/var/log/pflog"  # where pflogd should store the logfile
pflog_flags=""                  # additional flags for pflogd startup
```

Наконец, если за межсетевым экраном есть локальная сеть (LAN) и пакеты необходимо перенаправлять для компьютеров в этой сети, или требуется NAT, включите следующую опцию:

```
gateway_enable="YES"            # Enable as LAN gateway
```

После сохранения необходимых изменений, PF можно запустить с поддержкой журналирования, набрав:

```
# service pf start
# service pflog start
```

По умолчанию PF читает свои правила конфигурации из /etc/pf.conf и изменяет, отбрасывает или пропускает пакеты в соответствии с правилами или определениями, указанными в этом файле. Установка FreeBSD включает несколько примеров файлов, расположенных в /usr/share/examples/pf/. Полное описание наборов правил PF можно найти в PF FAQ.

Для управления PF используйте `pfctl`. Полезные опции `pfctl` содержит сводку полезных опций этой команды. За подробным описанием всех доступных опций обратитесь к pfctl(8):

Таблица 51. Полезные параметры `pfctl`

Команда	Назначение
`pfctl -e`	Включить PF.
`pfctl -d`	Отключить PF.
`pfctl -F all -f /etc/pf.conf`	Очистить все правила NAT, фильтрации, состояний и таблиц и перезагрузить /etc/pf.conf.
`pfctl -s [ rules | nat | states ]`	Отчёт о правилах фильтрации, правилах NAT или таблице состояний.

Команда	Назначение
`pfctl -vnf /etc/pf.conf`	Проверить /etc/pf.conf на ошибки, но не загружать набор правил.

Пакет security/sudo полезен для выполнения команд, таких как `pfctl`, которые требуют повышенных привилегий. Он может быть установлен из Коллекции портов.

Для наблюдения за трафиком, проходящим через межсетевой экран PF, рекомендуется установить пакет sysutils/pftop или порт. После установки можно запустить pftop для просмотра текущего снимка трафика в формате, аналогичном top(1).

34.3.2. Наборы правил PF

Этот раздел демонстрирует, как создать настраиваемый набор правил. Он начинается с простейшего набора правил и развивает его концепции, используя несколько примеров для демонстрации практического применения множества возможностей PF.

Самый простой возможный набор правил предназначен для одиночной машины, которая не запускает никаких сервисов и которой требуется доступ к одной сети, возможно, к Интернету. Чтобы создать этот минимальный набор правил, отредактируйте файл /etc/pf.conf, чтобы он выглядел следующим образом:

```
block in all
pass out all keep state
```

Первое правило по умолчанию запрещает весь входящий трафик. Второе правило разрешает исходящие соединения, созданные этой системой, сохраняя информацию о состоянии этих соединений. Эта информация о состоянии позволяет возвратному трафику для этих соединений проходить обратно и должна использоваться только на машинах, которым можно доверять. Набор правил можно загрузить с помощью:

```
# pfctl -e ; pfctl -f /etc/pf.conf
```

В дополнение к отслеживанию состояний, PF предоставляет *списки* и *макросы*, которые можно определить для использования при создании правил. Макросы могут включать списки и должны быть определены до их использования. В качестве примера вставьте следующие строки в самое начало набора правил:

```
tcp_services = "{ ssh, smtp, domain, www, pop3, auth, pop3s }"
udp_services = "{ domain }"
```

PF понимает как имена портов, так и их номера, при условии, что имена перечислены в /etc/services. В этом примере создаются два макроса. Первый — это список из семи имён TCP-портов, а второй — одно имя UDP-порта. После определения макросы можно использовать в

правилах. В данном примере весь трафик блокируется, за исключением соединений, инициированных этой системой для семи указанных TCP-сервисов и одного указанного UDP-сервиса:

```
tcp_services = "{ ssh, smtp, domain, www, pop3, auth, pop3s }"
udp_services = "{ domain }"
block all
pass out proto tcp to any port $tcp_services keep state
pass proto udp to any port $udp_services keep state
```

Хотя UDP считается протоколом без сохранения состояния, PF способен отслеживать некоторую информацию о состоянии. Например, когда передаётся UDP-запрос к серверу имён с вопросом о доменном имени, PF будет ожидать ответ, чтобы пропустить его обратно.

Всякий раз при внесении изменений в набор правил, новые правила должны быть загружены для их использования:

```
# pfctl -f /etc/pf.conf
```

Если нет синтаксических ошибок, `pfctl` не выводит сообщений во время загрузки правил. Правила также можно проверить перед попыткой их загрузки:

```
# pfctl -nf /etc/pf.conf
```

Включение опции `-n` приводит к тому, что правила только интерпретируются, но не загружаются. Это даёт возможность исправить любые ошибки. В любое время будет применяться последний загруженный корректный набор правил, пока либо PF не будет отключён, либо не будет загружен новый набор правил.

Добавление `-v` к проверке или загрузке набора правил `pfctl` отобразит полностью разобранные правила именно так, как они будут загружены. Это крайне полезно при отладке правил межсетевого экрана.

Простой шлюз с NAT

В этом разделе показано, как настроить систему FreeBSD с PF для работы в качестве шлюза как минимум для одного другого компьютера. Шлюзу требуется как минимум два сетевых интерфейса, каждый из которых подключен к отдельной сети. В этом примере xl0 подключен к Интернету, а xl1 подключен к внутренней сети.

Включите шлюз, чтобы позволить машине перенаправлять сетевой трафик, полученный на одном интерфейсе, на другой интерфейс. Этот параметр sysctl перенаправляет IPv4-пакеты:

```
# sysctl net.inet.ip.forwarding=1
```

Для переадресации IPv6-трафика используйте:

```
# sysctl net.inet6.ip6.forwarding=1
```

Чтобы включить эти настройки при загрузке системы, используйте sysrc(8) для их добавления в /etc/rc.conf:

```
# sysrc gateway_enable=yes
# sysrc ipv6_gateway_enable=yes
```

Проверьте с помощью `ifconfig`, что оба интерфейса активны и работают.

Далее создайте правила PF, чтобы позволить шлюзу передавать трафик. Хотя следующее правило разрешает трафик с сохранением статуса от хостов внутренней сети проходить к шлюзу, ключевое слово `to` не гарантирует прохождение всего пути от источника до назначения:

```
pass in on xl1 from xl1:network to xl0:network port $ports keep state
```

Это правило разрешает прохождение трафика только внутрь шлюза на внутреннем интерфейсе. Чтобы пакеты могли идти дальше, требуется соответствующее правило:

```
pass out on xl0 from xl1:network to xl0:network port $ports keep state
```

Хотя эти два правила будут работать, столь специфичные правила редко требуются. Для занятого сетевого администратора читаемый набор правил — это более безопасный набор правил. В оставшейся части этого раздела показано, как сохранять правила максимально простыми для удобочитаемости. Например, эти два правила можно заменить одним:

```
pass from xl1:network to any port $ports keep state
```

Обозначение `interface:network` может быть заменено макросом для повышения читаемости набора правил. Например, можно определить макрос `$localnet` как сеть, непосредственно подключённую к внутреннему интерфейсу (`$xl1:network`). Или определение `$localnet` может быть изменено на *IP-адрес/маску сети* для обозначения сети, например `192.168.100.1/24` для подсети частных адресов.

Если требуется, `$localnet` можно определить даже как список сетей. Независимо от конкретных потребностей, разумное определение `$localnet` может быть использовано в типичном правиле пропуска следующим образом:

```
pass from $localnet to any port $ports keep state
```

Следующий пример набора правил разрешает весь трафик, инициированный машинами во внутренней сети. Сначала определяются два макроса для представления внешнего и внутреннего интерфейсов 3COM настраиваемого шлюза.

Для пользователей коммутируемого доступа внешний интерфейс будет использовать tun0. Для ADSL-подключения, особенно тех, которые используют PPP через Ethernet (PPPoE), правильный внешний интерфейс — это tun0, а не физический Ethernet-интерфейс.

```
ext_if = "xl0"  # macro for external interface - use tun0 for PPPoE
int_if = "xl1"  # macro for internal interface
localnet = $int_if:network
# ext_if IP address could be dynamic, hence ($ext_if)
nat on $ext_if from $localnet to any -> ($ext_if)
block all
pass from { lo0, $localnet } to any keep state
```

Этот набор правил вводит правило `nat`, которое используется для обработки преобразования сетевых адресов из не маршрутизируемых адресов внутри внутренней сети в IP-адрес, назначенный внешнему интерфейсу. Скобки вокруг последней части правила nat `($ext_if)` включаются, когда IP-адрес внешнего интерфейса назначается динамически. Это гарантирует, что сетевой трафик будет работать без серьёзных перебоев, даже если внешний IP-адрес изменится.

Обратите внимание, что этот набор правил, вероятно, разрешает больше трафика для выхода из сети, чем необходимо. Один из разумных вариантов настройки может создать этот макрос:

```
client_out = "{ ftp-data, ftp, ssh, domain, pop3, auth, nntp, http, \
    https, cvspserver, 2628, 5999, 8000, 8080 }"
```

для использования в основном правиле пропуска:

```
pass inet proto tcp from $localnet to any port $client_out \
    flags S/SA keep state
```

Еще могут понадобиться несколько других правил пропуска . Это правило включает SSH на внешнем интерфейсе:

```
pass in inet proto tcp to $ext_if port ssh
```

Это определение макроса и правило разрешают DNS и NTP для внутренних клиентов:

```
udp_services = "{ domain, ntp }"
```

```
pass quick inet proto { tcp, udp } to any port $udp_services keep state
```

Обратите внимание на ключевое слово `quick` в этом правиле. Поскольку набор правил состоит из нескольких правил, важно понимать взаимосвязи между ними. Правила обрабатываются сверху вниз, в порядке их написания. Для каждого пакета или соединения, оцениваемого PF, *последнее совпадающее правило* в наборе является тем, которое применяется. Однако, когда пакет совпадает с правилом, содержащим ключевое слово `quick`, обработка правил прекращается, и пакет обрабатывается в соответствии с этим правилом. Это очень полезно, когда требуется исключение из общих правил.

Создание FTP-прокси

Настройка рабочих правил FTP может быть проблематичной из-за особенностей протокола FTP. Протокол FTP появился на несколько десятилетий раньше межсетевых экранов и небезопасен по своей конструкции. Наиболее распространённые аргументы против использования FTP включают:

- Пароли передаются в открытом виде.
- Протокол требует использования как минимум двух TCP-соединений (управляющего канала и канала данных) на отдельных портах.
- Когда сеанс установлен, данные передаются с использованием случайно выбранных портов.

Все эти моменты представляют собой проблемы безопасности, даже без учёта потенциальных уязвимостей в клиентском или серверном программном обеспечении. Более безопасные альтернативы для передачи файлов существуют, например, sftp(1) или scp(1), которые обеспечивают аутентификацию и передачу данных через зашифрованные соединения.

Для случаев, когда требуется FTP, PF предоставляет возможность перенаправления FTP-трафика на небольшую прокси-программу под названием ftp-proxy(8), которая включена в базовую систему FreeBSD. Роль прокси заключается в динамическом добавлении и удалении правил в наборе правил, используя набор якорей, для корректной обработки FTP-трафика.

Чтобы включить FTP-прокси, добавьте следующую строку в /etc/rc.conf:

```
ftpproxy_enable="YES"
```

Затем запустите прокси, выполнив:

```
# service ftp-proxy start
```

Для базовой конфигурации необходимо добавить три элемента в /etc/pf.conf. Во-первых, якоря, которые прокси будет использовать для вставки правил, генерируемых для FTP-сессий:

```
nat-anchor "ftp-proxy/*"
rdr-anchor "ftp-proxy/*"
```

Во-вторых, необходимо правило pass, чтобы разрешить FTP-трафик к прокси.

Третье, правила перенаправления и NAT должны быть определены до правил фильтрации. Вставьте это правило `rdr` сразу после правила `nat`:

```
rdr pass on $int_if proto tcp from any to any port ftp -> 127.0.0.1 port 8021
```

Наконец, разрешите перенаправленному трафику проходить:

```
pass out proto tcp from $proxy to any port ftp
```

где `$proxy` раскрывается в адрес, к которому привязан демон прокси.

Сохраните /etc/pf.conf, загрузите новые правила и проверьте с клиента, что FTP-подключения работают:

```
# pfctl -f /etc/pf.conf
```

Этот пример описывает базовую настройку, когда клиенты в локальной сети должны обращаться к FTP-серверам в других местах. Данная базовая конфигурация должна хорошо работать с большинством комбинаций FTP-клиентов и серверов. Как показано в ftp-proxy(8), поведение прокси можно изменить различными способами, добавляя параметры в строку `ftpproxy_flags=`. Некоторые клиенты или серверы могут иметь специфические особенности, которые необходимо учитывать в конфигурации, или может возникнуть необходимость интегрировать прокси определённым образом, например, назначить FTP-трафик в определённую очередь.

Для способов запуска FTP-сервера, защищённого PF и ftp-proxy(8), настройте отдельный `ftp-proxy` в обратном режиме с использованием `-R` на отдельном порту с собственным правилом перенаправления pass.

Управление ICMP

Многие инструменты, используемые для отладки или устранения неполадок в сети TCP/IP, основаны на протоколе ICMP (Internet Control Message Protocol), который был специально разработан для целей отладки.

Протокол ICMP отправляет и получает *управляющие сообщения* между хостами и шлюзами, в основном для предоставления отправителю обратной связи о любых необычных или сложных условиях на пути к целевому хосту. Маршрутизаторы используют ICMP для согласования размеров пакетов и других параметров передачи в процессе, часто называемом *обнаружением MTU пути*.

С точки зрения межсетевого экрана, некоторые управляющие ICMP-сообщения уязвимы к известным векторам атак. Кроме того, безусловный пропуск всего диагностического трафика упрощает отладку, но также облегчает для других получение информации о сети. По этим причинам следующее правило может быть неоптимальным:

```
pass inet proto icmp from any to any
```

Одно из решений — пропускать весь ICMP-трафик из локальной сети, блокируя все запросы извне:

```
pass inet proto icmp from $localnet to any keep state
pass inet proto icmp from any to $ext_if keep state
```

Дополнительные параметры демонстрируют гибкость PF. Например, вместо разрешения всех ICMP-сообщений можно указать сообщения, используемые ping(8) и traceroute(8). Начните с определения макроса для этого типа сообщения:

```
icmp_types = "echoreq"
```

и правила, которое использует макрос:

```
pass inet proto icmp all icmp-type $icmp_types keep state
```

Если требуются другие типы ICMP-пакетов, расширьте `icmp_types` до списка этих типов пакетов. Введите `more /usr/src/sbin/pfctl/pfctl_parser.c`, чтобы увидеть список типов ICMP-сообщений, поддерживаемых PF. Дополнительные сведения о каждом типе сообщения можно найти по ссылке http://www.iana.org/assignments/icmp-parameters/icmp-parameters.xhtml.

Поскольку Unix `traceroute` по умолчанию использует UDP, требуется дополнительное правило для разрешения Unix `traceroute`:

```
# allow out the default range for traceroute(8):
pass out on $ext_if inet proto udp from any to any port 33433 >< 33626 keep state
```

Поскольку `TRACERT.EXE` в системах Microsoft Windows использует сообщения запроса эхо-ответа ICMP, достаточно только первого правила, чтобы разрешить трассировку сети с этих систем. В Unix `traceroute` можно настроить на использование других протоколов, и он будет использовать сообщения запроса эхо-ответа ICMP, если указан параметр `-I`. Подробности смотрите на traceroute(8).

Обнаружение MTU пути

Протоколы Интернета разработаны так, чтобы быть независимыми от устройств, и одним

из следствий этой независимости является то, что оптимальный размер пакета для данного соединения не всегда может быть надёжно предсказан. Основное ограничение на размер пакета — это *Maximum Transmission Unit* (MTU), который устанавливает верхний предел размера пакета для интерфейса. Введите `ifconfig`, чтобы просмотреть MTU для сетевых интерфейсов системы.

TCP/IP использует процесс, известный как обнаружение MTU пути, для определения правильного размера пакета для соединения. Этот процесс отправляет пакеты различного размера с установленным флагом "Не фрагментировать", ожидая возврата ICMP-пакета с "типом 3, кодом 4", когда достигнут верхний предел. Тип 3 означает "адресат недостижим", а код 4 является сокращением для "требуется фрагментация, но установлен флаг 'не фрагментировать'". Чтобы разрешить обнаружение MTU пути для поддержки соединений с другими MTU, добавьте тип `destination unreachable` в макрос `icmp_types`:

```
icmp_types = "{ echoreq, unreach }"
```

Поскольку правило `pass` уже использует этот макрос, его не нужно изменять для поддержки нового типа ICMP:

```
pass inet proto icmp all icmp-type $icmp_types keep state
```

PF позволяет фильтрацию по всем вариантам типов и кодов ICMP. Список возможных типов и кодов документирован в icmp(4) и icmp6(4).

Использование таблиц

Некоторые типы данных важны для фильтрации и перенаправления в определённый момент времени, но их определение слишком длинное, чтобы включать его в файл набора правил. PF поддерживает использование таблиц — это определённые списки, которыми можно управлять без необходимости перезагружать весь набор правил, и которые обеспечивают быстрый поиск. Имена таблиц всегда заключаются в `< >`, например:

```
table <clients> { 192.168.2.0/24, !192.168.2.5 }
```

В этом примере сеть `192.168.2.0/24` является частью таблицы, за исключением адреса `192.168.2.5`, который исключен с помощью оператора `!`. Также можно загружать таблицы из файлов, где каждый элемент находится на отдельной строке, как показано в этом примере /etc/clients:

```
192.168.2.0/24
!192.168.2.5
```

Для обращения к файлу определите таблицу следующим образом:

```
table <clients> persist file "/etc/clients"
```

После определения таблицы на неё можно ссылаться в правиле:

```
pass inet proto tcp from <clients> to any port $client_out flags S/SA keep state
```

Содержимое таблицы можно изменять в реальном времени с помощью `pfctl`. В этом примере добавляется ещё одна сеть в таблицу:

```
# pfctl -t clients -T add 192.168.1.0/16
```

Обратите внимание, что любые изменения, внесённые таким образом, вступят в силу немедленно, что делает их идеальными для тестирования, но они не сохранятся после отключения питания или перезагрузки. Чтобы сделать изменения постоянными, необходимо изменить определение таблицы в наборе правил или отредактировать файл, на который ссылается таблица. Можно поддерживать копию таблицы на диске с помощью задания cron(8), которое периодически сохраняет содержимое таблицы на диск, используя команду вида `pfctl -t clients -T show >/etc/clients`. Или /etc/clients можно обновить содержимым таблицы из памяти:

```
# pfctl -t clients -T replace -f /etc/clients
```

Использование таблиц превышения нагрузки для защиты SSH

Те, кто запускает SSH на внешнем интерфейсе, вероятно, видели что-то подобное в журналах аутентификации:

```
Sep 26 03:12:34 skapet sshd[25771]: Failed password for root from 200.72.41.31 port
40992 ssh2
Sep 26 03:12:34 skapet sshd[5279]: Failed password for root from 200.72.41.31 port
40992 ssh2
Sep 26 03:12:35 skapet sshd[5279]: Received disconnect from 200.72.41.31: 11: Bye Bye
Sep 26 03:12:44 skapet sshd[29635]: Invalid user admin from 200.72.41.31
Sep 26 03:12:44 skapet sshd[24703]: input_userauth_request: invalid user admin
Sep 26 03:12:44 skapet sshd[24703]: Failed password for invalid user admin from
200.72.41.31 port 41484 ssh2
```

Это указывает на атаку методом грубой силы, когда кто-то или какая-то программа пытается подобрать имя пользователя и пароль, чтобы получить доступ к системе.

Если необходим внешний доступ по SSH для законных пользователей, изменение порта по умолчанию, используемого SSH, может обеспечить некоторую защиту. Однако PF предоставляет более элегантное решение. Правила `pass` могут содержать ограничения на действия подключающихся хостов, а нарушители могут быть заблокированы в таблице

адресов, которым запрещён частичный или полный доступ. Также возможно разорвать все существующие соединения с машинами, которые превышают установленные ограничения.

Для настройки создайте следующую таблицу в разделе tables набора правил:

```
table <bruteforce> persist
```

Затем, где-то в начале набора правил добавьте правила для блокировки перебора, разрешая при этом законный доступ:

```
block quick from <bruteforce>
pass inet proto tcp from any to $localnet port $tcp_services \
    flags S/SA keep state \
    (max-src-conn 100, max-src-conn-rate 15/5, \
    overload <bruteforce> flush global)
```

Часть в скобках определяет ограничения, и числа должны быть изменены в соответствии с местными требованиями. Это можно прочитать следующим образом:

`max-src-conn` — это количество одновременных подключений, разрешённых с одного хоста.

`max-src-conn-rate` — это частота новых соединений, разрешённых с любого отдельного хоста (*15*) за указанное количество секунд (*5*).

`overload <bruteforce>` означает, что любой хост, превышающий эти ограничения, добавляет свой адрес в таблицу `bruteforce`. Набор правил блокирует весь трафик с адресов, находящихся в таблице `bruteforce`.

Наконец, `flush global` означает, что когда хост достигает лимита, все (`global`) соединения этого хоста будут разорваны (`flush`).

Эти правила *не* заблокируют медленные брутфорсеры, как описано в http://home.nuug.no/~peter/hailmary2013/.

Этот пример набора правил предназначен в основном для иллюстрации. Например, если требуется разрешить большое количество соединений в целом, но необходимо быть более строгим в отношении ssh, дополните приведённое выше правило чем-то вроде следующего, в начале набора правил:

```
pass quick proto { tcp, udp } from any to any port ssh \
    flags S/SA keep state \
    (max-src-conn 15, max-src-conn-rate 5/3, \
    overload <bruteforce> flush global)
```

Не всегда необходимо блокировать всех превышающих нагрузки:

Стоит отметить, что механизм превышения нагрузки — это общий метод,

> который применяется не только к SSH, и не всегда оптимально полностью блокировать весь трафик от нарушителей.
>
> Например, правило превышения нагрузки может использоваться для защиты почтовой службы или веб-сервиса, а таблица превышения нагрузки может применяться в правиле для назначения нарушителей в очередь с минимальным выделением пропускной способности или для перенаправления на определённую веб-страницу.

Со временем таблицы будут заполняться правилами превышения нагрузки, и их размер будет постепенно увеличиваться, занимая больше памяти. Иногда заблокированный IP-адрес оказывается динамически назначенным, который с тех пор был выделен хосту, имеющему законные основания для взаимодействия с хостами в локальной сети.

Для подобных ситуаций `pfctl` предоставляет возможность удалять записи из таблиц. Например, следующая команда удалит записи из таблицы `<bruteforce>`, к которым не было обращений в течение `86400` секунд:

```
# pfctl -t bruteforce -T expire 86400
```

Аналогичную функциональность предоставляет пакет security/expiretable, который удаляет записи таблицы, к которым не было обращений в течение указанного периода времени.

После установки `expiretable` можно запустить для удаления записей из таблицы `<bruteforce>`, которые старше указанного времени. В этом примере удаляются все записи старше 24 часов:

```
/usr/local/sbin/expiretable -v -d -t 24h bruteforce
```

Защита от SPAM

mail/spamd можно настроить совместно с PF для обеспечения внешней защиты от SPAM (Не путайте его с демоном spamd, который поставляется вместе с spamassassin). Этот spamd интегрируется в конфигурацию PF с помощью набора перенаправлений.

Спамеры обычно рассылают большое количество сообщений, и СПАМ в основном отправляется из нескольких дружественных к спамерам сетей и множества взломанных машин, которые быстро попадают в *списки блокировки*.

Когда SMTP-соединение с адреса из блок-листа получено, spamd отображает свой баннер и немедленно переключается в режим, где он отвечает на SMTP-трафик по одному байту за раз. Эта техника, предназначенная для максимально возможной потери времени на стороне спамера, называется *тарпиттинг* (tarpitting, намеренное замедление). Конкретная реализация, использующая однобайтовые SMTP-ответы, часто упоминается как *прерывистые ответы* (stuttering, метод заикания).

В этом примере показана базовая процедура настройки spamd с автоматически

обновляемыми списками блокировки. Для получения дополнительной информации обратитесь к руководствам, которые устанавливаются вместе с пакетом mail/spamd.

Процедура: Настройка spamd

1. Установите пакет mail/spamd или порт. Для использования функции серого списка (greylisting) в spamd, необходимо подключить fdescfs(5) в /dev/fd. Добавьте следующую строку в /etc/fstab:

   ```
   fdescfs /dev/fd fdescfs rw 0 0
   ```

   Затем смонтируйте файловую систему:

   ```
   #  mount fdescfs
   ```

2. Далее отредактируйте набор правил PF, включив:

   ```
   table <spamd> persist
   table <spamd-white> persist
   rdr pass on $ext_if inet proto tcp from <spamd> to \
       { $ext_if, $localnet } port smtp -> 127.0.0.1 port 8025
   rdr pass on $ext_if inet proto tcp from !<spamd-white> to \
       { $ext_if, $localnet } port smtp -> 127.0.0.1 port 8025
   ```

   Две таблицы `<spamd>` и `<spamd-white>` являются обязательными. SMTP-трафик с адреса, указанного в `<spamd>`, но отсутствующего в `<spamd-white>`, перенаправляется к демону spamd, ожидающему на порту 8025.

3. Следующий шаг — настроить spamd в /usr/local/etc/spamd.conf и добавить параметры в rc.conf.

   Установка пакета mail/spamd включает образец файла конфигурации (/usr/local/etc/spamd.conf.sample) и man-страницу для spamd.conf. Обратитесь к ним для получения дополнительных параметров конфигурации, помимо показанных в этом примере.

   Одна из первых строк в файле конфигурации, которая не начинается с символа комментария `#`, содержит блок, определяющий список `all`, который указывает списки для использования:

   ```
   all:\
       :traplist:allowlist:
   ```

   Эта запись добавляет желаемые списки блокировок, разделённые двоеточиями (`:`). Чтобы использовать список разрешений для исключения адресов из списка блокировок, добавьте имя списка разрешений *непосредственно* после имени этого

списка блокировок. Например: `:blocklist:allowlist:`.

За этим следует определение указанного списка блокировки:

```
traplist:\
    :black:\
    :msg="SPAM. Your address %A has sent spam within the last 24 hours":\
    :method=http:\
    :file=www.openbsd.org/spamd/traplist.gz
```

где первая строка — это имя списка блокировки, а вторая строка определяет тип списка. Поле `msg` содержит сообщение, которое будет отображаться заблокированным отправителям во время SMTP-диалога. Поле `method` указывает, каким образом spamd-setup получает данные списка; поддерживаемые методы — `http`, `ftp`, из `file` в смонтированной файловой системе и через `exec` внешней программы. Наконец, поле `file` определяет имя файла, который ожидает получить spamd.

Определение указанного разрешенного списка аналогично, но опускает поле `msg`, так как сообщение не требуется:

```
allowlist:\
    :white:\
    :method=file:\
    :file=/var/mail/allowlist.txt
```

Выбирайте источники данных с осторожностью:

Использование всех блокирующих списков в примере spamd.conf приведет к блокировке больших сегментов Интернета. Администраторам необходимо отредактировать файл, чтобы создать оптимальную конфигурацию, которая использует подходящие источники данных и, при необходимости, пользовательские списки.

Далее добавьте эту запись в /etc/rc.conf. Дополнительные флаги описаны в man-странице, указанной в комментарии:

```
spamd_flags="-v" # use "" and see spamd-setup(8) for flags
```

После завершения перезагрузите набор правил, запустите spamd командой `service obspamd start` и завершите настройку с помощью `spamd-setup`. Наконец, создайте задание в cron(8), которое будет вызывать `spamd-setup` для обновления таблиц через разумные промежутки времени.

На типичном шлюзе перед почтовым сервером узлы начнут попадать в ловушку в течение от нескольких секунд до нескольких минут.

PF также поддерживает *greylisting*, который временно отклоняет сообщения от неизвестных хостов с кодами *45n*. Сообщения от хостов, находящихся в greylisting, которые повторяют попытку в разумные сроки, пропускаются. Трафик от отправителей, настроенных на работу в пределах, установленных RFC 1123 и RFC 2821, пропускается сразу.

Дополнительная информация о методе серых списков (greylisting) доступна на сайте greylisting.org. Самое удивительное в серых списках, помимо их простоты, это то, что они до сих пор работают. Спамеры и авторы вредоносного ПО очень медленно адаптируются, чтобы обойти этот метод.

Основная процедура настройки серого списка выглядит следующим образом:

Процедура: Настройка серого списка

1. Убедитесь, что fdescfs(5) смонтирован, как описано в Шаге 1 предыдущей процедуры.
2. Для запуска spamd в режиме серого списка добавьте следующую строку в /etc/rc.conf:

   ```
   spamd_grey="YES"  # use spamd greylisting if YES
   ```

   Обратитесь к man-странице spamd для описания дополнительных параметров, относящихся к серым спискам.

3. Для завершения настройки серого списка:

   ```
   #  service obspamd restart
   #  service obspamlogd start
   ```

За кулисами инструмент базы данных spamdb и обновляющий белый список spamlogd выполняют важные функции для механизма отложенной доставки. spamdb — это основной интерфейс администратора для управления списками блокировки, серого списка и списка с разрешениями через содержимое базы данных /var/db/spamdb.

Сетевая гигиена

В этом разделе описывается, как `block-policy`, `scrub` и `antispoof` могут быть использованы для обеспечения разумного поведения набора правил.

`block-policy` — это опция, которую можно установить в разделе `options` набора правил, предшествующем правилам перенаправления и фильтрации. Эта опция определяет, какую обратную связь (если таковая имеется) PF отправляет хостам, заблокированным правилом. Опция имеет два возможных значения: `drop` отбрасывает заблокированные пакеты без ответа, а `return` возвращает код состояния, например `Connection refused`.

Если не задано, политика по умолчанию — `drop`. Чтобы изменить `block-policy`, укажите желаемое значение:

```
set block-policy return
```

В PF `scrub` — это ключевое слово, которое включает нормализацию сетевых пакетов. Этот процесс пересобирает фрагментированные пакеты и отбрасывает TCP-пакеты с недопустимыми комбинациями флагов. Включение `scrub` обеспечивает защиту от определённых видов атак, основанных на некорректной обработке фрагментов пакетов. Доступно несколько опций, но простейшая форма подходит для большинства конфигураций:

```
scrub in all
```

Некоторые службы, такие как NFS, требуют специальных параметров обработки фрагментов. Дополнительную информацию можно найти по ссылке https://home.nuug.no/}peter/pf/en/scrub.html.

Этот пример собирает фрагменты, сбрасывает бит "не фрагментировать" и устанавливает максимальный размер сегмента в 1440 байт:

```
scrub in all fragment reassemble no-df max-mss 1440
```

Механизм `antispoof` защищает от активности с подделанных или фальсифицированных IP-адресов, в основном блокируя пакеты, появляющиеся на интерфейсах и в направлениях, которые логически невозможны.

Эти правила отсеивают поддельный трафик, поступающий из остального мира, а также любые поддельные пакеты, исходящие из локальной сети:

```
antispoof for $ext_if
antispoof for $int_if
```

Обработка немаршрутизируемых адресов

Даже при правильно настроенном шлюзе для обработки преобразования сетевых адресов может потребоваться компенсировать ошибки в конфигурации, сделанные другими людьми. Распространённой ошибкой является пропуск трафика с не маршрутизируемыми адресами в Интернет. Поскольку трафик с не маршрутизируемых адресов может использоваться в нескольких методах DoS-атак, рекомендуется явно блокировать такой трафик от попадания в сеть через внешний интерфейс.

В этом примере определяется макрос, содержащий не маршрутизируемые адреса, который затем используется в правилах блокировки. Трафик на эти адреса и с этих адресов тихо отбрасывается на внешнем интерфейсе шлюза.

```
martians = "{ 127.0.0.0/8, 192.168.0.0/16, 172.16.0.0/12, \
          10.0.0.0/8, 169.254.0.0/16, 192.0.2.0/24, \
          0.0.0.0/8, 240.0.0.0/4 }"

block drop in quick on $ext_if from $martians to any
block drop out quick on $ext_if from any to $martians
```

34.3.3. Включение ALTQ

На FreeBSD ALTQ можно использовать с PF для обеспечения качества обслуживания (QOS — Quality of Service). После включения ALTQ в наборе правил можно определить очереди, которые определяют приоритет обработки исходящих пакетов.

Прежде чем включить ALTQ, обратитесь к altq(4), чтобы определить, поддерживают ли его драйверы сетевых карт, установленных в системе.

ALTQ недоступен в виде загружаемого модуля ядра. Если интерфейсы системы поддерживают ALTQ, создайте собственное ядро, следуя инструкциям в Настройка ядра FreeBSD. Доступны следующие параметры ядра. Первый необходим для включения ALTQ. Хотя бы один из остальных параметров требуется для указания алгоритма планирования очередей:

```
options         ALTQ
options         ALTQ_CBQ        # Class Based Queuing (CBQ)
options         ALTQ_RED        # Random Early Detection (RED)
options         ALTQ_RIO        # RED In/Out
options         ALTQ_HFSC       # Hierarchical Packet Scheduler (HFSC)
options         ALTQ_PRIQ       # Priority Queuing (PRIQ)
```

Доступны следующие алгоритмы планировщика:

CBQ

Очереди на основе классов (CBQ — Class Based Queuing) используется для разделения пропускной способности соединения на различные классы или очереди с целью приоритизации трафика на основе правил фильтрации.

RED

Случайное раннее обнаружение (RED — Random Early Detection) используется для предотвращения перегрузки сети путем измерения длины очереди и сравнения её с минимальным и максимальным порогами для очереди. Когда очередь превышает максимальный порог, все новые пакеты случайным образом отбрасываются.

RIO

В режиме Случайного раннего обнаружения для входящего и исходящего трафика (RIO) RED поддерживает несколько средних длин очередей и несколько пороговых значений, по одному для каждого уровня QOS.

HFSC

Иерархический планировщик пакетов с гарантированной справедливой кривой обслуживания (HFSC) описан на http://www-2.cs.cmu.edu/~hzhang/HFSC/main.html.

PRIQ

Очереди с приоритетом (PRIQ) всегда пропускают трафик из более высокоуровневой очереди первым.

Дополнительная информация о алгоритмах планирования и примеры наборов правил доступны на архивированной странице OpenBSD.

34.4. IPFW

IPFW — это межсетевой экран с отслеживанием состояний, разработанный для FreeBSD, который поддерживает как IPv4, так и IPv6. Он состоит из нескольких компонентов: процессора правил фильтрации межсетевого экрана в ядре и встроенного механизма учёта пакетов, механизма журналирования, NAT, механизма управления трафиком dummynet(4), механизма переадресации, механизма моста и механизма ipstealth.

FreeBSD предоставляет пример набора правил в /etc/rc.firewall, который определяет несколько типов межсетевых экранов для распространённых сценариев, чтобы помочь новичкам в создании подходящего набора правил. IPFW предлагает мощный синтаксис, с помощью которого опытные пользователи могут создавать настраиваемые наборы правил, соответствующие требованиям безопасности заданного окружения.

Этот раздел описывает, как включить IPFW, предоставляет обзор синтаксиса его правил и демонстрирует несколько наборов правил для распространённых сценариев настройки.

34.4.1. Включение IPFW

IPFW включён в базовую установку FreeBSD в виде загружаемого модуля ядра, что означает, что для его включения не требуется собирать собственное ядро.

Для пользователей, которые хотят статически скомпилировать поддержку IPFW в собственном ядре, см. Параметры ядра IPFW.

Для настройки системы для включения IPFW при загрузке добавьте `firewall_enable="YES"` в файл /etc/rc.conf:

```
# sysrc firewall_enable="YES"
```

Чтобы использовать один из типов межсетевых экранов, предоставляемых FreeBSD, добавьте строку с указанием типа:

```
# sysrc firewall_type="open"
```

Доступные типы:

- `open`: пропускает весь трафик.
- `client`: защищает только эту машину.
- `simple`: защищает всю сеть.
- `closed`: полностью отключает IP-трафик, за исключением интерфейса loopback.
- `workstation`: защищает только эту машину с использованием правил с отслеживанием состояния.
- `UNKNOWN`: отключает загрузку правил межсетевого экрана.
- filename: полный путь к файлу, содержащему набор правил межсетевого экрана.

Если `firewall_type` установлен в значение `client` или `simple`, измените правила по умолчанию, указанные в /etc/rc.firewall, чтобы они соответствовали конфигурации системы.

Обратите внимание, что тип `filename` используется для загрузки пользовательского набора правил.

Альтернативный способ загрузки пользовательского набора правил — присвойте переменной `firewall_script` значение, равное абсолютному пути к *исполняемому скрипту*, который включает команды IPFW. Примеры, используемые в этом разделе, предполагают, что `firewall_script` установлен в /etc/ipfw.rules:

```
# sysrc firewall_script="/etc/ipfw.rules"
```

Чтобы включить ведение журнала через syslogd(8), добавьте следующую строку:

```
# sysrc firewall_logging="YES"
```

В журнал будут записываться только правила межсетевого экрана с опцией `log`. Правила по умолчанию не включают эту опцию, и её необходимо добавить вручную. Поэтому рекомендуется отредактировать набор правил по умолчанию для ведения журнала. Кроме того, может потребоваться ротация журналов, если они сохраняются в отдельный файл.

В файле /etc/rc.conf нет переменной для установки ограничений журналирования. Чтобы ограничить количество записей в журнале для каждого правила на одну попытку соединения, укажите число с помощью следующей строки в /etc/sysctl.conf:

```
# echo "net.inet.ip.fw.verbose_limit=5" >> /etc/sysctl.conf
```

Для включения журналирования через выделенный интерфейс с именем `ipfw0`, добавьте следующую строку в /etc/rc.conf вместо этого:

```
# sysrc firewall_logif="YES"
```

Затем используйте tcpdump для просмотра записываемых данных:

```
# tcpdump -t -n -i ipfw0
```

Нет накладных расходов из-за ведения журнала, если не подключен tcpdump.

После сохранения необходимых изменений запустите межсетевой экран. Чтобы сразу включить ограничения журналирования, также установите указанное выше значение `sysctl`:

```
# service ipfw start
# sysctl net.inet.ip.fw.verbose_limit=5
```

34.4.2. Синтаксис правил IPFW

Когда пакет попадает в межсетевой экран IPFW, он сравнивается с первым правилом в наборе правил и последовательно обрабатывается сверху вниз, правило за правилом. Если пакет соответствует параметрам выбора правила, выполняется действие этого правила, и поиск в наборе правил для данного пакета прекращается. Это называется «первое совпадение побеждает». Если пакет не соответствует ни одному из правил, он попадает под действие обязательного правила IPFW с номером 65535, которое запрещает все пакеты и тихо их отбрасывает. Однако, если пакет соответствует правилу, содержащему ключевые слова `count`, `skipto` или `tee`, поиск продолжается. Подробнее о том, как эти ключевые слова влияют на обработку правил, см. в ipfw(8).

При создании правила IPFW ключевые слова должны быть записаны в следующем порядке. Некоторые ключевые слова являются обязательными, а другие — опциональными. Слова, написанные в верхнем регистре, обозначают переменную, а слова в нижнем регистре должны предшествовать следующей за ними переменной. Символ `#` используется для обозначения начала комментария и может находиться в конце правила или на отдельной строке. Пустые строки игнорируются.

```
CMD RULE_NUMBER set SET_NUMBER ACTION log LOG_AMOUNT PROTO from SRC SRC_PORT to DST DST_PORT
OPTIONS
```

В этом разделе представлен обзор этих ключевых слов и их параметров. Это не исчерпывающий список всех возможных вариантов. Для полного описания синтаксиса правил, которые можно использовать при создании правил IPFW, обратитесь к ipfw(8).

CMD

Каждое правило должно начинаться с `ipfw add`.

RULE_NUMBER

Каждое правило связано с числом от `1` до `65534`. Этот номер используется для указания порядка обработки правил. Несколько правил могут иметь одинаковый номер, в таком случае они применяются в порядке их добавления.

SET_NUMBER

Каждое правило связано с номером набора от `0` до `31`. Наборы можно отключать или включать по отдельности, что позволяет быстро добавлять или удалять набор правил. Если `SET_NUMBER` не указан, правило будет добавлено в набор `0`.

ACTION

Правило может быть связано с одним из следующих действий. Указанное действие будет выполнено, когда пакет соответствует критерию выбора правила.

`allow | accept | pass | permit`: эти ключевые слова эквивалентны и разрешают пакеты, соответствующие правилу.

`check-state`: проверяет пакет по таблице динамических состояний. Если совпадение найдено, выполняется действие, связанное с правилом, которое создало это динамическое правило, в противном случае осуществляется переход к следующему правилу. Правило `check-state` не имеет критериев выбора. Если правило `check-state` отсутствует в наборе правил, таблица динамических правил проверяется при первом правиле `keep-state` или `limit`.

`count`: обновляет счетчики для всех пакетов, соответствующих правилу. Поиск продолжается со следующего правила.

`deny | drop`: любое из этих слов тихо отбрасывает пакеты, соответствующие этому правилу.

Доступны дополнительные действия. Подробности смотрите в ipfw(8).

LOG_AMOUNT

Когда пакет соответствует правилу с ключевым словом `log`, сообщение будет записано в syslogd(8) с именем средства `SECURITY`. Запись в журнал происходит только в том случае, если количество пакетов, зарегистрированных для этого конкретного правила, не превышает указанного LOG_AMOUNT. Если LOG_AMOUNT не указан, лимит берется из значения `net.inet.ip.fw.verbose_limit`. Значение нуля снимает ограничение на запись в журнал. Как только лимит достигнут, запись в журнал можно снова включить, сбросив счетчик журналирования или счетчик пакетов для этого правила с помощью `ipfw resetlog`.

Журналирование выполняется после того, как все остальные условия сопоставления пакета выполнены, и перед выполнением конечного действия с пакетом. Администратор решает, для каких правил включить журналирование.

PROTO

Это необязательное значение может использоваться для указания любого имени протокола или номера, найденного в /etc/protocols.

SRC

Ключевое слово `from` должно сопровождаться исходным адресом или ключевым словом,

представляющим исходный адрес. Адрес может быть представлен как `any`, `me` (любой адрес, настроенный на интерфейсе этой системы), `me6` (любой IPv6-адрес, настроенный на интерфейсе этой системы) или `table`, за которым следует номер таблицы поиска, содержащей список адресов. При указании IP-адреса, он может быть дополнительно указан с маской CIDR или маской подсети. Например, `1.2.3.4/25` или `1.2.3.4:255.255.255.128`.

SRC_PORT

Необязательный порт источника может быть указан с использованием номера порта или имени из /etc/services.

DST

Ключевое слово `to` должно сопровождаться адресом назначения или ключевым словом, представляющим адрес назначения. Те же ключевые слова и адреса, которые описаны в разделе SRC, могут быть использованы для описания назначения.

DST_PORT

Необязательный порт назначения может быть указан с использованием номера порта или имени из /etc/services.

OPTIONS

Несколько ключевых слов могут следовать за источником и назначением. Как следует из названия, OPTIONS являются необязательными. Часто используемые опции включают `in` или `out`, которые указывают направление потока пакетов, `icmptypes` с указанием типа ICMP-сообщения и `keep-state`.

Когда правило `keep-state` совпадает, межсетевой экран создаст динамическое правило, которое соответствует двунаправленному трафику между исходным и целевым адресами и портами, используя тот же протокол.

Функция динамических правил уязвима к истощению ресурсов из-за SYN-флуда, который может создать огромное количество динамических правил. Для защиты от такого типа атак в IPFW используйте параметр `limit`. Этот параметр ограничивает количество одновременных сессий, проверяя открытые динамические правила и подсчитывая, сколько раз встречалось сочетание данного правила и IP-адреса. Если это количество превышает значение, указанное в `limit`, пакет отбрасывается.

Доступны десятки параметров `OPTIONS`. Описание каждого доступного параметра можно найти в ipfw(8).

34.4.3. Пример набора правил

В этом разделе показано, как создать пример набора правил для статического межсетевого экрана в виде скрипта с именем /etc/ipfw.rules. В данном примере все правила соединений используют `in` или `out` для указания направления. Они также используют `via` *имя-интерфейса* для указания интерфейса, через который проходит пакет.

При первоначальном создании или тестировании набора правил межсетевого экрана рекомендуется временно установить этот параметр:

```
net.inet.ip.fw.default_to_accept="1"
```

Это устанавливает политику по умолчанию для ipfw(8) более разрешительной, чем стандартная `deny ip from any to any`, что немного снижает вероятность блокировки системы сразу после перезагрузки.

Скрипт межсетевого экрана начинается с указания, что это скрипт POSIX® shell, и очищает все существующие правила. Затем он создаёт переменную `cmd`, чтобы не приходилось вводить `ipfw add` в начале каждого правила. Также определяется переменная `pif`, которая представляет имя интерфейса, подключённого к Интернету.

```
#!/bin/sh
# Flush out the list before we begin.
ipfw -q -f flush

# Set rules command prefix
cmd="ipfw -q add"
pif="dc0"     # interface name of NIC attached to Internet
```

Первые два правила разрешают весь трафик на доверенном внутреннем интерфейсе и на loopback-интерфейсе:

```
# Change xl0 to LAN NIC interface name
$cmd 00005 allow all from any to any via xl0

# No restrictions on Loopback Interface
$cmd 00010 allow all from any to any via lo0
```

Следующее правило пропускает пакет, если он соответствует существующей записи в таблице динамических правил:

```
$cmd 00101 check-state
```

Следующий набор правил определяет, какие состояния соединений внутренние системы могут устанавливать с узлами в Интернете:

```
# Allow access to public DNS
# Replace x.x.x.x with the IP address of a public DNS server
# and repeat for each DNS server in /etc/resolv.conf
$cmd 00110 allow tcp from any to x.x.x.x 53 out via $pif setup keep-state
$cmd 00111 allow udp from any to x.x.x.x 53 out via $pif keep-state

# Allow access to ISP's DHCP server for cable/DSL configurations.
# Use the first rule and check log for IP address.
# Then, uncomment the second rule, input the IP address, and delete the first rule
```

```
$cmd 00120 allow log udp from any to any 67 out via $pif keep-state
#$cmd 00120 allow udp from any to x.x.x.x 67 out via $pif keep-state

# Allow outbound HTTP and HTTPS connections
$cmd 00200 allow tcp from any to any 80 out via $pif setup keep-state
$cmd 00220 allow tcp from any to any 443 out via $pif setup keep-state

# Allow outbound email connections
$cmd 00230 allow tcp from any to any 25 out via $pif setup keep-state
$cmd 00231 allow tcp from any to any 110 out via $pif setup keep-state

# Allow outbound ping
$cmd 00250 allow icmp from any to any out via $pif keep-state

# Allow outbound NTP
$cmd 00260 allow udp from any to any 123 out via $pif keep-state

# Allow outbound SSH
$cmd 00280 allow tcp from any to any 22 out via $pif setup keep-state

# deny and log all other outbound connections
$cmd 00299 deny log all from any to any out via $pif
```

Следующий набор правил управляет соединениями от хостов Интернета к внутренней сети. Он начинается с запрета пакетов, обычно связанных с атаками, а затем явно разрешает определённые типы соединений. Все авторизованные сервисы, поступающие из Интернета, используют `limit` для предотвращения перегрузки.

```
# Deny all inbound traffic from non-routable reserved address spaces
$cmd 00300 deny all from 192.168.0.0/16 to any in via $pif     #RFC 1918 private IP
$cmd 00301 deny all from 172.16.0.0/12 to any in via $pif      #RFC 1918 private IP
$cmd 00302 deny all from 10.0.0.0/8 to any in via $pif         #RFC 1918 private IP
$cmd 00303 deny all from 127.0.0.0/8 to any in via $pif        #loopback
$cmd 00304 deny all from 0.0.0.0/8 to any in via $pif          #loopback
$cmd 00305 deny all from 169.254.0.0/16 to any in via $pif     #DHCP auto-config
$cmd 00306 deny all from 192.0.2.0/24 to any in via $pif       #reserved for docs
$cmd 00307 deny all from 204.152.64.0/23 to any in via $pif    #Sun cluster
interconnect
$cmd 00308 deny all from 224.0.0.0/3 to any in via $pif        #Class D & E multicast

# Deny public pings
$cmd 00310 deny icmp from any to any in via $pif

# Deny ident
$cmd 00315 deny tcp from any to any 113 in via $pif

# Deny all Netbios services.
$cmd 00320 deny tcp from any to any 137 in via $pif
$cmd 00321 deny tcp from any to any 138 in via $pif
$cmd 00322 deny tcp from any to any 139 in via $pif
```

```
$cmd 00323 deny tcp from any to any 81 in via $pif

# Deny fragments
$cmd 00330 deny all from any to any frag in via $pif

# Deny ACK packets that did not match the dynamic rule table
$cmd 00332 deny tcp from any to any established in via $pif

# Allow traffic from ISP's DHCP server.
# Replace x.x.x.x with the same IP address used in rule 00120.
#$cmd 00360 allow udp from any to x.x.x.x 67 in via $pif keep-state

# Allow HTTP connections to internal web server
$cmd 00400 allow tcp from any to me 80 in via $pif setup limit src-addr 2

# Allow inbound SSH connections
$cmd 00410 allow tcp from any to me 22 in via $pif setup limit src-addr 2

# Reject and log all other incoming connections
$cmd 00499 deny log all from any to any in via $pif
```

Последнее правило записывает в журнал информацию о всех пакетах, которые не соответствуют ни одному из правил в наборе правил:

```
# Everything else is denied and logged
$cmd 00999 deny log all from any to any
```

34.4.4. NAT в ядре системы

Межсетевой экран IPFW в FreeBSD имеет две реализации NAT: реализацию в пользовательском пространстве natd(8) и более новую реализацию NAT в ядре. Обе работают совместно с IPFW для преобразования сетевых адресов. Это можно использовать для организации общего доступа в Интернет, чтобы несколько внутренних компьютеров могли подключаться к Интернету, используя один публичный IP-адрес.

Для этого машина FreeBSD, подключённая к Интернету, должна выступать в роли шлюза. Эта система должна иметь две сетевые карты (NIC), где одна подключена к Интернету, а другая — к внутренней локальной сети (LAN). Каждой машине в LAN должен быть назначен IP-адрес из частного адресного пространства, как определено в RFC 1918.

Для включения встроенной в ядро функции NAT в IPFW требуется дополнительная настройка. Чтобы поддержка NAT на уровне ядра включалась при загрузке, необходимо добавить следующие параметры в /etc/rc.conf:

```
gateway_enable="YES"
firewall_enable="YES"
firewall_nat_enable="YES"
```

Когда `firewall_nat_enable` установлен, но `firewall_enable` нет, это не будет иметь эффекта и ничего не сделает. Это связано с тем, что реализация NAT в ядре совместима только с IPFW.

Когда набор правил содержит правила с отслеживанием состояния, позиционирование правила NAT критически важно, и используется действие `skipto`. Действие `skipto` требует указания номера правила, чтобы знать, к какому правилу перейти. В приведённом ниже примере расширяется набор правил межсетевого экрана, показанный в предыдущем разделе. В него добавлены некоторые новые записи и изменены существующие правила для настройки межсетевого экрана с поддержкой NAT, встроенного в ядро. Начинается он с добавления дополнительных переменных, представляющих номер правила для перехода, опцию `keep-state` и список TCP-портов, который будет использоваться для сокращения количества правил.

```
#!/bin/sh
ipfw -q -f flush
cmd="ipfw -q add"
skip="skipto 1000"
pif=dc0
ks="keep-state"
good_tcpo="22,25,37,53,80,443,110"
```

При использовании NAT в ядре необходимо отключить аппаратную сегментацию TCP (TSO) из-за архитектуры libalias(3) — библиотеки, реализованной в виде модуля ядра для обеспечения функциональности NAT в IPFW. TSO можно отключить для каждого сетевого интерфейса с помощью ifconfig(8) или для всей системы с помощью sysctl(8). Чтобы отключить TSO для всей системы, необходимо добавить следующие настройки в /etc/sysctl.conf:

```
net.inet.tcp.tso="0"
```

Будет также настроен экземпляр NAT. Возможно иметь несколько экземпляров NAT, каждый со своей конфигурацией. Для этого примера нужен только один экземпляр NAT — экземпляр NAT номер 1. Конфигурация может принимать несколько опций, таких как: `if`, указывающий публичный интерфейс, `same_ports`, обеспечивающий одинаковое отображение алиасов портов и локальных номеров портов, `unreg_only`, приводящий к обработке только незарегистрированных (частных) адресных пространств экземпляром NAT, и `reset`, который помогает поддерживать работоспособность экземпляра NAT даже при изменении публичного IP-адреса машины с IPFW. Для всех возможных опций, которые могут быть переданы в конфигурацию отдельного экземпляра NAT, обратитесь к ipfw(8). При настройке межсетевого экрана с NAT и отслеживанием состояний необходимо разрешить повторное внедрение преобразованных пакетов в межсетевой экран для дальнейшей обработки. Это можно достичь, отключив поведение `one_pass` в начале скрипта межсетевого экрана.

```
ipfw disable one_pass
```

```
ipfw -q nat 1 config if $pif same_ports unreg_only reset
```

Правило входящего NAT вставляется *после* двух правил, которые разрешают весь трафик на доверенных интерфейсах и интерфейсе loopback, а также после правила пересборки, но *до* правила `check-state`. Важно, чтобы номер правила, выбранный для этого NAT-правила (в данном примере `100`), был больше, чем первые три правила, и меньше, чем правило `check-state`. Кроме того, из-за особенностей работы встроенного NAT рекомендуется размещать правило пересборки непосредственно перед первым NAT-правилом и после правил, разрешающих трафик на доверенных интерфейсах. Обычно фрагментация IP-пакетов не должна происходить, но при работе с туннелированным трафиком IPSEC/ESP/GRE это возможно, и пересборка фрагментов необходима перед передачей полного пакета встроенному механизму NAT.

Правило пересборки не требовалось при использовании пользовательского демона natd(8), поскольку внутренняя работа действия `divert` в ipfw(8) уже обеспечивает пересборку пакетов перед их передачей в сокет, как также указано в ipfw(8).

Экземпляр NAT и номер правила, используемые в этом примере, не совпадают с экземпляром NAT и номером правила, созданными по умолчанию в rc.firewall. rc.firewall — это скрипт, который настраивает правила межсетевого экрана по умолчанию в FreeBSD.

```
$cmd 005 allow all from any to any via xl0  # exclude LAN traffic
$cmd 010 allow all from any to any via lo0  # exclude loopback traffic
$cmd 099 reass all from any to any in       # reassemble inbound packets
$cmd 100 nat 1 ip from any to any in via $pif # NAT any inbound packets
# Allow the packet through if it has an existing entry in the dynamic rules table
$cmd 101 check-state
```

Правила для исходящего трафика изменены, чтобы заменить действие `allow` на переменную `$skip`, указывая, что обработка правил продолжится с правила `1000`. Семь правил для `tcp` заменены правилом `125`, так как переменная `$good_tcpo` содержит семь разрешённых исходящих портов.

Помните, что производительность IPFW в значительной степени определяется количеством правил в наборе правил межсетевого экрана.

```
# Authorized outbound packets
$cmd 120 $skip udp from any to x.x.x.x 53 out via $pif $ks
$cmd 121 $skip udp from any to x.x.x.x 67 out via $pif $ks
$cmd 125 $skip tcp from any to any $good_tcpo out via $pif setup $ks
$cmd 130 $skip icmp from any to any out via $pif $ks
```

Правила для входящего трафика остаются такими же, за исключением самого последнего правила, в котором удаляется `via $pif`, чтобы охватить как входящие, так и исходящие

правила. Правило NAT должно следовать за этим последним исходящим правилом, иметь номер выше, чем у последнего правила, и номер правила должен быть указан в действии `skipto`. В этом наборе правил правило номер `1000` обрабатывает передачу всех пакетов в настроенный экземпляр для обработки NAT. Следующее правило разрешает прохождение любого пакета, прошедшего обработку NAT.

```
$cmd 999 deny log all from any to any
$cmd 1000 nat 1 ip from any to any out via $pif # skipto location for outbound
stateful rules
$cmd 1001 allow ip from any to any
```

В этом примере правила `100`, `101`, `125`, `1000` и `1001` управляют преобразованием адресов исходящих и входящих пакетов, чтобы записи в таблице динамического состояния всегда указывали на частный LANIP-адрес.

Рассмотрим внутренний веб-браузер, который инициирует новый исходящий HTTP-сеанс через порт 80. Когда первый исходящий пакет попадает в межсетевой экран, он не соответствует правилу `100`, так как направлен наружу, а не внутрь. Он проходит правило `101`, так как это первый пакет и он ещё не был добавлен в таблицу динамического состояния. В итоге пакет соответствует правилу `125`, так как он исходящий на разрешённом порту и имеет исходный IP-адрес из внутренней LAN. При соответствии этому правилу выполняются два действия. Во-первых, действие `keep-state` добавляет запись в таблицу динамического состояния, и выполняется указанное действие `skipto rule 1000`. Затем пакет проходит NAT и отправляется в Интернет. Этот пакет достигает целевого веб-сервера, где генерируется и отправляется обратно ответный пакет. Этот новый пакет попадает в начало набора правил. Он соответствует правилу `100`, и его IP-адрес назначения преобразуется обратно в исходный внутренний адрес. Затем он обрабатывается правилом `check-state`, обнаруживается в таблице как существующий сеанс и передаётся в LAN.

На входящей стороне набор правил должен блокировать плохие пакеты и разрешать только авторизованные сервисы. Пакет, соответствующий входящему правилу, помещается в таблицу динамического состояния и выпускается в локальную сеть. Пакет, сгенерированный в ответ, распознаётся правилом `check-state` как принадлежащий существующему сеансу. Затем он отправляется к правилу `1000` для выполнения NAT перед выпуском на исходящий интерфейс.

Переход от пользовательской реализации natd(8) к NAT в ядре может сначала показаться простым, но есть небольшой нюанс. При использовании GENERIC ядра IPFW загрузит модуль ядра libalias.ko, когда в /etc/rc.conf включена опция `firewall_nat_enable`. Модуль ядра libalias.ko предоставляет только базовую функциональность NAT, в то время как пользовательская реализация natd(8) имеет все функции NAT в своей пользовательской библиотеке без дополнительной настройки. Под всей функциональностью подразумеваются следующие модули ядра, которые могут быть дополнительно загружены при необходимости, помимо стандартного модуля libalias.ko: alias_ftp.ko, alias_bbt.ko, skinny.ko, irc.ko, alias_pptp.ko и alias_smedia.ko с использованием директивы `kld_list` в /etc/rc.conf. Если

> используется собственное ядро, полная функциональность пользовательской библиотеки может быть встроена в ядро с помощью опции `options LIBALIAS`.

Перенаправление портов

Недостаток NAT в целом заключается в том, что клиенты в локальной сети недоступны из Интернета. Клиенты в локальной сети могут устанавливать исходящие соединения с внешним миром, но не могут принимать входящие. Это создаёт проблему, если необходимо запустить интернет-сервисы на одной из машин-клиентов локальной сети. Простое решение этой проблемы — перенаправление определённых портов из Интернета на машине, предоставляющей NAT, к клиенту в локальной сети.

Например, сервер IRC работает на клиенте `A`, а веб-сервер — на клиенте `B`. Для правильной работы соединения, полученные на портах 6667 (IRC) и 80 (HTTP), должны быть перенаправлены на соответствующие машины.

С встроенным в ядро NAT вся конфигурация выполняется в настройках экземпляра NAT. Полный список параметров, которые может использовать экземпляр встроенного в ядро NAT, приведён в ipfw(8). Синтаксис IPFW соответствует синтаксису natd. Синтаксис для `redirect_port` выглядит следующим образом:

```
redirect_port proto targetIP:targetPORT[-targetPORT]
  [aliasIP:]aliasPORT[-aliasPORT]
  [remoteIP[:remotePORT[-remotePORT]]]
```

Для настройки приведённой выше конфигурации аргументы должны быть:

```
redirect_port tcp 192.168.0.2:6667 6667
redirect_port tcp 192.168.0.3:80 80
```

После добавления этих аргументов в конфигурацию NAT-экземпляра 1 в приведённом выше наборе правил, TCP-порты будут проброшены на клиентские машины в локальной сети, на которых работают службы IRC и HTTP.

```
ipfw -q nat 1 config if $pif same_ports unreg_only reset \
  redirect_port tcp 192.168.0.2:6667 6667 \
  redirect_port tcp 192.168.0.3:80 80
```

Диапазоны портов вместо отдельных портов могут быть указаны с помощью `redirect_port`. Например, *tcp 192.168.0.2:2000-3000 2000-3000* перенаправит все соединения, полученные на портах с 2000 по 3000, на порты с 2000 по 3000 клиента `A`.

Перенаправление адресов

Перенаправление адресов полезно, если доступно более одного IP-адреса. Каждому клиенту

в локальной сети может быть назначен собственный внешний IP-адрес с помощью ipfw(8), который затем перезаписывает исходящие пакеты от клиентов локальной сети с соответствующим внешним IP-адресом и перенаправляет весь входящий трафик для этого конкретного IP-адреса обратно к определённому клиенту локальной сети. Это также известно как статический NAT. Например, если доступны IP-адреса `128.1.1.1`, `128.1.1.2` и `128.1.1.3`, то `128.1.1.1` может использоваться как внешний IP-адрес машины с ipfw(8), а `128.1.1.2` и `128.1.1.3` перенаправляются обратно клиентам `A` и `B` локальной сети.

Синтаксис `redirect_addr` приведён ниже, где `localIP` — это внутренний IP-адрес клиента в локальной сети, а `publicIP` — внешний IP-адрес, соответствующий клиенту в локальной сети.

```
redirect_addr localIP publicIP
```

В этом примере аргументы выглядели бы следующим образом:

```
redirect_addr 192.168.0.2 128.1.1.2
redirect_addr 192.168.0.3 128.1.1.3
```

Как и `redirect_port`, эти аргументы размещаются в конфигурации экземпляра NAT. При перенаправлении адреса нет необходимости в перенаправлении портов, так как перенаправляются все данные, полученные на определённый IP-адрес.

Внешние IP-адреса на машине с ipfw(8) должны быть активны и назначены как псевдонимы внешнему интерфейсу. Подробности см. в rc.conf(5).

Пользовательский NAT

Начнём с утверждения: реализация NAT в пользовательском пространстве: natd(8), имеет больше накладных расходов, чем NAT в ядре. Для работы natd(8) по преобразованию пакетов, пакеты должны копироваться из ядра в пользовательское пространство и обратно, что создаёт дополнительные накладные расходы, отсутствующие при использовании NAT в ядре.

Для включения демона NAT в пользовательском пространстве natd(8) при загрузке, следующая минимальная конфигурация должна быть добавлена в /etc/rc.conf. Параметр `natd_interface` должен быть установлен равным имени сетевого интерфейса, подключённого к Интернету. Скрипт rc(8) для natd(8) автоматически проверит, используется ли динамический IP-адрес, и настроит себя соответствующим образом.

```
gateway_enable="YES"
natd_enable="YES"
natd_interface="rl0"
```

В общем случае, приведённый выше набор правил, описанный для NAT в ядре, также может использоваться совместно с natd(8). Исключениями являются настройка экземпляра NAT в

ядре (`ipfw -q nat 1 config …`), которая не требуется вместе с правилом пересборки 99, так как его функциональность включена в действие `divert`. Правила 100 и 1000 необходимо немного изменить, как показано ниже.

```
$cmd 100 divert natd ip from any to any in via $pif
$cmd 1000 divert natd ip from any to any out via $pif
```

Для настройки перенаправления портов или адресов используется синтаксис, аналогичный NAT в ядре. Однако, в отличие от настройки в скрипте набора правил, как в случае с NAT в ядре, конфигурацию natd(8) лучше выполнять в конфигурационном файле. Для этого необходимо передать дополнительный флаг через /etc/rc.conf, который указывает путь к конфигурационному файлу.

```
natd_flags="-f /etc/natd.conf"
```

Указанный файл должен содержать список параметров конфигурации, по одному на строку. Для получения дополнительной информации о файле конфигурации и возможных переменных обратитесь к natd(8). Ниже приведены два примера записей, по одному на строку:

```
redirect_port tcp 192.168.0.2:6667 6667
redirect_addr 192.168.0.3 128.1.1.3
```

34.4.5. Команда IPFW

`ipfw` можно использовать для ручного добавления или удаления отдельных правил в активный межсетевой экран во время его работы. Проблема этого метода в том, что все изменения теряются при перезагрузке системы. Рекомендуется вместо этого записывать все правила в файл и использовать его для загрузки правил при запуске системы, а также для замены текущих правил межсетевого экрана при изменении этого файла.

`ipfw` — это полезный способ отображения текущих правил межсетевого экрана на экране консоли. Возможность учёта IPFW динамически создаёт счётчик для каждого правила, который подсчитывает каждый пакет, соответствующий правилу. В процессе тестирования правила, вывод правила с его счётчиком — это один из способов определить, работает ли правило так, как ожидается.

Чтобы вывести список всех активных правил в порядке их применения:

```
# ipfw list
```

Для вывода всех активных правил с отметкой времени последнего совпадения правила:

```
# ipfw -t list
```

Следующий пример выводит информацию о сборе статистики и количество пакетов для совпавших правил вместе с самими правилами. Первый столбец — это номер правила, за которым следует количество совпавших пакетов и байтов, а затем само правило.

```
# ipfw -a list
```

Для отображения динамических правил вместе со статическими:

```
# ipfw -d list
```

Чтобы также показать истекшие динамические правила:

```
# ipfw -d -e list
```

Для обнуления счетчиков:

```
# ipfw zero
```

Обнулить счетчики только для правила с номером *NUM*:

```
# ipfw zero NUM
```

Журналирование сообщений межсетевого экрана

Даже при включённой функции журналирования, IPFW не будет самостоятельно генерировать записи о правилах. Администратор межсетевого экрана решает, какие правила в наборе правил будут записываться в журнал, и добавляет ключевое слово `log` к этим правилам. Обычно журналируются только правила с действием deny. Принято дублировать правило "ipfw default deny everything" с включённым ключевым словом `log` в качестве последнего правила в наборе. Таким образом, можно увидеть все пакеты, которые не соответствуют ни одному из правил в наборе.

Журналирование — это обоюдоострый меч. Если не быть осторожным, переизбыток данных журналирования или атака типа DoS могут заполнить диск файлами журналов. Сообщения журнала записываются не только в syslogd, но и выводятся на корневую консоль, что быстро становится раздражающим.

Опция ядра `IPFIREWALL_VERBOSE_LIMIT=5` ограничивает количество последовательных сообщений, отправляемых в syslogd(8), касающихся совпадения пакетов с заданным правилом. Когда эта опция включена в ядре, количество последовательных сообщений, касающихся определённого правила, ограничивается указанным числом. Нет никакой

пользы от 200 одинаковых сообщений в журнале. При установке этого параметра в пять, пять последовательных сообщений, касающихся определённого правила, будут записаны в syslogd, а оставшиеся идентичные последовательные сообщения будут подсчитаны и отправлены в syslogd с фразой, подобной следующей:

```
last message repeated 45 times
```

Все сообщения о зарегистрированных пакетах по умолчанию записываются в /var/log/security, что определено в /etc/syslog.conf.

Построение скрипта правил

Большинство опытных пользователей IPFW создают файл, содержащий правила, и оформляют их таким образом, чтобы их можно было запускать как скрипт. Основное преимущество этого подхода заключается в том, что правила межсетевого экрана можно обновлять массово без необходимости перезагрузки системы для их активации. Этот метод удобен при тестировании новых правил, так как процедуру можно выполнять столько раз, сколько потребуется. Будучи скриптом, можно использовать символические подстановки для часто используемых значений, которые будут заменяться в нескольких правилах.

Синтаксис примера, приведённого ниже, совместим с синтаксисом, используемым оболочками sh(1), csh(1) и tcsh(1). Поля символьной подстановки (переменные — один из видов полей подстановки, прим. перев.) начинаются со знака доллара ($). Символьные поля не имеют префикса $. Значение для заполнения символьного поля должно быть заключено в двойные кавычки ("").

Начните файл правил следующим образом:

```
################ start of example ipfw rules script #############
#
ipfw -q -f flush       # Delete all rules
# Set defaults
oif="tun0"             # out interface
odns="192.0.2.11"      # ISP's DNS server IP address
cmd="ipfw -q add "     # build rule prefix
ks="keep-state"        # just too lazy to key this each time
$cmd 00500 check-state
$cmd 00502 deny all from any to any frag
$cmd 00501 deny tcp from any to any established
$cmd 00600 allow tcp from any to any 80 out via $oif setup $ks
$cmd 00610 allow tcp from any to $odns 53 out via $oif setup $ks
$cmd 00611 allow udp from any to $odns 53 out via $oif $ks
################### End of example ipfw rules script #############
```

Правила не важны, так как цель этого примера — показать, как заполняются поля символьной подстановки.

Если приведённый выше пример находится в /etc/ipfw.rules, правила можно перезагрузить

следующей командой:

```
# sh /etc/ipfw.rules
```

/etc/ipfw.rules может находиться в любом месте, и файл может иметь любое имя.

То же самое можно выполнить вручную, запустив следующие команды:

```
# ipfw -q -f flush
# ipfw -q add check-state
# ipfw -q add deny all from any to any frag
# ipfw -q add deny tcp from any to any established
# ipfw -q add allow tcp from any to any 80 out via tun0 setup keep-state
# ipfw -q add allow tcp from any to 192.0.2.11 53 out via tun0 setup keep-state
# ipfw -q add 00611 allow udp from any to 192.0.2.11 53 out via tun0 keep-state
```

34.4.6. Опции ядра для IPFW

Для статической компиляции поддержки IPFW в собственное ядро обратитесь к инструкциям в Настройка ядра FreeBSD. В файле конфигурации собственного ядра доступны следующие параметры:

```
options    IPFIREWALL           # enables IPFW
options    IPFIREWALL_VERBOSE       # enables logging for rules with log keyword to
syslogd(8)
options    IPFIREWALL_VERBOSE_LIMIT=5   # limits number of logged packets per-entry
options    IPFIREWALL_DEFAULT_TO_ACCEPT # sets default policy to pass what is not
explicitly denied
options    IPFIREWALL_NAT       # enables basic in-kernel NAT support
options    LIBALIAS         # enables full in-kernel NAT support
options    IPFIREWALL_NAT64     # enables in-kernel NAT64 support
options    IPFIREWALL_NPTV6     # enables in-kernel IPv6 NPT support
options    IPFIREWALL_PMOD      # enables protocols modification module support
options    IPDIVERT         # enables NAT through natd(8)
```

IPFW может быть загружен как модуль ядра: указанные выше параметры по умолчанию собираются как модули или могут быть установлены во время выполнения с помощью tunables.

34.5. IPFILTER (IPF)

IPFILTER, также известный как IPF, представляет собой кроссплатформенный межсетевой экран с открытым исходным кодом, который был портирован на несколько операционных систем, включая FreeBSD, NetBSD, OpenBSD и Solaris™.

IPFILTER — это механизм межсетевого экрана и NAT на уровне ядра, которым можно

управлять и отслеживать его работу с помощью пользовательских программ. Правила межсетевого экрана можно устанавливать или удалять с помощью `ipf`, правила NAT — с помощью `ipnat`, статистику работы ядерной части IPFILTER в реальном времени можно выводить с помощью `ipfstat`, а `ipmon` позволяет записывать действия IPFILTER в системные журналы.

IPF изначально был написан с использованием логики обработки правил "последнее совпавшее правило побеждает" и использовал только правила без состояния. С тех пор IPF был улучшен и теперь включает опции `quick` и `keep state`.

Часто задаваемые вопросы по IPF находятся на сайте http://www.phildev.net/ipf/index.html. Доступен поисковый архив списка рассылки IPFilter по адресу http://marc.info/?l=ipfilter.

Этот раздел Руководства посвящен IPF в контексте FreeBSD. В нём приведены примеры правил, содержащих опции `quick` и `keep state`.

34.5.1. Включение IPF

IPF включён в базовую установку FreeBSD в виде загружаемого модуля ядра, что означает, что для включения IPF не требуется создавать собственное ядро.

Для пользователей, которые предпочитают статически компилировать поддержку IPF в пользовательское ядро, обратитесь к инструкциям в Настройка ядра FreeBSD. Доступны следующие параметры ядра:

```
options IPFILTER
options IPFILTER_LOG
options IPFILTER_LOOKUP
options IPFILTER_DEFAULT_BLOCK
```

где `options IPFILTER` включает поддержку IPFILTER, `options IPFILTER_LOG` включает журналирование IPF с использованием псевдоустройства ipl для записи пакетов для каждого правила, содержащего ключевое слово `log`, `IPFILTER_LOOKUP` включает пулы IP для ускорения поиска IP, а `options IPFILTER_DEFAULT_BLOCK` изменяет поведение по умолчанию так, что любой пакет, не соответствующий правилу `pass` межсетевого экрана, блокируется.

Для настройки системы на включение IPF при загрузке добавьте следующие записи в /etc/rc.conf. Эти записи также включат журналирование и политику `default pass all`. Чтобы изменить политику по умолчанию на `block all` без компиляции пользовательского ядра, не забудьте добавить правило `block all` в конец набора правил.

```
ipfilter_enable="YES"            # Start ipf firewall
ipfilter_rules="/etc/ipf.rules"  # loads rules definition text file
ipv6_ipfilter_rules="/etc/ipf6.rules" # loads rules definition text file for IPv6
ipmon_enable="YES"               # Start IP monitor log
ipmon_flags="-Ds"                # D = start as daemon
                                 # s = log to syslog
                                 # v = log tcp window, ack, seq
```

```
                                # n = map IP & port to names
```

Если требуется функциональность NAT, также добавьте следующие строки:

```
gateway_enable="YES"              # Enable as LAN gateway
ipnat_enable="YES"                # Start ipnat function
ipnat_rules="/etc/ipnat.rules"    # rules definition file for ipnat
```

Затем, чтобы запустить IPF сейчас:

```
#  service ipfilter start
```

Для загрузки правил межсетевого экрана укажите имя файла набора правил, используя `ipf`. Следующая команда может быть использована для замены текущих работающих правил межсетевого экрана:

```
# ipf -Fa -f /etc/ipf.rules
```

где `-Fa` очищает все внутренние таблицы правил, а `-f` указывает файл, содержащий правила для загрузки.

Это предоставляет возможность вносить изменения в пользовательский набор правил и обновлять работающий межсетевой экран новой копией правил без необходимости перезагрузки системы. Этот метод удобен для тестирования новых правил, так как процедуру можно выполнять столько раз, сколько потребуется.

Обратитесь к ipf(8) для получения подробностей о других флагах, доступных с этой командой.

34.5.2. Синтаксис правил IPF

В этом разделе описывается синтаксис правил IPF, используемый для создания правил с состояниями. При создании правил следует помнить, что если в правиле не указано ключевое слово `quick`, каждое правило обрабатывается по порядку, и *последнее совпавшее правило* будет применено. Это означает, что даже если первое совпавшее правило разрешает (`pass`), но далее есть совпадающее правило, которое запрещает (`block`), пакет будет отброшен. Примеры наборов правил можно найти в /usr/share/examples/ipfilter.

При создании правил символ `#` используется для обозначения начала комментария и может находиться в конце правила, чтобы пояснить его функцию, или на отдельной строке. Пустые строки игнорируются.

Ключевые слова, используемые в правилах, должны быть записаны в определённом порядке, слева направо. Некоторые ключевые слова являются обязательными, а другие — опциональными. Некоторые ключевые слова имеют подопции, которые могут сами быть ключевыми словами и включать дополнительные подопции. Порядок ключевых слов

следующий, где слова, написанные в ВЕРХНЕМ РЕГИСТРЕ, представляют переменную, а слова, написанные в нижнем регистре, должны предшествовать следующей за ними переменной:

```
ACTION DIRECTION OPTIONS proto PROTO_TYPE from SRC_ADDR SRC_PORT to DST_ADDR DST_PORT
TCP_FLAG|ICMP_TYPE keep state STATE
```

В этом разделе описываются ключевые слова и их параметры. Это не исчерпывающий список всех возможных параметров. Полное описание синтаксиса правил, который можно использовать при создании правил IPF, а также примеры использования каждого ключевого слова приведены в ipf(5).

ACTION

Ключевое слово `action` указывает действие — что делать с пакетом, если он соответствует данному правилу. Каждое правило *должно* иметь действие. Поддерживаются следующие действия:

`block`: отбрасывает пакет.

`pass`: разрешает пакет.

`log`: создаёт запись в журнале.

`count`: подсчитывает количество пакетов и байтов, что может дать представление о том, как часто используется правило.

`auth`: ставит пакет в очередь для дальнейшей обработки другой программой.

`call`: предоставляет доступ к функциям, встроенным в IPF, которые позволяют выполнять более сложные действия.

`decapsulate`: удаляет любые заголовки для обработки содержимого пакета.

DIRECTION

Далее, каждое правило должно явно указывать `direction ` — направление трафика с использованием одного из следующих ключевых слов:

`in`: правило применяется к входящему пакету.

`out`: правило применяется к исходящему пакету.

`all`: правило применяется в обоих направлениях.

Если в системе несколько интерфейсов, можно указать интерфейс вместе с направлением. Примером может быть `in on fxp0`.

OPTIONS

`options` — параметры необязательны. Однако, если указано несколько параметров, они должны использоваться в порядке, указанном здесь.

`log`: при выполнении указанного ДЕЙСТВИЯ содержимое заголовков пакета будет

записано в псевдоустройство журнала пакетов ipl(4).

`quick`: если пакет соответствует этому правилу, происходит ДЕЙСТВИЕ, указанное в правиле, и дальнейшая обработка последующих правил для этого пакета не выполняется.

`on`: должен сопровождаться именем интерфейса, как отображается в ifconfig(8). Правило будет применяться только в том случае, если пакет проходит через указанный интерфейс в указанном направлении.

При использовании ключевого слова `log` следующие квалификаторы могут быть указаны в таком порядке:

`body`: указывает, что первые 128 байт содержимого пакета будут записаны в журнал после заголовков.

`first`: если ключевое слово `log` используется вместе с опцией `keep state`, рекомендуется использовать эту опцию, чтобы регистрировался только инициирующий пакет, а не каждый пакет, соответствующий соединению с отслеживанием состояний.

Дополнительные параметры доступны для указания сообщений об ошибках. Подробности смотрите в ipf(5).

PROTO_TYPE

Тип протокола является необязательным. Однако он обязателен, если в правиле необходимо указать SRC_PORT или DST_PORT, так как он определяет тип протокола. При указании типа протокола используйте ключевое слово `proto`, за которым следует номер протокола или его имя из /etc/protocols. Примеры названий протоколов: `tcp`, `udp` или `icmp`. Если указан PROTO_TYPE, но не указаны SRC_PORT или DST_PORT, все номера портов для этого протокола будут соответствовать этому правилу.

SRC_ADDR

Ключевое слово `from` является обязательным и сопровождается ключевым словом, обозначающим источник пакета. Источником может быть имя хоста, IP-адрес с указанием маски CIDR, пул адресов или ключевое слово `all`. Примеры можно найти в ipf(5).

Единственный способ задать диапазоны IP-адресов, это представить их в нотации — числа, разделённые точками / длина маски. Пакет или порт net-mgmt/ipcalc может быть использован для упрощения расчета маски CIDR. Дополнительная информация доступна на веб-странице утилиты: http://jodies.de/ipcalc.

SRC_PORT

Номер порта источника является необязательным. Однако, если он используется, требуется, чтобы PROTO_TYPE был сначала определён в правиле. Номер порта также должен предваряться ключевым словом `proto`.

Поддерживается несколько различных операторов сравнения: `=` (равно), `!=` (не равно), `<` (меньше), `>` (больше), ⇐ (меньше или равно) и `>=` (больше или равно).

Для указания диапазона портов поместите два номера порта между `<>` (меньше и больше), `><` (больше и меньше) или `:` (больше или равно и меньше или равно).

DST_ADDR

Ключевое слово `to` является обязательным и за ним следует ключевое слово, обозначающее назначение пакета. Аналогично SRC_ADDR, это может быть имя хоста, IP-адрес с маской CIDR, пул адресов или ключевое слово `all`.

DST_PORT

Аналогично SRC_PORT, номер порта назначения является необязательным. Однако, если он используется, требуется, чтобы PROTO_TYPE был сначала определён в правиле. Номер порта также должен предваряться ключевым словом `proto`.

TCP_FLAG|ICMP_TYPE

Если в качестве `PROTO_TYPE` указан `tcp`, флаги могут быть заданы в виде букв, где каждая буква представляет один из возможных флагов TCP, используемых для определения состояния соединения. Возможные значения: `S` (SYN), `A` (ACK), `P` (PSH), `F` (FIN), `U` (URG), `R` (RST), `C` (CWN) и `E` (ECN).

Если указан `icmp` в качестве PROTO_TYPE, можно указать тип ICMP для сопоставления. Допустимые типы приведены в ipf(5).

STATE

Если правило `pass` содержит `keep state`, IPF добавит запись в свою таблицу динамического состояния и разрешит последующие пакеты, соответствующие соединению. IPF может отслеживать состояние сеансов TCP, UDP и ICMP. Любой пакет, который IPF может однозначно идентифицировать как часть активного сеанса, даже если это другой протокол, будет разрешён.

В IPF пакеты, предназначенные для выхода через интерфейс, подключённый к публичному интернету, сначала проверяются в динамической таблице состояний. Если пакет соответствует следующему ожидаемому пакету активного сеанса связи, он проходит через межсетевой экран, и состояние потока сеанса обновляется в динамической таблице состояний. Пакеты, не принадлежащие к уже активному сеансу, проверяются в соответствии с набором правил для исходящего трафика. Пакеты, поступающие через интерфейс, подключённый к публичному интернету, сначала проверяются в динамической таблице состояний. Если пакет соответствует следующему ожидаемому пакету активного сеанса, он проходит через межсетевой экран, и состояние потока сеанса обновляется в динамической таблице состояний. Пакеты, не принадлежащие к уже активному сеансу, проверяются в соответствии с набором правил для входящего трафика.

Несколько ключевых слов могут быть добавлены после `keep state`. Если они используются, эти ключевые слова устанавливают различные параметры, управляющие фильтрацией с отслеживанием статуса, такие как установка ограничений на соединения или времени жизни соединения. Подробный список доступных параметров и их описания можно найти в ipf(5).

34.5.3. Пример набора правил

Этот раздел демонстрирует, как создать пример набора правил, который разрешает только сервисы, соответствующие правилам `pass`, и блокирует все остальные.

FreeBSD использует loopback-интерфейс (интерфейс обратной петли) (lo0) и IP-адрес `127.0.0.1` для внутреннего взаимодействия. Набор правил межсетевого экрана должен включать правила, разрешающие свободное перемещение этих внутренних пакетов:

```
# no restrictions on loopback interface
pass in quick on lo0 all
pass out quick on lo0 all
```

Публичный интерфейс, подключённый к Интернету, используется для авторизации и контроля доступа всех исходящих и входящих соединений. Если один или несколько интерфейсов подключены к частным сетям, этим внутренним интерфейсам могут потребоваться правила, разрешающие передачу пакетов, исходящих из локальной сети, между внутренними сетями или к интерфейсу, подключённому к Интернету. Набор правил должен быть организован в три основных раздела: доверенные внутренние интерфейсы, исходящие соединения через публичный интерфейс и входящие соединения через публичный интерфейс.

Эти два правила разрешают весь трафик через доверенный интерфейс LAN с именем xl0:

```
# no restrictions on inside LAN interface for private network
pass out quick on xl0 all
pass in quick on xl0 all
```

Правила для исходящего и входящего разделов публичного интерфейса должны иметь наиболее часто совпадающие правила, расположенные перед менее часто совпадающими, а последнее правило в разделе должно блокировать и журналировать все пакеты для этого интерфейса и направления.

Этот набор правил определяет исходящий раздел общедоступного интерфейса с именем dc0. Эти правила сохраняют состояние и определяют конкретные службы, для которых внутренние системы авторизованы для доступа к общедоступному Интернету. Все правила используют `quick` и указывают соответствующие номера портов и, где применимо, адреса назначения.

```
# interface facing Internet (outbound)
# Matches session start requests originating from or behind the
# firewall, destined for the Internet.

# Allow outbound access to public DNS servers.
# Replace x.x.x.x with address listed in /etc/resolv.conf.
# Repeat for each DNS server.
pass out quick on dc0 proto tcp from any to x.x.x.x port = 53 flags S keep state
```

```
pass out quick on dc0 proto udp from any to x.x.x.x port = 53 keep state

# Allow access to ISP's specified DHCP server for cable or DSL networks.
# Use the first rule, then check log for the IP address of DHCP server.
# Then, uncomment the second rule, replace z.z.z.z with the IP address,
# and comment out the first rule
pass out log quick on dc0 proto udp from any to any port = 67 keep state
#pass out quick on dc0 proto udp from any to z.z.z.z port = 67 keep state

# Allow HTTP and HTTPS
pass out quick on dc0 proto tcp from any to any port = 80 flags S keep state
pass out quick on dc0 proto tcp from any to any port = 443 flags S keep state

# Allow email
pass out quick on dc0 proto tcp from any to any port = 110 flags S keep state
pass out quick on dc0 proto tcp from any to any port = 25 flags S keep state

# Allow NTP
pass out quick on dc0 proto tcp from any to any port = 37 flags S keep state

# Allow FTP
pass out quick on dc0 proto tcp from any to any port = 21 flags S keep state

# Allow SSH
pass out quick on dc0 proto tcp from any to any port = 22 flags S keep state

# Allow ping
pass out quick on dc0 proto icmp from any to any icmp-type 8 keep state

# Block and log everything else
block out log first quick on dc0 all
```

Этот пример правил в разделе входящего трафика публичного интерфейса сначала блокирует все нежелательные пакеты. Это уменьшает количество пакетов, регистрируемых последним правилом.

```
# interface facing Internet (inbound)
# Block all inbound traffic from non-routable or reserved address spaces
block in quick on dc0 from 192.168.0.0/16 to any    #RFC 1918 private IP
block in quick on dc0 from 172.16.0.0/12 to any     #RFC 1918 private IP
block in quick on dc0 from 10.0.0.0/8 to any        #RFC 1918 private IP
block in quick on dc0 from 127.0.0.0/8 to any       #loopback
block in quick on dc0 from 0.0.0.0/8 to any         #loopback
block in quick on dc0 from 169.254.0.0/16 to any    #DHCP auto-config
block in quick on dc0 from 192.0.2.0/24 to any      #reserved for docs
block in quick on dc0 from 204.152.64.0/23 to any   #Sun cluster interconnect
block in quick on dc0 from 224.0.0.0/3 to any       #Class D & E multicast

# Block fragments and too short tcp packets
block in quick on dc0 all with frags
```

```
block in quick on dc0 proto tcp all with short

# block source routed packets
block in quick on dc0 all with opt lsrr
block in quick on dc0 all with opt ssrr

# Block OS fingerprint attempts and log first occurrence
block in log first quick on dc0 proto tcp from any to any flags FUP

# Block anything with special options
block in quick on dc0 all with ipopts

# Block public pings and ident
block in quick on dc0 proto icmp all icmp-type 8
block in quick on dc0 proto tcp from any to any port = 113

# Block incoming Netbios services
block in log first quick on dc0 proto tcp/udp from any to any port = 137
block in log first quick on dc0 proto tcp/udp from any to any port = 138
block in log first quick on dc0 proto tcp/udp from any to any port = 139
block in log first quick on dc0 proto tcp/udp from any to any port = 81
```

Всякий раз, когда регистрируются сообщения о правиле с опцией `log first`, выполните команду `ipfstat -hio`, чтобы оценить, сколько раз правило сработало. Большое количество срабатываний может указывать на то, что система подвергается атаке.

Остальные правила в разделе входящего трафика определяют, какие соединения могут быть инициированы из Интернета. Последнее правило запрещает все соединения, которые не были явно разрешены предыдущими правилами в этом разделе.

```
# Allow traffic in from ISP's DHCP server. Replace z.z.z.z with
# the same IP address used in the outbound section.
pass in quick on dc0 proto udp from z.z.z.z to any port = 68 keep state

# Allow public connections to specified internal web server
pass in quick on dc0 proto tcp from any to x.x.x.x port = 80 flags S keep state

# Block and log only first occurrence of all remaining traffic.
block in log first quick on dc0 all
```

34.5.4. Настройка NAT

Чтобы включить NAT, добавьте следующие выражения в /etc/rc.conf и укажите имя файла, содержащего правила NAT:

```
gateway_enable="YES"
ipnat_enable="YES"
```

```
ipnat_rules="/etc/ipnat.rules"
```

Правила NAT гибки и могут выполнять множество различных задач, подходящих как для коммерческих, так и для домашних пользователей. Синтаксис правил, представленный здесь, упрощен для демонстрации типичного использования. Полное описание синтаксиса правил можно найти в ipnat(5).

Базовая синтаксическая структура правила NAT выглядит следующим образом, где `map` начинает правило, а *IF* следует заменить на имя внешнего интерфейса:

```
map IF LAN_IP_RANGE -> PUBLIC_ADDRESS
```

LAN_IP_RANGE — это диапазон IP-адресов, используемых внутренними клиентами. Обычно это частный диапазон адресов, например `192.168.1.0/24`. *PUBLIC_ADDRESS* может быть либо статическим внешним IP-адресом, либо ключевым словом `0/32`, которое представляет IP-адрес, назначенный *IF*.

В IPF, когда пакет прибывает на межсетевой экран из локальной сети с публичным адресом назначения, он сначала проходит через исходящие правила набора правил межсетевого экрана. Затем пакет передаётся в набор правил NAT, который читается сверху вниз, и первое совпадающее правило применяется. IPF проверяет каждое правило NAT на соответствие имени интерфейса и исходному IP-адресу пакета. Когда имя интерфейса пакета совпадает с правилом NAT, исходный IP-адрес пакета в частной локальной сети проверяется на вхождение в диапазон IP-адресов, указанный в *LAN_IP_RANGE*. При совпадении исходный IP-адрес пакета перезаписывается публичным IP-адресом, указанным в *PUBLIC_ADDRESS*. IPF добавляет запись в свою внутреннюю таблицу NAT, чтобы при возврате пакета из Интернета он мог быть сопоставлен с исходным частным IP-адресом перед передачей в набор правил межсетевого экрана для дальнейшей обработки.

Для сетей с большим количеством внутренних систем или несколькими подсетями процесс перенаправления каждого частного IP-адреса в один публичный IP-адрес становится проблемой с точки зрения ресурсов. Доступны два метода для решения этой проблемы.

Первый метод заключается в назначении диапазона портов для использования в качестве исходных портов. Добавление ключевого слова `portmap` позволяет указать NAT использовать только исходные порты из заданного диапазона:

```
map dc0 192.168.1.0/24 -> 0/32 portmap tcp/udp 20000:60000
```

Или используйте ключевое слово `auto`, которое указывает NAT определить порты, доступные для использования:

```
map dc0 192.168.1.0/24 -> 0/32 portmap tcp/udp auto
```

Второй метод заключается в использовании пула публичных адресов. Это полезно, когда количество локальных адресов слишком велико, чтобы уместиться в один публичный

адрес, и доступен блок публичных IP-адресов. Эти публичные адреса могут использоваться как пул, из которого NAT выбирает IP-адрес для отображения адреса пакета при его выходе.

Диапазон публичных IP-адресов может быть указан с использованием маски сети или нотации CIDR. Эти два правила эквивалентны:

```
map dc0 192.168.1.0/24 -> 204.134.75.0/255.255.255.0
map dc0 192.168.1.0/24 -> 204.134.75.0/24
```

Распространённой практикой является размещение общедоступного веб-сервера или почтового сервера в изолированном сегменте внутренней сети. Трафик с этих серверов всё равно проходит через NAT, но требуется перенаправление портов для направления входящего трафика на нужный сервер. Например, чтобы сопоставить веб-сервер с внутренним адресом `10.0.10.25` с его публичным IP-адресом `20.20.20.5`, используйте следующее правило:

```
rdr dc0 20.20.20.5/32 port 80 -> 10.0.10.25 port 80
```

Если это единственный веб-сервер, это правило также будет работать, так как оно перенаправляет все внешние HTTP-запросы на `10.0.10.25`:

```
rdr dc0 0.0.0.0/0 port 80 -> 10.0.10.25 port 80
```

В IPF встроен FTP-прокси, который можно использовать с NAT. Он отслеживает весь исходящий трафик на предмет запросов активных или пассивных FTP-соединений и динамически создаёт временные правила фильтрации, содержащие номер порта, используемого FTP-каналом данных. Это устраняет необходимость открывать большие диапазоны портов высокого порядка для FTP-соединений.

В этом примере первое правило вызывает прокси для исходящего FTP-трафика из внутренней локальной сети. Второе правило пропускает FTP-трафик из межсетевого экрана в Интернет, а третье правило обрабатывает весь не-FTP трафик из внутренней локальной сети:

```
map dc0 10.0.10.0/29 -> 0/32 proxy port 21 ftp/tcp
map dc0 0.0.0.0/0 -> 0/32 proxy port 21 ftp/tcp
map dc0 10.0.10.0/29 -> 0/32
```

Правила FTP `map` размещаются перед правилом NAT, так что при совпадении пакета с правилом FTP прокси-сервер FTP создаёт временные правила фильтрации, позволяющие пакетам FTP-сессии проходить и подвергаться NAT. Все пакеты из локальной сети, не относящиеся к FTP, не будут соответствовать правилам FTP, но подвергнутся NAT, если они соответствуют третьему правилу.

Без FTP-прокси потребуются следующие правила межсетевого экрана. Обратите внимание,

что без прокси необходимо разрешить все порты выше 1024:

```
# Allow out LAN PC client FTP to public Internet
# Active and passive modes
pass out quick on rl0 proto tcp from any to any port = 21 flags S keep state

# Allow out passive mode data channel high order port numbers
pass out quick on rl0 proto tcp from any to any port > 1024 flags S keep state

# Active mode let data channel in from FTP server
pass in quick on rl0 proto tcp from any to any port = 20 flags S keep state
```

Всякий раз, когда редактируется файл, содержащий правила NAT, выполните `ipnat` с `-CF`, чтобы удалить текущие правила NAT и очистить содержимое таблицы динамической трансляции. Используйте `-f` и укажите имя набора правил NAT для загрузки:

```
# ipnat -CF -f /etc/ipnat.rules
```

Для отображения статистики NAT:

```
# ipnat -s
```

Для отображения текущих сопоставлений таблицы NAT:

```
# ipnat -l
```

Чтобы включить подробный режим и отображать информацию, связанную с обработкой правил, активными правилами и записями таблиц:

```
# ipnat -v
```

34.5.5. Просмотр статистики IPF

В пакет IPF входит программа ipfstat(8), которую можно использовать для получения и отображения статистики, собираемой при совпадении пакетов с правилами при прохождении через межсетевой экран. Статистика накапливается с момента последнего запуска межсетевого экрана или с момента последнего сброса статистики в ноль с помощью команды `ipf -Z`.

Вывод `ipfstat` по умолчанию выглядит следующим образом:

```
input packets: blocked 99286 passed 1255609 nomatch 14686 counted 0
 output packets: blocked 4200 passed 1284345 nomatch 14687 counted 0
 input packets logged: blocked 99286 passed 0
```

```
output packets logged: blocked 0 passed 0
packets logged: input 0 output 0
log failures: input 3898 output 0
fragment state(in): kept 0 lost 0
fragment state(out): kept 0 lost 0
packet state(in): kept 169364 lost 0
packet state(out): kept 431395 lost 0
ICMP replies: 0 TCP RSTs sent: 0
Result cache hits(in): 1215208 (out): 1098963
IN Pullups succeeded: 2 failed: 0
OUT Pullups succeeded: 0 failed: 0
Fastroute successes: 0 failures: 0
TCP cksum fails(in): 0 (out): 0
Packet log flags set: (0)
```

Доступно несколько вариантов. При указании `-i` для входящего или `-o` для исходящего трафика команда получит и отобразит соответствующий список правил фильтрации, установленных и используемых ядром. Чтобы также увидеть номера правил, добавьте `-n`. Например, `ipfstat -on` выводит таблицу исходящих правил с номерами:

```
@1 pass out on xl0 from any to any
@2 block out on dc0 from any to any
@3 pass out quick on dc0 proto tcp/udp from any to any keep state
```

Включите `-h`, чтобы добавить перед каждым правилом количество его совпадений. Например, `ipfstat -oh` выводит таблицу внутренних правил для исходящего трафика, добавляя перед каждым правилом количество его использований:

```
2451423 pass out on xl0 from any to any
354727 block out on dc0 from any to any
430918 pass out quick on dc0 proto tcp/udp from any to any keep state
```

Для отображения таблицы состояний в формате, аналогичном top(1), используйте `ipfstat -t`. Когда межсетевой экран подвергается атаке, эта опция позволяет идентифицировать и просматривать атакующие пакеты. Дополнительные подфлаги дают возможность выбора IP-адреса назначения или источника, порта или протокола для мониторинга в реальном времени. Подробности смотрите в ipfstat(8).

34.5.6. Журналирование IPF

IPF предоставляет `ipmon`, который может использоваться для записи информации журналирования межсетевого экрана в удобочитаемом формате. Для этого требуется сначала добавить `options IPFILTER_LOG` в собственное ядро, следуя инструкциям в Настройка ядра FreeBSD.

Эта команда обычно запускается в режиме демона для обеспечения непрерывного ведения системного журнала, чтобы можно было просматривать записи о прошлых событиях.

Поскольку FreeBSD имеет встроенное средство syslogd(8) для автоматической ротации системных журналов, параметр `ipmon_flags` по умолчанию в rc.conf использует `-Ds`:

```
ipmon_flags="-Ds" # D = start as daemon
                  # s = log to syslog
                  # v = log tcp window, ack, seq
                  # n = map IP & port to names
```

Ведение журналов предоставляет возможность последующего просмотра информации, такой как: какие пакеты были отброшены, с каких адресов они пришли и куда направлялись. Эта информация полезна при отслеживании злоумышленников.

После включения системы журналирования в rc.conf и запуска с помощью `service ipmon start`, IPF будет записывать в журнал только те правила, которые содержат ключевое слово `log`. Администратор межсетевого экрана решает, какие правила из набора должны записываться в журнал, и обычно регистрируются только правила с deny. Обычно ключевое слово `log` включают в последнее правило набора. Это позволяет увидеть все пакеты, которые не соответствуют ни одному из правил набора.

По умолчанию режим `ipmon -Ds` использует `local0` как средство ведения журнала. Для дальнейшего разделения регистрируемых данных можно использовать следующие уровни ведения журнала:

```
LOG_INFO - packets logged using the "log" keyword as the action rather than pass or
block.
LOG_NOTICE - packets logged which are also passed
LOG_WARNING - packets logged which are also blocked
LOG_ERR - packets which have been logged and which can be considered short due to an
incomplete header
```

Чтобы настроить IPF для записи всех данных в /var/log/ipfilter.log, сначала создайте пустой файл:

```
# touch /var/log/ipfilter.log
```

Затем, чтобы записывать все журналируемые сообщения в указанный файл, добавьте следующую строку в /etc/syslog.conf:

```
local0.* /var/log/ipfilter.log
```

Для активации изменений и указания syslogd(8) прочитать изменённый файл /etc/syslog.conf, выполните команду `service syslogd reload`.

Не забудьте отредактировать /etc/newsyslog.conf для ротации нового файла журнала.

Сообщения, генерируемые `ipmon`, состоят из полей данных, разделённых пробелами. Общие

для всех сообщений поля:

1. The date of packet receipt.
2. Время получения пакета. Указывается в формате ЧЧ:ММ:СС.Д, где ЧЧ - часы, ММ - минуты, СС - секунды, а Д - доли секунды.
3. Имя интерфейса, обработавшего пакет.
4. Группа и номер правила в формате `@0:17`.
5. Действие: `p` — пропущено, `b` — заблокировано, `S` — короткий пакет, `n` — не совпало ни с одним правилом, `L` — правило для журналирования.
6. Адреса записываются в виде трёх полей: исходный адрес и порт, разделённые запятой, символ `→` и адрес назначения с портом. Например: `209.53.17.22,80 → 198.73.220.17,1722`.
7. `PR` с указанием имени или номера протокола: например, `PR tcp`.
8. `len`, за которым следует длина заголовка и общая длина пакета: например, `len 20 40`.

Если пакет является TCP-пакетом, будет дополнительное поле, начинающееся с дефиса, за которым следуют буквы, соответствующие установленным флагам. Список букв и соответствующих флагов приведён в ipf(5).

Если пакет является ICMP-пакетом, в конце будут два поля: первое всегда `icmp`, а следующее — тип ICMP-сообщения и подтип, разделённые косой чертой. Например: `icmp 3/3` для сообщения о недоступности порта.

34.6. Blocklistd

Blocklistd — это демон, который прослушивает сокеты, ожидая получения уведомлений от других демонов о неудачных или успешных попытках подключения. Он наиболее широко используется для блокировки чрезмерного количества попыток подключения к открытым портам. Типичный пример — SSH, работающий в интернете и получающий множество запросов от ботов или скриптов, пытающихся угадать пароли и получить доступ. Используя blocklistd, демон может уведомить межсетевой экран о необходимости создания правила фильтрации для блокировки чрезмерных попыток подключения с одного источника после нескольких попыток. Blocklistd был первоначально разработан в NetBSD и появился там в версии 7. FreeBSD 11 импортировал blocklistd из NetBSD (ранее он назывался blacklist).

В этой главе описывается, как настроить blocklistd, его конфигурация, а также приводятся примеры использования. Читатели должны быть знакомы с основными концепциями межсетевого экрана, такими как правила. Подробности см. в главе о межсетевом экране. В примерах используется PF, но другие межсетевые экраны, доступные в FreeBSD, также должны работать с blocklistd.

34.6.1. Включение blocklistd

Основная конфигурация для blocklistd хранится в man: blocklistd.conf[5]. Также доступны различные параметры командной строки для изменения поведения blocklistd во время выполнения. Постоянная конфигурация, сохраняемая после перезагрузок, должна храниться в /etc/blocklistd.conf. Чтобы включить демон во время загрузки системы, добавьте

строку `blocklistd_enable` в /etc/rc.conf следующим образом:

```
# sysrc blocklistd_enable=yes
```

Чтобы запустить службу вручную, выполните следующую команду:

```
# service blocklistd start
```

34.6.2. Создание набора правил для blocklistd

Правила для blocklistd настраиваются в man: blocklistd.conf[5], по одному правилу на строку. Каждое правило содержит кортеж, разделённый пробелами или табуляциями. Правила относятся либо к `local`, либо к `remote`, что применяется соответственно к машине, на которой работает blocklistd, или к внешнему источнику.

Правила local

Пример записи в blocklistd.conf для локального правила выглядит следующим образом:

```
[local]
ssh             stream  *       *               *       3       24h
```

Все правила, следующие за разделом `[local]`, рассматриваются как локальные правила (что является значением по умолчанию) и применяются к локальной машине. При обнаружении раздела `[remote]` все последующие правила обрабатываются как правила для удалённых машин.

Семь полей, разделённых табуляцией или пробелами, определяют правило. Первые четыре поля идентифицируют трафик, который должен быть внесён в список блокировки. Следующие три поля определяют поведение blocklistd. Подстановочные символы обозначаются звёздочками (`*`), которые соответствуют любому значению в данном поле. Первое поле определяет местоположение. В локальных правилах это сетевые порты. Синтаксис для поля местоположения следующий:

```
[address|interface][/mask][:port]
```

Адреса могут быть указаны в виде IPv4 в числовом формате или IPv6 в квадратных скобках. Также можно использовать имя интерфейса, например `em0`.

Тип сокета определяется вторым полем. TCP-сокеты имеют тип `stream`, тогда как UDP обозначается как `dgram`. В приведённом выше примере используется TCP, так как SSH работает по этому протоколу.

Протокол может быть указан в третьем поле правила blocklistd. Доступны следующие протоколы: `tcp`, `udp`, `tcp6`, `udp6` или числовое значение. Подстановочный символ, как в

примере, обычно используется для соответствия всем протоколам, если нет необходимости различать трафик по определённому протоколу.

В четвёртом поле определяется эффективный пользователь или владелец процесса демона, который сообщает о событии. Здесь можно использовать имя пользователя или UID, а также подстановочный знак (см. пример правила выше).

Имя правила фильтра пакетов объявляется пятым полем, которое начинает поведенческую часть правила. По умолчанию, blocklistd помещает все блокировки в якорь pf под названием `blocklistd ` в pf.conf следующим образом:

```
anchor "blocklistd/*" in on $ext_if
block in
pass out
```

Для отдельных списков блокировки в этом поле можно использовать имя якоря. В остальных случаях подойдет подстановочный знак. Если имя начинается с дефиса (`-`), это означает, что следует использовать якорь с добавленным именем правила по умолчанию. Модифицированный пример из вышеприведённого с использованием дефиса будет выглядеть так:

```
ssh             stream  *       *               -ssh       3       24h
```

С таким правилом любые новые правила из списка блокировки добавляются к якорю с именем `blocklistd-ssh`.

Для блокировки целых подсетей за одно нарушение правила можно использовать символ `/` в имени правила. Это приводит к тому, что оставшаяся часть имени интерпретируется как маска, применяемая к адресу, указанному в правиле. Например, это правило заблокирует все адреса, входящие в подсеть `/24`.

```
22              stream  tcp       *               */24    3       24h
```

Здесь важно указать правильный протокол. IPv4 и IPv6 обрабатывают /24 по-разному, поэтому `*` нельзя использовать в третьем поле для этого правила.

Это правило определяет, что если любой хост в этой сети ведёт себя неправильно, всё остальное в этой сети также будет заблокировано.

Шестое поле, называемое `nfail`, устанавливает количество неудачных попыток входа, необходимых для внесения IP-адреса внешнего хоста в чёрный список. Если в этой позиции используется подстановочный знак, это означает, что блокировки никогда не будут происходить. В приведённом выше примере правила установлен лимит в три попытки, что означает, что после трёх попыток входа через SSH с одного соединения IP-адрес будет заблокирован.

Последнее поле в определении правила blocklistd указывает, как долго хост будет находиться в чёрном списке. По умолчанию единицей измерения являются секунды, но также можно указать суффиксы `m`, `h` и `d` для минут, часов и дней соответственно.

Пример правила в полном объёме означает, что после трёх попыток аутентификации по SSH будет создано новое правило блокировки PF для этого хоста. Совпадения правил проверяются путем последовательной проверки локальных правил от наиболее специфичных к наименее специфичным. Когда совпадение найдено, применяются правила `remote`, а поля `name`, `nfail` и `duration` изменяются в соответствии с совпавшим правилом `remote`.

Правила remote

Удалённые правила используются для указания того, как blocklistd изменяет своё поведение в зависимости от удалённого хоста, который в данный момент оценивается. Каждое поле в удалённом правиле совпадает с таковым в локальном правиле. Единственное различие заключается в том, как blocklistd их использует. Для объяснения используется следующее примерное правило:

```
[remote]
203.0.113.128/25 *      *       *               =/25    =       48h
```

Поле `address` может содержать IP-адрес (как v4, так и v6), порт или оба варианта. Это позволяет задавать специальные правила для определённого диапазона удалённых адресов, как в этом примере. Поля для типа сокета, протокола и владельца интерпретируются так же, как в локальном правиле.

Поле name отличается: знак равенства (`=`) в удалённом правиле указывает blocklistd использовать значение из соответствующего локального правила. Это означает, что запись правила межсетевого экрана берется, и добавляется префикс `/25` (маска сети `255.255.255.128`). Когда соединение из этого диапазона адресов попадает в черный список, затрагивается вся подсеть. Здесь также можно использовать имя якоря PF, и в этом случае blocklistd добавит правила для этого блока адресов в якорь с указанным именем. Если указана подстановка, используется таблица по умолчанию.

В столбце `nfail` можно задать произвольное количество неудачных попыток для адреса. Это полезно для исключений из конкретного правила, например, чтобы разрешить кому-то менее строгое применение правил или немного больше попыток входа. Блокировка отключается, если в шестом поле указана звёздочка.

Удалённые правила позволяют более строго ограничивать попытки входа по сравнению с попытками, поступающими из локальной сети, например, из офиса.

34.6.3. Конфигурация клиента blocklistd

В FreeBSD есть несколько программных пакетов, которые могут использовать функциональность blocklistd. Два наиболее заметных — это ftp/freebsd-ftpd и sshd(8), предназначенные для блокировки чрезмерных попыток подключения. Чтобы активировать blocklistd в демоне SSH, добавьте следующую строку в /etc/ssh/sshd_config:

```
UseBlocklist yes
```

Перезапустите sshd, чтобы изменения вступили в силу.

Список блокировки для ftp/freebsd-ftpd включается с помощью `-B`, либо в /etc/inetd.conf, либо как флаг в /etc/rc.conf следующим образом:

```
ftpd_flags="-B"
```

Вот и всё, что требуется для настройки взаимодействия этих программ с blocklistd.

34.6.4. Управление blocklistd

Blocklistd предоставляет пользователю утилиту управления под названием blocklistctl(8). Она отображает заблокированные адреса и сети, которые внесены в список блокировки согласно правилам, определённым в blacklistd.conf(5). Чтобы просмотреть список текущих заблокированных хостов, используйте команду `dump` с параметром `-b`, например.

```
# blocklistctl dump -b
rulename                address/ma:port id      nfail   last access
blocklistd        213.0.123.128/25:22   OK      6/3     2019/06/08 14:30:19
```

Этот пример показывает, что было 6 попыток из трёх разрешённых на порт 22, исходящих из диапазона адресов `213.0.123.128/25`. Количество попыток превышает разрешённое, потому что SSH позволяет клиенту выполнять несколько попыток входа в рамках одного TCP-соединения. Текущее соединение не прерывается blocklistd. Последняя попытка соединения указана в столбце `last access` вывода.

Чтобы увидеть оставшееся время, в течение которого этот хост будет в списке блокировки, добавьте `-r` к предыдущей команде.

```
# blocklistctl dump -br
rulename                address/ma:port id      nfail   remaining time
blocklistd        213.0.123.128/25:22   OK      6/3     36s
```

В этом примере осталось 36 секунд, пока этот хост не будет разблокирован.

34.6.5. Удаление узлов из списка блокировки

Иногда необходимо удалить узел из чёрного списка до истечения оставшегося времени. К сожалению, в blocklistd нет функциональности для этого. Однако можно удалить адрес из таблицы PF с помощью pfctl. Для каждого заблокированного порта существует дочерний якорь внутри якоря blocklistd, определённого в /etc/pf.conf. Например, если есть дочерний якорь для блокировки порта 22, он называется `blocklistd/22`. Внутри этого дочернего якоря находится таблица, содержащая заблокированные адреса. Эта таблица называется port с

указанием номера порта. В данном примере она будет называться `port22`. Имея эту информацию, можно использовать pfctl(8) для отображения всех перечисленных адресов следующим образом:

```
# pfctl -a blocklistd/22 -t port22 -T show
...
213.0.123.128/25
...
```

После определения адреса, который нужно разблокировать из списка, следующая команда удаляет его из списка:

```
# pfctl -a blocklistd/22 -t port22 -T delete 213.0.123.128/25
```

Адрес теперь удалён из PF, но всё ещё будет отображаться в списке blocklistctl, так как он не знает о внесённых в PF изменениях. Запись в базе данных blocklistd со временем истечёт и будет удалена из вывода. Запись будет добавлена снова, если хост снова совпадёт с одним из правил блокировки в blocklistd.

Глава 35. Сложные вопросы работы в сети

35.1. Обзор

Эта глава охватывает ряд сложных тем, связанных с сетями.

Прочтите эту главу, чтобы узнать:

- Основы шлюзов и маршрутов.
- Как настроить USB-тетеринг.
- Как настроить устройства IEEE® 802.11 и Bluetooth®.
- Как сделать так, чтобы система FreeBSD работала как мост.
- Как настроить загрузку системы из сети с помощью PXE.
- Как включить и использовать возможности протокола Common Address Redundancy Protocol (CARP) в FreeBSD.
- Как настроить несколько VLAN в FreeBSD.
- Как настроить гарнитуру Bluetooth.

Прежде чем читать эту главу, необходимо:

- Понимать основы скриптов /etc/rc.
- Знать основные термины и понятия сетевых технологий.
- Понимать базовые настройки сети в FreeBSD (Сеть FreeBSD).
- Знать, как настроить и установить новое ядро FreeBSD (Настройка ядра FreeBSD).
- Знать, как устанавливать дополнительное стороннее программное обеспечение (Установка приложений: Пакеты и Порты).

35.2. Шлюзы и маршруты

Маршрутизация — это механизм, позволяющий системе находить сетевой путь к другой системе. *Маршрут* — это определённая пара адресов, представляющих "назначение" и "шлюз". Маршрут указывает, что при попытке достичь указанного назначения пакеты должны отправляться через указанный шлюз. Существует три типа назначений: отдельные хосты, подсети и "маршрут по умолчанию". "Маршрут по умолчанию" используется, если не подходит ни один другой маршрут. Также существует три типа шлюзов: отдельные хосты, интерфейсы (также называемые линками) и аппаратные (MAC) адреса Ethernet. Известные маршруты хранятся в таблице маршрутизации.

В этом разделе представлен обзор основ маршрутизации. Затем показано, как настроить систему FreeBSD в качестве маршрутизатора, и даны некоторые советы по устранению неполадок.

35.2.1. Основы маршрутизации

Для просмотра таблицы маршрутизации системы FreeBSD используйте netstat(1):

```
% netstat -r
Routing tables

Internet:
Destination      Gateway            Flags     Refs      Use     Netif Expire
default          outside-gw         UGS         37      418       em0
localhost        localhost          UH           0      181       lo0
test0            0:e0:b5:36:cf:4f   UHLW         5    63288       re0     77
10.20.30.255     link#1             UHLW         1     2421
example.com      link#1             UC           0        0
host1            0:e0:a8:37:8:1e    UHLW         3     4601       lo0
host2            0:e0:a8:37:8:1e    UHLW         0        5       lo0 =>
host2.example.com link#1            UC           0        0
224              link#1             UC           0        0
```

Записи в этом примере следующие:

default

Первый маршрут в этой таблице указывает маршрут по умолчанию (`default`). Когда локальной системе требуется установить соединение с удалённым узлом, она проверяет таблицу маршрутизации, чтобы определить, существует ли известный путь. Если удалённый узел соответствует записи в таблице, система проверяет, может ли она подключиться, используя интерфейс, указанный в этой записи.

Если назначение не соответствует ни одной записи или если все известные пути недоступны, система использует запись для маршрута по умолчанию. Для хостов в локальной сети поле `Gateway` в маршруте по умолчанию указывает на систему, имеющую прямое подключение к Интернету. При чтении этой записи убедитесь, что столбец `Flags` указывает на то, что шлюз доступен (`UG`).

Маршрут по умолчанию для машины, которая сама функционирует как шлюз во внешний мир, будет шлюзом провайдера интернет-услуг (ISP).

localhost

Второй маршрут — это маршрут `localhost`. Интерфейс, указанный в столбце `Netif` для `localhost`, — это lo0, также известное как loopback-устройство. Это означает, что весь трафик для этого назначения должен быть внутренним, а не отправляться через сеть.

MAC адрес

Адреса, начинающиеся с `0:e0:`, являются MAC-адресами. FreeBSD автоматически определит любые хосты, например `test0`, в локальной сети Ethernet и добавит маршрут для этого хоста через интерфейс Ethernet re0. Такой маршрут имеет время жизни, указанное в столбце `Expire`, которое используется, если хост не отвечает в течение определённого времени. В этом случае маршрут к этому хосту будет автоматически

удалён. Эти хосты определяются с помощью Протокола маршрутной информации (RIP — Routing Information Protocol), который вычисляет маршруты к локальным хостам на основе определения кратчайшего пути.

subnet

FreeBSD автоматически добавит маршруты для локальной подсети. В этом примере `10.20.30.255` — это широковещательный адрес для подсети `10.20.30`, а `example.com` — доменное имя, связанное с этой подсетью. Обозначение `link#1` относится к первой Ethernet-карте в машине.

Локальные хосты сети и локальные подсети автоматически получают маршруты через демон routed(8). Если он не запущен, будут существовать только маршруты, статически определённые администратором.

host

Строка `host1` ссылается на хост по его Ethernet-адресу. Поскольку это отправляющий хост, FreeBSD использует loopback-интерфейс (lo0) вместо Ethernet-интерфейса.

Две строки `host2` представляют собой псевдонимы, созданные с помощью ifconfig(8). Символ ⇒ после интерфейса lo0 указывает, что помимо loopback-адреса был установлен псевдоним. Такие маршруты отображаются только на хосте, поддерживающем псевдоним, а все остальные хосты в локальной сети будут иметь строку `link#1` для таких маршрутов.

224

Последняя строка (подсеть назначения `224`) относится к многоадресной рассылке.

Различные атрибуты каждого маршрута можно увидеть в столбце `Flags`. Часто встречающиеся флаги таблицы маршрутизации содержит сводку некоторых из этих флагов и их значений:

Таблица 52. Часто встречающиеся флаги таблицы маршрутизации

Flag	Назначение
U	Маршрут активен (поднят).
H	Целью маршрута является отдельный хост.
G	Отправляйте всё для этого назначения на этот шлюз, который разберётся, куда это нужно отправить.
S	Этот маршрут был настроен статически.
C	Клонирует новый маршрут на основе данного для подключения машин. Такой тип маршрута обычно используется для локальных сетей.
W	Маршрут был автоматически настроен на основе локальной сети (клон) маршрута.

Flag	Назначение
L	Маршрут включает ссылки на оборудование Ethernet (link).

На системе FreeBSD маршрут по умолчанию может быть определён в /etc/rc.conf путём указания IP-адреса шлюза по умолчанию:

```
defaultrouter="10.20.30.1"
```

Также можно вручную добавить маршрут с помощью `route`:

```
# route add default 10.20.30.1
```

Обратите внимание, что вручную добавленные маршруты не сохранятся после перезагрузки. Для получения дополнительной информации о ручном управлении таблицами сетевой маршрутизации обратитесь к route(8).

35.2.2. Настройка маршрутизатора со статическими маршрутами

Система FreeBSD может быть настроена как шлюз по умолчанию или маршрутизатор для сети, если она является двухдоменной системой. Двухдоменная система — это хост, который находится как минимум в двух разных сетях. Обычно каждая сеть подключена к отдельному сетевому интерфейсу, хотя IP-алиасинг может использоваться для привязки нескольких адресов, каждый в своей подсети, к одному физическому интерфейсу.

Для того чтобы система могла пересылать пакеты между интерфейсами, FreeBSD должна быть настроена как маршрутизатор. Интернет-стандарты и лучшие инженерные практики не позволяют проекту FreeBSD включать эту функцию по умолчанию, но её можно настроить для запуска при загрузке, добавив следующую строку в /etc/rc.conf:

```
gateway_enable="YES"          # Set to YES if this host will be a gateway
```

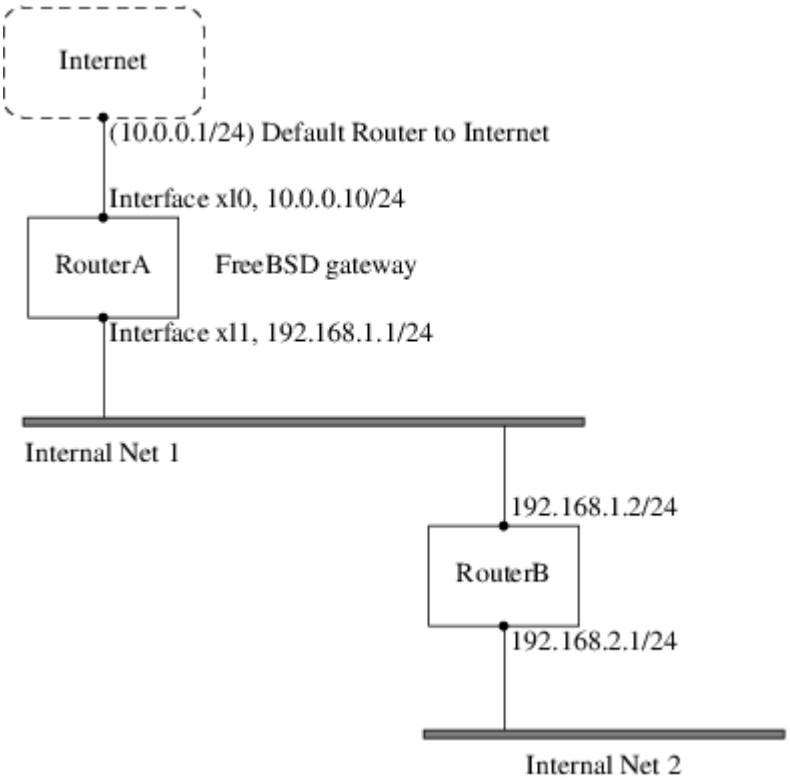
Чтобы теперь включить маршрутизацию, установите переменную sysctl(8) `net.inet.ip.forwarding` в значение `1`. Для отключения маршрутизации сбросьте эту переменную в `0`.

Таблица маршрутизации маршрутизатора требует дополнительных маршрутов, чтобы он знал, как достичь других сетей. Маршруты могут быть добавлены вручную с использованием статических маршрутов или могут быть автоматически созданы обучением с помощью протокола маршрутизации. Статические маршруты подходят для небольших сетей, и в этом разделе описывается, как добавить запись статической маршрутизации для небольшой сети.

Для больших сетей статические маршруты быстро становятся неэффективными. FreeBSD включает стандартный демон маршрутизации

> routed(8), который поддерживает протоколы RIP версий 1 и 2, а также IRDP. Поддержка протоколов маршрутизации BGP и OSPF может быть установлена с помощью пакета net/quagga или порта.

Рассмотрим следующую сеть:



В этом сценарии `RouterA` — это машина FreeBSD, которая выступает в качестве маршрутизатора для остальной части Интернета. У неё установлен маршрут по умолчанию на `10.0.0.1`, что позволяет ей соединяться с внешним миром. `RouterB` уже настроен на использование `192.168.1.1` в качестве шлюза по умолчанию.

Прежде чем добавлять статические маршруты, таблица маршрутизации на `RouterA` выглядит следующим образом:

```
% netstat -nr
Routing tables

Internet:
Destination        Gateway            Flags    Refs      Use  Netif  Expire
default            10.0.0.1           UGS         0    49378    xl0
127.0.0.1          127.0.0.1          UH          0        6    lo0
10.0.0.0/24        link#1             UC          0        0    xl0
192.168.1.0/24     link#2             UC          0        0    xl1
```

С текущей таблицей маршрутизации `RouterA` не имеет маршрута к сети `192.168.2.0/24`. Следующая команда добавляет сеть `Internal Net 2` в таблицу маршрутизации `RouterA`, используя `192.168.1.2` в качестве следующего прыжка:

```
# route add -net 192.168.2.0/24 192.168.1.2
```

Теперь RouterA может достигать любого узла в сети 192.168.2.0/24. Однако информация о маршрутизации не сохранится после перезагрузки системы FreeBSD. Если требуется, чтобы статический маршрут был постоянным, добавьте его в /etc/rc.conf:

```
# Add Internal Net 2 as a persistent static route
static_routes="internalnet2"
route_internalnet2="-net 192.168.2.0/24 192.168.1.2"
```

Переменная конфигурации static_routes представляет собой список строк, разделённых пробелом, где каждая строка ссылается на имя маршрута. Переменная route_internalnet2 содержит статический маршрут для этого имени маршрута.

Использование более одной строки в static_routes создаёт несколько статических маршрутов. Ниже приведён пример добавления статических маршрутов для сетей 192.168.0.0/24 и 192.168.1.0/24:

```
static_routes="net1 net2"
route_net1="-net 192.168.0.0/24 192.168.0.1"
route_net2="-net 192.168.1.0/24 192.168.1.1"
```

35.2.3. Устранение неполадок

Когда адресное пространство назначается сети, поставщик услуг настраивает свои таблицы маршрутизации так, чтобы весь трафик для сети отправлялся по каналу связи к сайту. Но как внешние сайты узнают, что их пакеты нужно отправлять к межсетевому экрану провайдера сети?

Существует система, которая отслеживает все выделенные адресные пространства и определяет их точку подключения к магистрали Интернета или основным магистральным линиям, передающим интернет-трафик по стране и по всему миру. Каждая машина магистрали имеет копию главного набора таблиц, которые направляют трафик для определённой сети к конкретному магистральному оператору, а оттуда по цепочке поставщиков услуг, пока он не достигнет конкретной сети.

Это задача поставщика услуг — сообщить магистральным узлам, что они являются точкой подключения и, следовательно, путем внутрь для сайта. Это известно как распространение маршрутов.

Иногда возникают проблемы с распространением маршрутов, и некоторые сайты не могут подключиться. Возможно, наиболее полезная команда для выяснения, где происходит разрыв маршрутизации, — это traceroute. Она полезна, когда ping не срабатывает.

При использовании traceroute укажите адрес удалённого хоста для подключения. В выводе будут показаны шлюзы на пути попытки соединения, в конечном итоге достигая целевого

хоста или прерываясь из-за отсутствия соединения. Для получения дополнительной информации обратитесь к traceroute(8).

35.2.4. Аспекты многоадресной рассылки (multicast)

FreeBSD изначально поддерживает как приложения с многоадресной рассылкой, так и маршрутизацию многоадресной рассылки. Для работы приложений с многоадресной рассылкой на FreeBSD не требуется специальной настройки. Для поддержки маршрутизации многоадресной рассылки необходимо включить следующую опцию в собственном ядре:

```
options MROUTING
```

Демон маршрутизации многоадресной рассылки, mrouted, может быть установлен с помощью пакета net/mrouted или порта. Этот демон реализует протокол маршрутизации многоадресной рассылки DVMRP и настраивается путём редактирования файла /usr/local/etc/mrouted.conf для настройки туннелей и DVMRP. Установка mrouted также устанавливает map-mbone и mrinfo, а также связанные с ними man-страницы. Обратитесь к ним за примерами конфигурации.

DVMRP во многом заменён протоколом PIM во многих инсталляциях с использованием многоадресной рассылки. Дополнительную информацию можно найти в pim(4).

35.3. Виртуальные узлы

Распространённое использование FreeBSD — это виртуальный хостинг сайтов, когда один сервер представляется в сети как множество серверов. Это достигается путём назначения нескольких сетевых адресов одному интерфейсу.

Указанный сетевой интерфейс имеет один "реальный" адрес и может иметь любое количество "псевдонимных" адресов. Эти псевдонимы обычно добавляются путём размещения записей alias в /etc/rc.conf, как показано в этом примере:

```
# sysrc ifconfig_fxp0_alias0="inet xxx.xxx.xxx.xxx netmask xxx.xxx.xxx.xxx"
```

Записи псевдонимов должны начинаться с `alias0`, используя последовательные числа, такие как `alias0`, `alias1` и так далее. Процесс настройки остановится при первом пропущенном числе.

Расчёт масок подсети для псевдонимов важен. Для заданного интерфейса должен быть один адрес, который корректно представляет маску подсети сети. Любые другие адреса, попадающие в эту сеть, должны иметь маску подсети, состоящую из всех `1`, выраженную как `255.255.255.255` или `0xffffffff`.

Например, рассмотрим случай, когда интерфейс `fxp0` подключён к двум сетям: `10.1.1.0` с

маской сети 255.255.255.0 и 202.0.75.16 с маской сети 255.255.255.240. Система должна быть настроена так, чтобы находиться в диапазонах 10.1.1.1–10.1.1.5 и 202.0.75.17–202.0.75.20. Только первый адрес в каждом диапазоне должен иметь реальную маску сети. Все остальные (10.1.1.2–10.1.1.5 и 202.0.75.18–202.0.75.20) должны быть настроены с маской 255.255.255.255.

Для данного сценария правильно настраивают адаптер следующие записи в /etc/rc.conf :

```
# sysrc ifconfig_fxp0="inet 10.1.1.1 netmask 255.255.255.0"
# sysrc ifconfig_fxp0_alias0="inet 10.1.1.2 netmask 255.255.255.255"
# sysrc ifconfig_fxp0_alias1="inet 10.1.1.3 netmask 255.255.255.255"
# sysrc ifconfig_fxp0_alias2="inet 10.1.1.4 netmask 255.255.255.255"
# sysrc ifconfig_fxp0_alias3="inet 10.1.1.5 netmask 255.255.255.255"
# sysrc ifconfig_fxp0_alias4="inet 202.0.75.17 netmask 255.255.255.240"
# sysrc ifconfig_fxp0_alias5="inet 202.0.75.18 netmask 255.255.255.255"
# sysrc ifconfig_fxp0_alias6="inet 202.0.75.19 netmask 255.255.255.255"
# sysrc ifconfig_fxp0_alias7="inet 202.0.75.20 netmask 255.255.255.255"
```

Более простой способ выразить это — использовать список диапазонов IP-адресов, разделённых пробелами. Первому адресу будет назначена указанная маска подсети, а дополнительным адресам — маска подсети 255.255.255.255.

```
# sysrc ifconfig_fxp0_aliases="inet 10.1.1.1-5/24 inet 202.0.75.17-20/28"
```

35.4. Расширенная аутентификация в беспроводной сети

FreeBSD поддерживает различные способы подключения к беспроводной сети. В этом разделе описано, как выполнить расширенную аутентификацию в беспроводной сети.

Для подключения и базовой аутентификации в беспроводной сети раздел Подключение и аутентификация в беспроводной сети в главе "Сеть" описывает, как это сделать.

35.4.1. WPA с EAP-TLS

Второй способ использования WPA — с сервером аутентификации 802.1X. В этом случае WPA называется WPA Enterprise, чтобы отличать его от менее безопасного WPA Personal. Аутентификация в WPA Enterprise основана на расширяемом протоколе аутентификации (EAP).

EAP не включает в себя метод шифрования. Вместо этого EAP встраивается в зашифрованный туннель. Существует множество методов аутентификации EAP, но наиболее распространены EAP-TLS, EAP-TTLS и EAP-PEAP.

EAP с защитой на транспортном уровне (EAP-TLS) — это широко поддерживаемый протокол аутентификации беспроводных сетей, так как он был первым методом EAP,

сертифицированным Альянсом Wi-Fi. Для работы EAP-TLS требуется три сертификата: сертификат центра сертификации (CA), установленный на всех машинах, сертификат сервера для сервера аутентификации и один клиентский сертификат для каждого беспроводного клиента. В этом методе EAP и сервер аутентификации, и беспроводной клиент аутентифицируют друг друга, предоставляя свои соответствующие сертификаты, а затем проверяют, что эти сертификаты были подписаны CA организации.

Как и ранее, настройка выполняется через /etc/wpa_supplicant.conf:

```
network={
  ssid="freebsdap" ①
  proto=RSN  ②
  key_mgmt=WPA-EAP ③
  eap=TLS ④
  identity="loader" ⑤
  ca_cert="/etc/certs/cacert.pem" ⑥
  client_cert="/etc/certs/clientcert.pem" ⑦
  private_key="/etc/certs/clientkey.pem" ⑧
  private_key_passwd="freebsdmallclient" ⑨
}
```

① Это поле указывает имя сети (SSID).

② Этот пример использует протокол RSN IEEE® 802.11i, также известный как WPA2.

③ Строка `key_mgmt` указывает на используемый протокол управления ключами. В данном примере это WPA с аутентификацией EAP.

④ Это поле указывает метод EAP для подключения.

⑤ Поле `identity` содержит строку идентификации для EAP.

⑥ Поле `ca_cert` указывает путь к файлу сертификата CA. Этот файл необходим для проверки сертификата сервера.

⑦ Строка `client_cert` указывает путь к файлу сертификата клиента. Этот сертификат уникален для каждого беспроводного клиента в сети.

⑧ Поле `private_key` содержит путь к файлу закрытого ключа клиентского сертификата.

⑨ Поле `private_key_passwd` содержит парольную фразу для закрытого ключа.

Затем добавьте следующие строки в /etc/rc.conf:

```
wlans_ath0="wlan0"
ifconfig_wlan0="WPA DHCP"
```

Следующий шаг — поднять интерфейс:

```
# service netif start
Starting wpa_supplicant.
DHCPREQUEST on wlan0 to 255.255.255.255 port 67 interval 7
```

```
DHCPREQUEST on wlan0 to 255.255.255.255 port 67 interval 15
DHCPACK from 192.168.0.20
bound to 192.168.0.254 -- renewal in 300 seconds.
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
      ether 00:11:95:d5:43:62
      inet 192.168.0.254 netmask 0xffffff00 broadcast 192.168.0.255
      media: IEEE 802.11 Wireless Ethernet DS/11Mbps mode 11g
      status: associated
      ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
      country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
      AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
      bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
      wme burst roaming MANUAL
```

Также можно поднять интерфейс вручную с помощью wpa_supplicant(8) и ifconfig(8).

35.4.2. WPA с EAP-TTLS

С EAP-TLS и сервер аутентификации, и клиент нуждаются в сертификате. С EAP-TTLS сертификат клиента необязателен. Этот метод аналогичен веб-серверу, который создаёт защищённый SSL-туннель, даже если у посетителей нет клиентских сертификатов. EAP-TTLS использует зашифрованный TLS-туннель для безопасной передачи данных аутентификации.

Требуемая конфигурация может быть добавлена в /etc/wpa_supplicant.conf:

```
network={
  ssid="freebsdap"
  proto=RSN
  key_mgmt=WPA-EAP
  eap=TTLS ①
  identity="test" ②
  password="test" ③
  ca_cert="/etc/certs/cacert.pem" ④
  phase2="auth=MD5" ⑤
}
```

① Это поле определяет метод EAP для подключения.

② Поле `identity` содержит строку идентификации для аутентификации EAP внутри зашифрованного TLS-туннеля.

③ Поле `password` содержит парольную фразу для аутентификации EAP.

④ Поле `ca_cert` указывает путь к файлу сертификата CA. Этот файл необходим для проверки сертификата сервера.

⑤ Это поле определяет метод аутентификации, используемый в зашифрованном TLS-туннеле. В данном примере используется EAP с MD5-Challenge. Фаза "внутренней аутентификации" часто называется "phase2".

Далее добавьте следующие строки в /etc/rc.conf:

```
wlans_ath0="wlan0"
ifconfig_wlan0="WPA DHCP"
```

Следующий шаг — поднять интерфейс:

```
# service netif start
Starting wpa_supplicant.
DHCPREQUEST on wlan0 to 255.255.255.255 port 67 interval 7
DHCPREQUEST on wlan0 to 255.255.255.255 port 67 interval 15
DHCPREQUEST on wlan0 to 255.255.255.255 port 67 interval 21
DHCPACK from 192.168.0.20
bound to 192.168.0.254 -- renewal in 300 seconds.
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
      ether 00:11:95:d5:43:62
      inet 192.168.0.254 netmask 0xffffff00 broadcast 192.168.0.255
      media: IEEE 802.11 Wireless Ethernet DS/11Mbps mode 11g
      status: associated
      ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
      country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
      AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
      bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
      wme burst roaming MANUAL
```

35.4.3. WPA с EAP-PEAP

PEAPv0/EAP-MSCHAPv2 является наиболее распространённым методом PEAP. В этой главе термин PEAP используется для обозначения данного метода.

Защищённый EAP (PEAP) разработан как альтернатива EAP-TTLS и является наиболее используемым стандартом EAP после EAP-TLS. В сети с разными операционными системами PEAP должен быть наиболее поддерживаемым стандартом после EAP-TLS.

PEAP аналогичен EAP-TTLS, так как использует сертификат на стороне сервера для аутентификации клиентов путем создания зашифрованного TLS-туннеля между клиентом и сервером аутентификации, что защищает последующий обмен аутентификационной информацией. Аутентификация PEAP отличается от EAP-TTLS тем, что передаёт имя пользователя в открытом виде, и только пароль отправляется в зашифрованном TLS-туннеле. EAP-TTLS использует TLS-туннель как для имени пользователя, так и для пароля.

Добавьте следующие строки в /etc/wpa_supplicant.conf для настройки параметров, связанных с EAP-PEAP:

```
network={
  ssid="freebsdap"
  proto=RSN
```

```
  key_mgmt=WPA-EAP
  eap=PEAP ①
  identity="test" ②
  password="test" ③
  ca_cert="/etc/certs/cacert.pem" ④
  phase1="peaplabel=0" ⑤
  phase2="auth=MSCHAPV2" ⑥
}
```

① Это поле определяет метод EAP для подключения.

② Поле `identity` содержит строку идентификации для аутентификации EAP внутри зашифрованного TLS-туннеля.

③ Поле `password` содержит парольную фразу для аутентификации EAP.

④ Поле `ca_cert` указывает путь к файлу сертификата CA. Этот файл необходим для проверки сертификата сервера.

⑤ Это поле содержит параметры для первой фазы аутентификации, TLS-туннеля. В зависимости от используемого сервера аутентификации укажите конкретную метку для аутентификации. В большинстве случаев меткой будет "client EAP encryption", которая устанавливается с помощью `peaplabel=0`. Дополнительную информацию можно найти в wpa_supplicant.conf(5).

⑥ Это поле определяет протокол аутентификации, используемый в зашифрованном TLS-туннеле. В случае PEAP, это `auth=MSCHAPV2`.

Добавьте следующее в /etc/rc.conf:

```
wlans_ath0="wlan0"
ifconfig_wlan0="WPA DHCP"
```

Затем поднимите интерфейс:

```
# service netif start
Starting wpa_supplicant.
DHCPREQUEST on wlan0 to 255.255.255.255 port 67 interval 7
DHCPREQUEST on wlan0 to 255.255.255.255 port 67 interval 15
DHCPREQUEST on wlan0 to 255.255.255.255 port 67 interval 21
DHCPACK from 192.168.0.20
bound to 192.168.0.254 -- renewal in 300 seconds.
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
      ether 00:11:95:d5:43:62
      inet 192.168.0.254 netmask 0xffffff00 broadcast 192.168.0.255
      media: IEEE 802.11 Wireless Ethernet DS/11Mbps mode 11g
      status: associated
      ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
      country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
      AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
      bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
```

```
        wme burst roaming MANUAL
```

35.5. Беспроводное соединение в режиме Ad-hoc

Режим IBSS, также называемый ad-hoc режимом, предназначен для соединений точка-точка. Например, чтобы создать ad-hoc сеть между машинами A и B, выберите два IP-адреса и SSID.

На A:

```
# ifconfig wlan0 create wlandev ath0 wlanmode adhoc
# ifconfig wlan0 inet 192.168.0.1 netmask 255.255.255.0 ssid freebsdap
# ifconfig wlan0
  wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
      ether 00:11:95:c3:0d:ac
      inet 192.168.0.1 netmask 0xffffff00 broadcast 192.168.0.255
      media: IEEE 802.11 Wireless Ethernet autoselect mode 11g <adhoc>
      status: running
      ssid freebsdap channel 2 (2417 Mhz 11g) bssid 02:11:95:c3:0d:ac
      country US ecm authmode OPEN privacy OFF txpower 21.5 scanvalid 60
      protmode CTS wme burst
```

Параметр adhoc указывает, что интерфейс работает в режиме IBSS.

B теперь должен иметь возможность обнаруживать A:

```
# ifconfig wlan0 create wlandev ath0 wlanmode adhoc
# ifconfig wlan0 up scan
  SSID/MESH ID    BSSID              CHAN RATE   S:N     INT CAPS
  freebsdap       02:11:95:c3:0d:ac    2   54M -64:-96  100 IS   WME
```

I в выводе подтверждает, что A находится в режиме ad-hoc. Теперь настройте B с другим IP-адресом:

```
# ifconfig wlan0 inet 192.168.0.2 netmask 255.255.255.0 ssid freebsdap
# ifconfig wlan0
  wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
      ether 00:11:95:d5:43:62
      inet 192.168.0.2 netmask 0xffffff00 broadcast 192.168.0.255
      media: IEEE 802.11 Wireless Ethernet autoselect mode 11g <adhoc>
      status: running
      ssid freebsdap channel 2 (2417 Mhz 11g) bssid 02:11:95:c3:0d:ac
      country US ecm authmode OPEN privacy OFF txpower 21.5 scanvalid 60
      protmode CTS wme burst
```

Оба A и B теперь готовы обмениваться информацией.

35.5.1. Хост FreeBSD в роли точки доступа

FreeBSD может функционировать как точка доступа (AP), что устраняет необходимость покупки аппаратной точки доступа или организации ad-hoc сети. Это может быть особенно полезно, когда машина FreeBSD выступает в качестве шлюза к другой сети, например, к Интернету.

Основные настройки

Прежде чем настраивать машину FreeBSD в качестве точки доступа, ядро должно быть сконфигурировано с соответствующей поддержкой сети для беспроводной карты, а также используемых протоколов безопасности. Для получения дополнительной информации см. Базовые настройки.

Драйвер-оболочка NDIS для драйверов Windows® в настоящее время не поддерживает работу в режиме точки доступа. Только родные беспроводные драйверы FreeBSD поддерживают режим AP.

После загрузки поддержки беспроводной сети проверьте, поддерживает ли беспроводное устройство режим точки доступа на основе хоста, также известный как режим hostap:

```
# ifconfig wlan0 create wlandev ath0
# ifconfig wlan0 list caps
drivercaps=6f85edc1<STA,FF,TURBOP,IBSS,HOSTAP,AHDEMO,TXPMGT,SHSLOT,SHPREAMBLE,MONITOR,
MBSS,WPA1,WPA2,BURST,WME,WDS,BGSCAN,TXFRAG>
cryptocaps=1f<WEP,TKIP,AES,AES_CCM,TKIPMIC>
```

Этот вывод показывает возможности карты. Слово HOSTAP подтверждает, что эта беспроводная карта может работать как точка доступа. Также перечислены различные поддерживаемые алгоритмы шифрования: WEP, TKIP и AES. Эта информация указывает, какие протоколы безопасности можно использовать на точке доступа.

Беспроводное устройство можно перевести в режим hostap только во время создания сетевого псевдоустройства, поэтому ранее созданное устройство необходимо сначала удалить:

```
# ifconfig wlan0 destroy
```

затем повторно создать с правильной опцией перед установкой остальных параметров:

```
# ifconfig wlan0 create wlandev ath0 wlanmode hostap
# ifconfig wlan0 inet 192.168.0.1 netmask 255.255.255.0 ssid freebsdap mode 11g
channel 1
```

Используйте ifconfig(8) снова, чтобы посмотреть состояние интерфейса wlan0:

```
# ifconfig wlan0
  wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
      ether 00:11:95:c3:0d:ac
      inet 192.168.0.1 netmask 0xffffff00 broadcast 192.168.0.255
      media: IEEE 802.11 Wireless Ethernet autoselect mode 11g <hostap>
      status: running
      ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
      country US ecm authmode OPEN privacy OFF txpower 21.5 scanvalid 60
      protmode CTS wme burst dtimperiod 1 -dfs
```

Параметр `hostap` указывает, что интерфейс работает в режиме точки доступа на основе хоста.

Настройка интерфейса может быть выполнена автоматически при загрузке, если добавить следующие строки в /etc/rc.conf:

```
wlans_ath0="wlan0"
create_args_wlan0="wlanmode hostap"
ifconfig_wlan0="inet 192.168.0.1 netmask 255.255.255.0 ssid freebsdap mode 11g channel
1"
```

Точка доступа на основе хоста без аутентификации или шифрования

Хотя не рекомендуется запускать точку доступа без какой-либо аутентификации или шифрования, это простой способ проверить, работает ли точка доступа. Такая конфигурация также важна для отладки проблем с клиентами.

После настройки точки доступа выполните сканирование с другого беспроводного устройства для её обнаружения:

```
# ifconfig wlan0 create wlandev ath0
# ifconfig wlan0 up scan
SSID/MESH ID    BSSID              CHAN RATE   S:N     INT CAPS
freebsdap       00:11:95:c3:0d:ac    1   54M -66:-96  100 ES   WME
```

Клиентская машина обнаружила точку доступа и может быть ассоциирована с ней:

```
# ifconfig wlan0 inet 192.168.0.2 netmask 255.255.255.0 ssid freebsdap
# ifconfig wlan0
  wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
      ether 00:11:95:d5:43:62
      inet 192.168.0.2 netmask 0xffffff00 broadcast 192.168.0.255
      media: IEEE 802.11 Wireless Ethernet OFDM/54Mbps mode 11g
      status: associated
      ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
      country US ecm authmode OPEN privacy OFF txpower 21.5 bmiss 7
      scanvalid 60 bgscan bgscanintvl 300 bgscanidle 250 roam:rssi 7
```

```
        roam:rate 5 protmode CTS wme burst
```

Точка доступа WPA2 на основе хоста

Этот раздел посвящён настройке точки доступа на хосте FreeBSD с использованием протокола безопасности WPA2. Подробнее о WPA и настройке беспроводных клиентов на основе WPA можно узнать в WPA с EAP-TLS.

Демон hostapd(8) используется для обработки аутентификации клиентов и управления ключами на точке доступа с поддержкой WPA2.

В следующих операциях конфигурации выполняются на машине FreeBSD, выступающей в качестве точки доступа (AP). После того как точка доступа работает корректно, hostapd(8) можно автоматически запускать при загрузке, добавив эту строку в /etc/rc.conf:

```
hostapd_enable="YES"
```

Прежде чем пытаться настроить hostapd(8), сначала настройте основные параметры, описанные в Основные настройки.

WPA2-PSK

WPA2-PSK предназначен для небольших сетей, где использование сервера аутентификации невозможно или нежелательно.

Конфигурация выполняется в /etc/hostapd.conf:

```
interface=wlan0                 ①
debug=1                         ②
ctrl_interface=/var/run/hostapd ③
ctrl_interface_group=wheel      ④
ssid=freebsdap                  ⑤
wpa=2                           ⑥
wpa_passphrase=freebsdmall      ⑦
wpa_key_mgmt=WPA-PSK            ⑧
wpa_pairwise=CCMP               ⑨
```

① Беспроводной интерфейс, используемый для точки доступа.

② Уровень детализации, используемый во время выполнения hostapd(8). Значение 1 представляет минимальный уровень.

③ Путь к каталогу, используемому hostapd(8) для хранения файлов доменных сокетов для взаимодействия с внешними программами, такими как hostapd_cli(8). В этом примере используется значение по умолчанию.

④ Группа, которой разрешён доступ к файлам управляющего интерфейса.

⑤ Имя беспроводной сети, или SSID, которое будет отображаться при сканировании беспроводных сетей.

⑥ Включает WPA и указывает, какой протокол аутентификации WPA будет использоваться. Значение 2 настраивает точку доступа на WPA2 и является рекомендуемым. Установите значение 1 только если требуется устаревший WPA.

⑦ ASCII-пароль для аутентификации WPA.

⑧ Протокол управления ключами для использования. В этом примере установлен WPA-PSK.

⑨ Алгоритмы шифрования, принимаемые точкой доступа. В этом примере принимается только шифр CCMP (AES). CCMP является альтернативой TKIP и настоятельно рекомендуется к использованию, когда это возможно. TKIP следует разрешать только в случае наличия станций, не способных использовать CCMP.

Следующий шаг — запустить hostapd(8):

```
# service hostapd forcestart
```

```
# ifconfig wlan0
wlan0: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> metric 0 mtu 1500
    ether 04:f0:21:16:8e:10
    inet6 fe80::6f0:21ff:fe16:8e10%wlan0 prefixlen 64 scopeid 0x9
    nd6 options=21<PERFORMNUD,AUTO_LINKLOCAL>
    media: IEEE 802.11 Wireless Ethernet autoselect mode 11na <hostap>
    status: running
    ssid No5ignal channel 36 (5180 MHz 11a ht/40+) bssid 04:f0:21:16:8e:10
    country US ecm authmode WPA2/802.11i privacy MIXED deftxkey 2
    AES-CCM 2:128-bit AES-CCM 3:128-bit txpower 17 mcastrate 6 mgmtrate 6
    scanvalid 60 ampdulimit 64k ampdudensity 8 shortgi wme burst
    dtimperiod 1 -dfs
    groups: wlan
```

После запуска точки доступа клиенты могут подключиться к ней. Подробнее см. в разделе Основные настройки. Список станций, подключённых к точке доступа, можно просмотреть с помощью команды `ifconfig wlan0 list sta`.

35.6. Раздача интернета через USB

Многие мобильные телефоны предоставляют возможность совместного использования своего интернет-подключения через USB (часто называемую "тетеринг, раздача Интернета или режим модема"). Эта функция использует один из протоколов: RNDIS, CDC или проприетарный протокол Apple® iPhone®/iPad®.

- Устройства Android™ обычно используют драйвер urndis(4).
- Устройства Apple® используют драйвер ipheth(4).
- Старые устройства часто используют драйвер cdce(4).

Перед подключением устройства загрузите соответствующий драйвер в ядро:

```
# kldload if_urndis
# kldload if_cdce
# kldload if_ipheth
```

После подключения устройства `ue`*0* будет доступен для использования как обычное сетевое устройство. Убедитесь, что на устройстве включена опция "USB-тетеринг".

Чтобы сделать это изменение постоянным и загружать драйвер как модуль при загрузке, добавьте соответствующую строку из следующих в /boot/loader.conf:

```
if_urndis_load="YES"
if_cdce_load="YES"
if_ipheth_load="YES"
```

35.7. Bluetooth

Bluetooth — это беспроводная технология для создания персональных сетей, работающих в нелицензируемом диапазоне 2.4 ГГц, с радиусом действия до 10 метров. Сети обычно формируются на лету из портативных устройств, таких как мобильные телефоны, карманные компьютеры и ноутбуки. В отличие от технологии Wi-Fi, Bluetooth предоставляет сервисы более высокого уровня, такие как FTP-подобные файловые серверы, передача файлов, передача голоса, эмуляция последовательной линии и многое другое.

Этот раздел описывает использование USB Bluetooth адаптера в системе FreeBSD. Затем рассматриваются различные протоколы и утилиты Bluetooth.

35.7.1. Загрузка поддержки Bluetooth

Стек Bluetooth в FreeBSD реализован с использованием фреймворка netgraph(4). Широкий спектр Bluetooth USB-адаптеров поддерживается драйвером ng_ubt(4). Устройства Bluetooth на базе Broadcom BCM2033 поддерживаются драйверами ubtbcmfw(4) и ng_ubt(4). Карта Bluetooth PC Card 3CRWB60-A от 3Com поддерживается драйвером ng_bt3c(4). Bluetooth-устройства на основе последовательного порта и UART поддерживаются драйвером ng_h4(4) и утилитой hcseriald(8).

Прежде чем подключить устройство, определите, какой из вышеуказанных драйверов оно использует, затем загрузите драйвер. Например, если устройство использует драйвер ng_ubt(4):

```
# kldload ng_ubt
```

Если устройство Bluetooth будет подключено к системе во время её загрузки, можно настроить систему на автоматическую загрузку модуля, добавив драйвер в /boot/loader.conf:

```
ng_ubt_load="YES"
```

После загрузки драйвера подключите USB-адаптер. Если загрузка драйвера прошла успешно, на консоли и в /var/log/messages появится вывод, похожий на следующий:

```
ubt0: vendor 0x0a12 product 0x0001, rev 1.10/5.25, addr 2
ubt0: Interface 0 endpoints: interrupt=0x81, bulk-in=0x82, bulk-out=0x2
ubt0: Interface 1 (alt.config 5) endpoints: isoc-in=0x83, isoc-out=0x3,
      wMaxPacketSize=49, nframes=6, buffer size=294
```

Для запуска и остановки стека Bluetooth используйте его стартовый скрипт. Рекомендуется остановить стек перед отключением устройства. Запуск стека Bluetooth может потребовать запуска hcsecd(8). При запуске стека вывод должен быть похож на следующий:

```
# service bluetooth start ubt0
BD_ADDR: 00:02:72:00:d4:1a
Features: 0xff 0xff 0xf 00 00 00 00 00
<3-Slot> <5-Slot> <Encryption> <Slot offset>
<Timing accuracy> <Switch> <Hold mode> <Sniff mode>
<Park mode> <RSSI> <Channel quality> <SCO link>
<HV2 packets> <HV3 packets> <u-law log> <A-law log> <CVSD>
<Paging scheme> <Power control> <Transparent SCO data>
Max. ACL packet size: 192 bytes
Number of ACL packets: 8
Max. SCO packet size: 64 bytes
Number of SCO packets: 8
```

35.7.2. Поиск других устройств Bluetooth

Интерфейс Host Controller Interface (HCI) предоставляет единый метод доступа к базовым возможностям Bluetooth. В FreeBSD узел netgraph HCI создаётся для каждого устройства Bluetooth. Подробнее см. ng_hci(4).

Одной из наиболее распространённых задач является обнаружение Bluetooth-устройств в радиусе действия. Эта операция называется *сканирование* (inquiry). Запрос и другие операции, связанные с HCI, выполняются с помощью hccontrol(8). В приведённом ниже примере показано, как выяснить, какие Bluetooth-устройства находятся в зоне действия. Список устройств должен отобразиться через несколько секунд. Обратите внимание, что удалённое устройство ответит на запрос только в том случае, если оно находится в режиме *обнаруживаемое* (discoverable).

```
% hccontrol -n ubt0hci inquiry
Inquiry result, num_responses=1
Inquiry result #0
       BD_ADDR: 00:80:37:29:19:a4
       Page Scan Rep. Mode: 0x1
```

```
        Page Scan Period Mode: 00
        Page Scan Mode: 00
        Class: 52:02:04
        Clock offset: 0x78ef
Inquiry complete. Status: No error [00]
```

BD_ADDR — это уникальный адрес Bluetooth-устройства, аналогичный MAC-адресу сетевой карты. Этот адрес необходим для дальнейшего взаимодействия с устройством, и ему можно присвоить удобочитаемое имя. Информация об известных Bluetooth-хостах содержится в файле /etc/bluetooth/hosts. В следующем примере показано, как получить удобочитаемое имя, присвоенное удалённому устройству:

```
% hccontrol -n ubt0hci remote_name_request 00:80:37:29:19:a4
BD_ADDR: 00:80:37:29:19:a4
Name: Pav's T39
```

Если выполняется запрос к удалённому устройству Bluetooth, компьютер будет обнаружен как "your.host.name (ubt0)". Имя, назначенное локальному устройству, можно изменить в любое время.

Удалённым устройствам могут быть назначены псевдонимы в /etc/bluetooth/hosts. Дополнительная информация о файле /etc/bluetooth/hosts может быть найдена в bluetooth.hosts(5).

Система Bluetooth обеспечивает соединение точка-точка между двумя устройствами Bluetooth или соединение точка-многоточка, разделяемое между несколькими устройствами Bluetooth. В следующем примере показано, как создать соединение с удалённым устройством:

```
% hccontrol -n ubt0hci create_connection BT_ADDR
```

create_connection принимает BT_ADDR, а также псевдонимы хостов в файле /etc/bluetooth/hosts.

Следующий пример показывает, как получить список активных базовых соединений для локального устройства:

```
% hccontrol -n ubt0hci read_connection_list
Remote BD_ADDR    Handle Type Mode Role Encrypt Pending Queue State
00:80:37:29:19:a4     41  ACL    0 MAST    NONE       0     0 OPEN
```

Дескриптор соединения (connection handle) полезен, когда требуется разрыв базового соединения, хотя обычно это не нужно делать вручную. Стек автоматически разрывает неактивные базовые соединения.

```
# hccontrol -n ubt0hci disconnect 41
Connection handle: 41
```

```
Reason: Connection terminated by local host [0x16]
```

Введите `hccontrol help` для получения полного списка доступных команд HCI. Большинство команд HCI не требуют прав суперпользователя.

35.7.3. Сопряжение устройств

По умолчанию Bluetooth-связь не требует аутентификации, и любое устройство может взаимодействовать с любым другим устройством. Устройство Bluetooth, такое как сотовый телефон, может потребовать аутентификацию для предоставления определённой услуги. Аутентификация Bluetooth обычно выполняется с помощью *PIN-кода* — строки ASCII длиной до 16 символов. Пользователь должен ввести один и тот же PIN-код на обоих устройствах. После ввода PIN-кода оба устройства сгенерируют *ключ связи*. Затем ключ связи может быть сохранен либо в самих устройствах, либо в постоянном хранилище. В следующий раз оба устройства будут использовать ранее сгенерированный ключ связи. Эта процедура называется *сопряжением (pairing)*. Обратите внимание, что если ключ связи будет утерян одним из устройств, спаривание необходимо повторить.

Демон hcsecd(8) отвечает за обработку запросов аутентификации Bluetooth. Конфигурационный файл по умолчанию — /etc/bluetooth/hcsecd.conf. Пример раздела для мобильного телефона с PIN-кодом `1234` приведён ниже:

```
device {
        bdaddr  00:80:37:29:19:a4;
        name    "Pav's T39";
        key     nokey;
        pin     "1234";
      }
```

Единственное ограничение PIN-кодов — их длина. Некоторые устройства, например Bluetooth-гарнитуры, могут иметь встроенный фиксированный PIN-код. Ключ `-d` заставляет hcsecd(8) оставаться на переднем плане, что упрощает отслеживание происходящего. Настройте удалённое устройство на приём сопряжения и инициируйте Bluetooth-соединение с ним. Удалённое устройство должно подтвердить принятие сопряжения и запросить PIN-код. Введите тот же PIN-код, который указан в hcsecd.conf. Теперь компьютер и удалённое устройство сопряжены. Также сопряжение можно инициировать с удалённого устройства.

Следующую строку можно добавить в /etc/rc.conf, чтобы настроить автоматический запуск hcsecd(8) при загрузке системы:

```
hcsecd_enable="YES"
```

Вот пример вывода демона hcsecd(8):

```
hcsecd[16484]: Got Link_Key_Request event from 'ubt0hci', remote bdaddr
```

```
0:80:37:29:19:a4
hcsecd[16484]: Found matching entry, remote bdaddr 0:80:37:29:19:a4, name 'Pav's T39',
link key doesn't exist
hcsecd[16484]: Sending Link_Key_Negative_Reply to 'ubt0hci' for remote bdaddr
0:80:37:29:19:a4
hcsecd[16484]: Got PIN_Code_Request event from 'ubt0hci', remote bdaddr
0:80:37:29:19:a4
hcsecd[16484]: Found matching entry, remote bdaddr 0:80:37:29:19:a4, name 'Pav's T39',
PIN code exists
hcsecd[16484]: Sending PIN_Code_Reply to 'ubt0hci' for remote bdaddr 0:80:37:29:19:a4
```

35.7.4. Доступ в сеть с профилями PPP

Профиль Dial-Up Networking (DUN) может использоваться для настройки сотового телефона в качестве беспроводного модема для подключения к серверу доступа в Интернет через коммутируемое соединение. Он также может применяться для настройки компьютера для приёма входящих вызовов передачи данных с сотового телефона.

Доступ к сети с профилем PPP может использоваться для предоставления доступа к LAN для одного устройства Bluetooth или нескольких устройств Bluetooth. Также он может обеспечить соединение между компьютерами с использованием PPP-сетей через эмуляцию последовательного кабеля.

В FreeBSD эти профили реализованы с помощью ppp(8) и обёртки rfcomm_pppd(8), которая преобразует Bluetooth-соединение в форму, пригодную для использования PPP. Перед использованием профиля необходимо создать новую метку PPP в /etc/ppp/ppp.conf. Примеры можно найти в rfcomm_pppd(8).

В этом примере rfcomm_pppd(8) используется для открытия соединения с удалённым устройством с `BD_ADDR 00:80:37:29:19:a4` на DUNRFCOMM канале:

```
# rfcomm_pppd -a 00:80:37:29:19:a4 -c -C dun -l rfcomm-dialup
```

Фактический номер канала будет получен с удалённого устройства с использованием протокола SDP. Можно указать канал RFCOMM вручную, и в этом случае rfcomm_pppd(8) не будет выполнять запрос SDP. Используйте sdpcontrol(8), чтобы узнать канал RFCOMM на удалённом устройстве.

Для предоставления сетевого доступа через службу PPPLAN необходимо, чтобы работал sdpd(8), и была создана новая запись для клиентов LAN в файле /etc/ppp/ppp.conf. Примеры можно найти в rfcomm_pppd(8). Наконец, запустите сервер RFCOMMPPP на допустимом номере канала RFCOMM. Сервер RFCOMMPPP автоматически зарегистрирует службу Bluetooth LAN в локальном демоне SDP. В приведённом ниже примере показано, как запустить сервер RFCOMMPPP.

```
# rfcomm_pppd -s -C 7 -l rfcomm-server
```

35.7.5. Протоколы Bluetooth

Этот раздел предоставляет обзор различных протоколов Bluetooth, их функций и связанных с ними утилит.

Logical Link Control and Adaptation Protocol (L2CAP)

Протокол управления логическим соединением и адаптации (L2CAP) предоставляет сервисы передачи данных с установлением соединения и без него для протоколов верхнего уровня. L2CAP позволяет протоколам более высокого уровня и приложениям передавать и принимать пакеты данных L2CAP размером до 64 килобайт.

L2CAP основан на концепции *каналов*. Канал — это логическое соединение поверх базового соединения, где каждый канал связан с одним протоколом по принципу "многие к одному". Несколько каналов могут быть связаны с одним и тем же протоколом, но канал не может быть связан с несколькими протоколами. Каждый полученный L2CAP-пакет на канале направляется соответствующему протоколу более высокого уровня. Несколько каналов могут совместно использовать одно и то же базовое соединение.

В FreeBSD для каждого устройства Bluetooth создаётся узел netgraph типа L2CAP. Этот узел обычно соединён с нижестоящим узлом Bluetooth HCI и вышестоящими узлами Bluetooth-сокет. По умолчанию узел L2CAP имеет имя "devicel2cap". Для получения дополнительной информации обратитесь к ng_l2cap(4).

Полезной командой является l2ping(8), которую можно использовать для проверки связи с другими устройствами. Некоторые реализации Bluetooth могут не возвращать все отправленные им данные, поэтому `0 байт` в следующем примере является нормой.

```
# l2ping -a 00:80:37:29:19:a4
0 bytes from 0:80:37:29:19:a4 seq_no=0 time=48.633 ms result=0
0 bytes from 0:80:37:29:19:a4 seq_no=1 time=37.551 ms result=0
0 bytes from 0:80:37:29:19:a4 seq_no=2 time=28.324 ms result=0
0 bytes from 0:80:37:29:19:a4 seq_no=3 time=46.150 ms result=0
```

Утилита l2control(8) используется для выполнения различных операций с узлами L2CAP. Этот пример показывает, как получить список логических соединений (каналов) и список базовых соединений для локального устройства:

```
% l2control -a 00:02:72:00:d4:1a read_channel_list
L2CAP channels:
Remote BD_ADDR     SCID/ DCID   PSM  IMTU/ OMTU State
00:07:e0:00:0b:ca    66/   64     3   132/  672 OPEN
% l2control -a 00:02:72:00:d4:1a read_connection_list
L2CAP connections:
Remote BD_ADDR    Handle Flags Pending State
00:07:e0:00:0b:ca     41 O           0 OPEN
```

Еще один инструмент диагностики — btsockstat(1). Он похож на netstat(1), но предназначен

для структур данных, связанных с Bluetooth-сетями. В примере ниже показано то же логическое соединение, что и в l2control(8) выше.

```
% btsockstat
Active L2CAP sockets
PCB      Recv-Q Send-Q Local address/PSM       Foreign address   CID   State
c2afe900      0      0 00:02:72:00:d4:1a/3     00:07:e0:00:0b:ca 66    OPEN
Active RFCOMM sessions
L2PCB    PCB      Flag MTU   Out-Q DLCs State
c2afe900 c2b53380 1    127   0     Yes  OPEN
Active RFCOMM sockets
PCB      Recv-Q Send-Q Local address     Foreign address   Chan DLCI State
c2e8bc80      0    250 00:02:72:00:d4:1a 00:07:e0:00:0b:ca 3    6    OPEN
```

Радиочастотная связь (RFCOMM)

Протокол RFCOMM обеспечивает эмуляцию последовательных портов поверх протокола L2CAP. RFCOMM — это простой транспортный протокол с дополнительными возможностями для эмуляции 9 последовательных портов RS-232 (EIATIA-232-E). Он поддерживает до 60 одновременных соединений (каналов RFCOMM) между двумя устройствами Bluetooth.

Для целей RFCOMM полный путь передачи данных включает два приложения, работающие на конечных точках соединения, и сегмент передачи данных между ними. RFCOMM предназначен для приложений, использующих последовательные порты устройств, в которых они работают. Сегмент передачи данных представляет собой прямое Bluetooth-соединение между устройствами.

RFCOMM занимается только соединением между устройствами в случае прямого подключения или между устройством и модемом в случае сетевого подключения. RFCOMM может поддерживать другие конфигурации, такие как модули, которые обмениваются данными через технологию беспроводной связи Bluetooth с одной стороны и предоставляют проводной интерфейс с другой стороны.

В FreeBSD RFCOMM реализован на уровне сокетов Bluetooth.

Протокол обнаружения служб (SDP)

Протокол обнаружения служб (SDP) предоставляет клиентским приложениям возможность обнаруживать существование служб, предоставляемых серверными приложениями, а также атрибуты этих служб. Атрибуты службы включают тип или класс предоставляемой службы, а также информацию о механизме или протоколе, необходимую для использования службы.

SDP включает взаимодействие между сервером SDP и клиентом SDP. Сервер хранит список записей служб, которые описывают характеристики служб, связанных с сервером. Каждая запись службы содержит информацию об отдельной службе. Клиент может получить информацию из записи службы, хранящейся на сервере SDP, отправив запрос SDP. Если клиент или приложение, связанное с клиентом, решает использовать службу, он должен

установить отдельное соединение с провайдером службы для её использования. SDP предоставляет механизм для обнаружения служб и их атрибутов, но не предоставляет механизма для использования этих служб.

Обычно клиент SDP ищет услуги на основе определённых желаемых характеристик. Однако бывают случаи, когда необходимо обнаружить, какие типы услуг описаны в записях сервера SDP без какой-либо предварительной информации об этих услугах. Этот процесс поиска любых предлагаемых услуг называется *обзором (browsing)*.

Сервер Bluetooth SDP, sdpd(8), и клиент командной строки — sdpcontrol(8), включены в стандартную установку FreeBSD. В следующем примере показано, как выполнить запрос обзора SDP.

```
% sdpcontrol -a 00:01:03:fc:6e:ec browse
Record Handle: 00000000
Service Class ID List:
        Service Discovery Server (0x1000)
Protocol Descriptor List:
        L2CAP (0x0100)
                Protocol specific parameter #1: u/int/uuid16 1
                Protocol specific parameter #2: u/int/uuid16 1

Record Handle: 0x00000001
Service Class ID List:
        Browse Group Descriptor (0x1001)

Record Handle: 0x00000002
Service Class ID List:
        LAN Access Using PPP (0x1102)
Protocol Descriptor List:
        L2CAP (0x0100)
        RFCOMM (0x0003)
                Protocol specific parameter #1: u/int8/bool 1
Bluetooth Profile Descriptor List:
        LAN Access Using PPP (0x1102) ver. 1.0
```

Обратите внимание, что каждая служба имеет список атрибутов, таких как канал RFCOMM. В зависимости от службы пользователю может потребоваться запомнить некоторые из атрибутов. Некоторые реализации Bluetooth не поддерживают обзор служб и могут возвращать пустой список. В этом случае можно выполнить поиск конкретной службы. В примере ниже показано, как выполнить поиск службы OBEX Object Push (OPUSH):

```
% sdpcontrol -a 00:01:03:fc:6e:ec search OPUSH
```

Предоставление услуг на FreeBSD клиентам Bluetooth осуществляется с помощью сервера sdpd(8). Следующую строку можно добавить в /etc/rc.conf:

```
sdpd_enable="YES"
```

Затем демон sdpd(8) можно запустить с помощью:

```
# service sdpd start
```

Локальное серверное приложение, которое хочет предоставить сервис Bluetooth удалённым клиентам, зарегистрирует сервис в локальном демоне SDP. Примером такого приложения является rfcomm_pppd(8). После запуска оно зарегистрирует сервис Bluetooth LAN в локальном демоне SDP.

Список служб, зарегистрированных на локальном сервере SDP, можно получить, выполнив запрос обзора SDP через локальный управляющий канал:

```
# sdpcontrol -l browse
```

Отправка объектов протоколом OBEX (Object Push — OPUSH)

Обмен объектами (OBEX) — это широко используемый протокол для простой передачи файлов между мобильными устройствами. Основное применение он находит в инфракрасной связи, где используется для передачи файлов общего назначения между ноутбуками или КПК, а также для отправки визитных карточек или записей календаря между сотовыми телефонами и другими устройствами с приложениями для управления персональной информацией (PIM).

Сервер и клиент OBEX реализованы в obexapp, который можно установить с помощью comms/obexapp или порта.

Клиент OBEX используется для отправки и/или получения объектов с сервера OBEX. Пример объекта — визитная карточка или встреча. Клиент OBEX может получить номер канала RFCOMM от удалённого устройства через SDP. Это можно сделать, указав имя службы вместо номера канала RFCOMM. Поддерживаемые имена служб: `IrMC`, `FTRN` и `OPUSH`. Также можно указать канал RFCOMM в виде числа. Ниже приведён пример сеанса OBEX, в котором объект информации об устройстве получается с мобильного телефона, а новый объект, визитная карточка, отправляется в директорию телефона.

```
% obexapp -a 00:80:37:29:19:a4 -C IrMC
obex> get telecom/devinfo.txt devinfo-t39.txt
Success, response: OK, Success (0x20)
obex> put new.vcf
Success, response: OK, Success (0x20)
obex> di
Success, response: OK, Success (0x20)
```

Для предоставления службы OPUSH должен быть запущен sdpd(8), а также должна быть

создана корневая папка, в которой будут храниться все входящие объекты. Путь к корневой папке по умолчанию — /var/spool/obex. Наконец, запустите сервер OBEX на допустимом номере канала RFCOMM. Сервер OBEX автоматически зарегистрирует службу OPUSH в локальном демоне SDP. В приведённом ниже примере показано, как запустить сервер OBEX.

```
# obexapp -s -C 10
```

Профиль последовательного порта (Serial Port Profile — SPP)

Профиль последовательного порта (SPP) позволяет устройствам Bluetooth эмулировать последовательное соединение. Этот профиль позволяет устаревшим приложениям использовать Bluetooth в качестве замены кабеля через абстракцию виртуального последовательного порта.

В FreeBSD rfcomm_sppd(1) реализует SPP, а псевдо-tty используется как абстракция виртуального последовательного порта. В примере ниже показано, как подключиться к сервису последовательного порта удалённого устройства. RFCOMM-канал не обязательно указывать, так как rfcomm_sppd(1) может получить его с удалённого устройства через SDP. Чтобы переопределить это, укажите RFCOMM-канал в командной строке.

```
# rfcomm_sppd -a 00:07:E0:00:0B:CA -t
rfcomm_sppd[94692]: Starting on /dev/pts/6...
/dev/pts/6
```

После подключения псевдотерминал можно использовать как последовательный порт:

```
# cu -l /dev/pts/6
```

Псевдотерминал выводится в stdout и может быть прочитан обёрточными скриптами:

```
PTS=`rfcomm_sppd -a 00:07:E0:00:0B:CA -t`
cu -l $PTS
```

35.7.6. Устранение неполадок

По умолчанию, когда хост FreeBSD принимает новое соединение, он пытается выполнить смену роли и стать ведущим. Некоторые старые устройства Bluetooth, которые не поддерживают смену роли, не смогут подключиться. Поскольку смена роли выполняется при установке нового соединения, невозможно запросить у удалённого устройства, поддерживает ли оно смену роли. Однако существует опция HCI для отключения смены роли на локальной стороне:

```
# hccontrol -n ubt0hci write_node_role_switch 0
```

Для отображения пакетов Bluetooth используйте сторонний пакет hcidump, который можно установить с помощью comms/hcidump или порта. Эта утилита аналогична tcpdump(1) и может использоваться для вывода содержимого пакетов Bluetooth в терминал и для сохранения этих пакетов в файл.

35.8. Создание моста

Иногда полезно разделить сеть, например, сегмент Ethernet, на части без необходимости создания IP-подсетей и использования маршрутизатора для соединения сегментов. Устройство, которое так соединяет две сети, называется "мостом".

Мост работает, изучая MAC-адреса устройств на каждом из своих сетевых интерфейсов. Он передаёт трафик между сетями только в том случае, если исходный и целевой MAC-адреса находятся в разных сетях. Во многих отношениях мост похож на Ethernet-коммутатор с очень малым количеством портов. Система FreeBSD с несколькими сетевыми интерфейсами может быть настроена как мост.

Мост может быть полезен в следующих ситуациях:

Соединение сетей

Основная функция моста — объединить два или более сетевых сегмента. Существует множество причин использовать мост на основе хоста вместо сетевого оборудования, таких как ограничения по кабелям или межсетевой экран. Мост также может соединить беспроводной интерфейс, работающий в режиме hostap, с проводной сетью и выступать в качестве точки доступа.

Межсетевой экран с фильтрацией и управлением трафиком

Мост может использоваться, когда требуется функциональность межсетевого экрана без маршрутизации или преобразования сетевых адресов (NAT).

Пример — небольшая компания, подключённая через DSL или ISDN к провайдеру. У неё есть тринадцать публичных IP-адресов от провайдера и десять компьютеров в сети. В этой ситуации использование маршрутизатора с межсетевым экраном затруднено из-за проблем с подсетями. Межсетевой экран на основе моста можно настроить без каких-либо проблем с IP-адресацией.

Ответвитель сетевого трафика (Network Tap)

Мост может объединить два сетевых сегмента для проверки всех Ethernet-кадров, проходящих между ними, с использованием bpf(4) и tcpdump(1) на интерфейсе моста, или путем отправки копии всех кадров на дополнительный интерфейс, известный как span-порт.

VPN уровня 2

Две Ethernet-сети могут быть соединены через IP-канал путем моста между сетями через туннель EtherIP или решение на основе tap(4), такое как OpenVPN.

Избыточность уровня 2

Сеть может быть соединена несколькими каналами и использовать протокол Spanning

Tree Protocol (STP) для блокировки избыточных путей.

В этом разделе описывается, как настроить систему FreeBSD в качестве моста с использованием if_bridge(4). Также доступен драйвер моста на основе netgraph, который описан в ng_bridge(4).

Фильтрация пакетов может использоваться с любым пакетом межсетевого экрана, который интегрируется в фреймворк pfil(9). Мост может использоваться как шейпер (ограничитель) трафика с altq(4) или dummynet(4).

35.8.1. Включение моста

В FreeBSD if_bridge(4) — это модуль ядра, который автоматически загружается с помощью ifconfig(8) при создании мостового интерфейса. Также можно включить поддержку моста в собственное ядро, добавив `device if_bridge` в конфигурационный файл собственного ядра.

Мост создаётся с помощью клонирования интерфейса. Чтобы создать интерфейс моста:

```
# ifconfig bridge create
bridge0
# ifconfig bridge0
bridge0: flags=8802<BROADCAST,SIMPLEX,MULTICAST> metric 0 mtu 1500
        ether 96:3d:4b:f1:79:7a
        id 00:00:00:00:00:00 priority 32768 hellotime 2 fwddelay 15
        maxage 20 holdcnt 6 proto rstp maxaddr 100 timeout 1200
        root id 00:00:00:00:00:00 priority 0 ifcost 0 port 0
```

При создании интерфейса моста ему автоматически назначается случайно сгенерированный Ethernet-адрес. Параметры `maxaddr` и `timeout` определяют, сколько MAC-адресов будет хранить межсетевой экран в своей таблице переадресации и через сколько секунд каждая запись будет удалена после последнего обнаружения. Остальные параметры управляют работой STP.

Далее укажите, какие сетевые интерфейсы добавить в качестве членов моста. Чтобы мост мог передавать пакеты, все интерфейсы-участники и сам мост должны быть включены:

```
# ifconfig bridge0 addm fxp0 addm fxp1 up
# ifconfig fxp0 up
# ifconfig fxp1 up
```

Мост теперь может передавать Ethernet-кадры между fxp0 и fxp1. Добавьте следующие строки в /etc/rc.conf, чтобы мост создавался при загрузке:

```
cloned_interfaces="bridge0"
ifconfig_bridge0="addm fxp0 addm fxp1 up"
ifconfig_fxp0="up"
```

```
ifconfig_fxp1="up"
```

Если мостовому хосту нужен IP-адрес, установите его на интерфейсе моста, а не на интерфейсах-участниках. Адрес можно задать статически или через DHCP. В этом примере задаётся статический IP-адрес:

```
# ifconfig bridge0 inet 192.168.0.1/24
```

Также можно назначит IPv6 адрес интерфейсу моста.

Чтобы изменения стали постоянными, добавьте информацию об адресации в /etc/rc.conf.

Этот пример использует динамический адрес IPv6:

```
ifconfig_bridge0_ipv6="inet6 auto_linklocal accept_rtadv"
```

Когда включена фильтрация пакетов, транзитные пакеты будут проходить через фильтр на входящем интерфейсе мостового интерфейса и исходящем на соответствующих интерфейсах. Любой из этапов может быть отключен. Если направление потока пакетов важно, лучше настроить межсетевой экран на интерфейсах-участниках, а не на самом мосте.

Мост имеет несколько настраиваемых параметров для передачи не-IP и IP пакетов, а также межсетевой экран второго уровня с ipfw(8). Подробнее см. if_bridge(4).

35.8.2. Включение протокола островного дерева (STP)

Для правильной работы Ethernet-сети между двумя устройствами может существовать только один активный путь. Протокол STP обнаруживает петли и переводит избыточные соединения в заблокированное состояние. Если одно из активных соединений выйдет из строя, STP вычисляет новое дерево и активирует один из заблокированных путей, чтобы восстановить подключение ко всем точкам сети.

Протокол Rapid Spanning Tree (RSTP или 802.1w) обеспечивает обратную совместимость с устаревшим STP. RSTP предоставляет более быструю сходимость и обменивается информацией с соседними коммутаторами для быстрого перехода в режим передачи без создания петель. FreeBSD поддерживает RSTP и STP в качестве режимов работы, причём RSTP является режимом по умолчанию.

STP может быть включен на интерфейсах участников с помощью ifconfig(8). Для моста с интерфейсами fxp0 и fxp1 включите STP командой:

```
# ifconfig bridge0 stp fxp0 stp fxp1
bridge0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        ether d6:cf:d5:a0:94:6d
```

```
        id 00:01:02:4b:d4:50 priority 32768 hellotime 2 fwddelay 15
        maxage 20 holdcnt 6 proto rstp maxaddr 100 timeout 1200
        root id 00:01:02:4b:d4:50 priority 32768 ifcost 0 port 0
        member: fxp0 flags=1c7<LEARNING,DISCOVER,STP,AUTOEDGE,PTP,AUTOPTP>
                port 3 priority 128 path cost 200000 proto rstp
                role designated state forwarding
        member: fxp1 flags=1c7<LEARNING,DISCOVER,STP,AUTOEDGE,PTP,AUTOPTP>
                port 4 priority 128 path cost 200000 proto rstp
                role designated state forwarding
```

Этот мост имеет идентификатор остовного дерева `00:01:02:4b:d4:50` и приоритет `32768`. Поскольку `root id` совпадает, это указывает на то, что данный мост является корневым для дерева.

Еще один мост в сети также имеет включённый STP:

```
bridge0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        ether 96:3d:4b:f1:79:7a
        id 00:13:d4:9a:06:7a priority 32768 hellotime 2 fwddelay 15
        maxage 20 holdcnt 6 proto rstp maxaddr 100 timeout 1200
        root id 00:01:02:4b:d4:50 priority 32768 ifcost 400000 port 4
        member: fxp0 flags=1c7<LEARNING,DISCOVER,STP,AUTOEDGE,PTP,AUTOPTP>
                port 4 priority 128 path cost 200000 proto rstp
                role root state forwarding
        member: fxp1 flags=1c7<LEARNING,DISCOVER,STP,AUTOEDGE,PTP,AUTOPTP>
                port 5 priority 128 path cost 200000 proto rstp
                role designated state forwarding
```

Строка `root id 00:01:02:4b:d4:50 priority 32768 ifcost 400000 port 4` показывает, что корневым мостом является `00:01:02:4b:d4:50` с стоимостью пути `400000` от данного моста. Путь к корневому мосту проходит через `port 4`, которым является fxp0.

35.8.3. Параметры интерфейса моста

Несколько параметров `ifconfig` уникальны для мостовых интерфейсов. В этом разделе приведены некоторые общие варианты использования этих параметров. Полный список доступных параметров описан в ifconfig(8).

private

Приватный интерфейс не передаёт трафик на другие порты, также обозначенные как приватные. Трафик блокируется безусловно, поэтому Ethernet-кадры, включая ARP-пакеты, не будут передаваться. Если требуется выборочная блокировка трафика, следует использовать межсетевой экран.

span

Порт зеркалирования (SPAN —Switch Port Analyzer) передаёт копию каждого Ethernet-фрейма, полученного мостом. Количество портов зеркалирования, настроенных на мосту, не ограничено, но если интерфейс назначен как порт зеркалирования, он не

может использоваться в качестве обычного порта моста. Это наиболее полезно для пассивного мониторинга сети, подключённой к мосту, на другом хосте, подключённом к одному из портов зеркалирования моста. Например, чтобы отправить копию всех фреймов через интерфейс с именем fxp4:

```
# ifconfig bridge0 span fxp4
```

sticky

Если интерфейс участника моста помечен как фиксированный (sticky), динамически изученные записи адресов обрабатываются как статические записи в кэше пересылки. Фиксированные записи никогда не удаляются из кэша и не заменяются, даже если адрес обнаружен на другом интерфейсе. Это даёт преимущество статических записей адресов без необходимости предварительного заполнения таблицы пересылки. Клиенты, изученные на определённом сегменте моста, не могут перемещаться на другой сегмент.

Пример использования фиксированных адресов — это объединение моста с VLAN для изоляции сетей клиентов без потерь IP-адресного пространства. Предположим, что `CustomerA` находится в `vlan100`, `CustomerB` — в `vlan101`, а мост имеет адрес `192.168.0.1`:

```
# ifconfig bridge0 addm vlan100 sticky vlan100 addm vlan101 sticky vlan101
# ifconfig bridge0 inet 192.168.0.1/24
```

В этом примере оба клиента видят `192.168.0.1` в качестве своего шлюза по умолчанию. Поскольку кэш моста устойчив, один узел не может подделать MAC-адрес другого клиента, чтобы перехватить его трафик.

Любое взаимодействие между VLAN может быть заблокировано с помощью межсетевого экрана или, как показано в этом примере, частных интерфейсов:

```
# ifconfig bridge0 private vlan100 private vlan101
```

Клиенты полностью изолированы друг от друга, и весь диапазон адресов `/24` может быть выделен без разделения на подсети.

Количество уникальных исходных MAC-адресов за интерфейсом может быть ограничено. Как только лимит будет достигнут, пакеты с неизвестными исходными адресами будут отбрасываться до тех пор, пока существующая запись в кэше хоста не истечёт или не будет удалена.

Следующий пример устанавливает максимальное количество Ethernet-устройств для `CustomerA` на `vlan100` равным 10:

```
# ifconfig bridge0 ifmaxaddr vlan100 10
```

Мостовые интерфейсы также поддерживают режим мониторинга, в котором пакеты

отбрасываются после обработки bpf(4) и не обрабатываются или передаются дальше. Это можно использовать для мультиплексирования входа двух или более интерфейсов в один поток bpf(4). Это полезно для восстановления трафика сетевых кранов, которые передают сигналы RX/TX через два отдельных интерфейса. Например, чтобы читать входные данные с четырёх сетевых интерфейсов как один поток:

```
# ifconfig bridge0 addm fxp0 addm fxp1 addm fxp2 addm fxp3 monitor up
# tcpdump -i bridge0
```

35.8.4. Мониторинг SNMP

Интерфейс моста и параметры STP можно отслеживать с помощью bsnmpd(1), который включён в базовую систему FreeBSD. Экспортируемые MIB моста соответствуют стандартам IETF, поэтому для получения данных можно использовать любой SNMP-клиент или пакет мониторинга.

Чтобы включить мониторинг на мосту, раскомментируйте эту строку в /etc/snmpd.config, удалив символ # в начале:

```
begemotSnmpdModulePath."bridge" = "/usr/lib/snmp_bridge.so"
```

Другие параметры конфигурации, такие как имена сообществ и списки доступа, возможно, потребуется изменить в этом файле. Подробнее см. в bsnmpd(1) и snmp_bridge(3). После сохранения изменений добавьте следующую строку в /etc/rc.conf:

```
bsnmpd_enable="YES"
```

Затем, запустите bsnmpd(1):

```
# service bsnmpd start
```

Следующие примеры используют программное обеспечение Net-SNMP (net-mgmt/net-snmp) для запроса моста с клиентской системы. Также можно использовать порт net-mgmt/bsnmptools. На SNMP-клиенте, где запущен Net-SNMP, добавьте следующие строки в $HOME/.snmp/snmp.conf, чтобы импортировать определения MIB для моста:

```
mibdirs +/usr/share/snmp/mibs
mibs +BRIDGE-MIB:RSTP-MIB:BEGEMOT-MIB:BEGEMOT-BRIDGE-MIB
```

Для мониторинга одного моста с использованием IETF BRIDGE-MIB (RFC4188):

```
% snmpwalk -v 2c -c public bridge1.example.com mib-2.dot1dBridge
BRIDGE-MIB::dot1dBaseBridgeAddress.0 = STRING: 66:fb:9b:6e:5c:44
```

```
BRIDGE-MIB::dot1dBaseNumPorts.0 = INTEGER: 1 ports
BRIDGE-MIB::dot1dStpTimeSinceTopologyChange.0 = Timeticks: (189959) 0:31:39.59 centi-
seconds
BRIDGE-MIB::dot1dStpTopChanges.0 = Counter32: 2
BRIDGE-MIB::dot1dStpDesignatedRoot.0 = Hex-STRING: 80 00 00 01 02 4B D4 50
...
BRIDGE-MIB::dot1dStpPortState.3 = INTEGER: forwarding(5)
BRIDGE-MIB::dot1dStpPortEnable.3 = INTEGER: enabled(1)
BRIDGE-MIB::dot1dStpPortPathCost.3 = INTEGER: 200000
BRIDGE-MIB::dot1dStpPortDesignatedRoot.3 = Hex-STRING: 80 00 00 01 02 4B D4 50
BRIDGE-MIB::dot1dStpPortDesignatedCost.3 = INTEGER: 0
BRIDGE-MIB::dot1dStpPortDesignatedBridge.3 = Hex-STRING: 80 00 00 01 02 4B D4 50
BRIDGE-MIB::dot1dStpPortDesignatedPort.3 = Hex-STRING: 03 80
BRIDGE-MIB::dot1dStpPortForwardTransitions.3 = Counter32: 1
RSTP-MIB::dot1dStpVersion.0 = INTEGER: rstp(2)
```

Значение `dot1dStpTopChanges.0` равно двум, что указывает на двукратное изменение топологии STP-моста. Изменение топологии означает, что одна или несколько связей в сети изменились или вышли из строя, и было выполнено перерасчёт дерева. Значение `dot1dStpTimeSinceTopologyChange.0` покажет, когда это произошло.

Для мониторинга нескольких интерфейсов моста можно использовать приватный BEGEMOT-BRIDGE-MIB:

```
% snmpwalk -v 2c -c public bridge1.example.com
enterprises.fokus.begemot.begemotBridge
BEGEMOT-BRIDGE-MIB::begemotBridgeBaseName."bridge0" = STRING: bridge0
BEGEMOT-BRIDGE-MIB::begemotBridgeBaseName."bridge2" = STRING: bridge2
BEGEMOT-BRIDGE-MIB::begemotBridgeBaseAddress."bridge0" = STRING: e:ce:3b:5a:9e:13
BEGEMOT-BRIDGE-MIB::begemotBridgeBaseAddress."bridge2" = STRING: 12:5e:4d:74:d:fc
BEGEMOT-BRIDGE-MIB::begemotBridgeBaseNumPorts."bridge0" = INTEGER: 1
BEGEMOT-BRIDGE-MIB::begemotBridgeBaseNumPorts."bridge2" = INTEGER: 1
...
BEGEMOT-BRIDGE-MIB::begemotBridgeStpTimeSinceTopologyChange."bridge0" = Timeticks:
(116927) 0:19:29.27 centi-seconds
BEGEMOT-BRIDGE-MIB::begemotBridgeStpTimeSinceTopologyChange."bridge2" = Timeticks:
(82773) 0:13:47.73 centi-seconds
BEGEMOT-BRIDGE-MIB::begemotBridgeStpTopChanges."bridge0" = Counter32: 1
BEGEMOT-BRIDGE-MIB::begemotBridgeStpTopChanges."bridge2" = Counter32: 1
BEGEMOT-BRIDGE-MIB::begemotBridgeStpDesignatedRoot."bridge0" = Hex-STRING: 80 00 00 40
95 30 5E 31
BEGEMOT-BRIDGE-MIB::begemotBridgeStpDesignatedRoot."bridge2" = Hex-STRING: 80 00 00 50
8B B8 C6 A9
```

Для изменения мониторинга интерфейса моста через поддерево `mib-2.dot1dBridge`:

```
% snmpset -v 2c -c private bridge1.example.com
BEGEMOT-BRIDGE-MIB::begemotBridgeDefaultBridgeIf.0 s bridge2
```

35.9. Агрегация каналов и отказоустойчивость

FreeBSD предоставляет интерфейс lagg(4), который может использоваться для объединения нескольких сетевых интерфейсов в один виртуальный интерфейс с целью обеспечения отказоустойчивости и агрегации каналов. Отказоустойчивость позволяет трафику продолжать передаваться, пока хотя бы один из объединённых сетевых интерфейсов имеет установленное соединение. Агрегация каналов наиболее эффективна на коммутаторах, поддерживающих LACP, так как этот протокол распределяет трафик в обоих направлениях, реагируя на отказ отдельных каналов.

Протоколы агрегации, поддерживаемые интерфейсом lagg, определяют, какие порты используются для исходящего трафика и принимает ли конкретный порт входящий трафик. В lagg(4) поддерживаются следующие протоколы:

failover

Этот режим отправляет и получает трафик только через основной порт. Если основной порт становится недоступным, используется следующий активный порт. Первый интерфейс, добавленный в виртуальный интерфейс, является основным портом, а все последующие добавленные интерфейсы используются как устройства резервирования. Если происходит переключение на неосновной порт, исходный порт снова становится основным, как только становится доступным.

loadbalance

Это обеспечивает статическую настройку и не выполняет согласование агрегации с узлом или обмен кадрами для мониторинга связи. Если коммутатор поддерживает LACP, следует использовать его.

lacp

Протокол управления агрегацией каналов IEEE® 802.3ad (LACP) согласует набор агрегируемых каналов с узлом-партнёром в один или несколько агрегированных групп каналов (LAG). Каждая LAG состоит из портов с одинаковой скоростью, настроенных на полнодуплексный режим работы, а трафик распределяется между портами в LAG с наибольшей общей скоростью. Обычно существует только одна LAG, содержащая все порты. В случае изменений физической связности LACP быстро сходится к новой конфигурации.

LACP распределяет исходящий трафик по активным портам на основе хешированной информации заголовков протоколов и принимает входящий трафик с любого активного порта. Хеш включает исходный и целевой Ethernet-адреса, а также, если доступно, тег VLAN и исходный и целевой IPv4 или IPv6 адреса.

roundrobin

Этот режим распределяет исходящий трафик с помощью циклического планировщика через все активные порты и принимает входящий трафик с любого активного порта. Поскольку этот режим нарушает порядок следования Ethernet-кадров, его следует использовать с осторожностью.

broadcast

Этот режим отправляет исходящий трафик на все порты, настроенные на интерфейсе lagg, и принимает кадры на любом порту.

35.9.1. Примеры конфигурации

В этом разделе показано, как настроить коммутатор Cisco® и систему FreeBSD для балансировки нагрузки LACP. Затем демонстрируется настройка двух Ethernet-интерфейсов в режиме отказоустойчивости, а также настройка режима отказоустойчивости между Ethernet и беспроводным интерфейсом.

Пример 44. Агрегация каналов с использованием LACP и коммутатора Cisco®

В этом примере два Ethernet-интерфейса fxp(4) на машине FreeBSD подключены к первым двум Ethernet-портам коммутатора Cisco® в виде единого отказоустойчивого канала с балансировкой нагрузки. Дополнительные интерфейсы могут быть добавлены для увеличения пропускной способности и отказоустойчивости. Замените названия портов Cisco®, Ethernet-устройств, номер группы каналов и IP-адрес, указанные в примере, в соответствии с вашей конфигурацией.

Порядок кадров обязателен на Ethernet-соединениях, и любой трафик между двумя станциями всегда проходит через одно и то же физическое соединение, что ограничивает максимальную скорость пропускной способностью одного интерфейса. Алгоритм передачи пытается использовать как можно больше информации для различения отдельных потоков трафика и балансировки этих потоков между доступными интерфейсами.

На коммутаторе Cisco® добавьте интерфейсы *FastEthernet0/1* и *FastEthernet0/2* в группу каналов *1*:

```
interface FastEthernet0/1
 channel-group 1 mode active
 channel-protocol lacp
!
interface FastEthernet0/2
 channel-group 1 mode active
 channel-protocol lacp
```

На системе FreeBSD создайте интерфейс lagg(4), используя физические интерфейсы *fxp0* и *fxp1*, и поднимите интерфейсы с IP-адресом *10.0.0.3/24*:

```
# ifconfig fxp0 up
# ifconfig fxp1 up
# ifconfig lagg0 create
# ifconfig lagg0 up laggproto lacp laggport fxp0 laggport fxp1 10.0.0.3/24
```

Далее проверьте состояние виртуального интерфейса:

```
# ifconfig lagg0
lagg0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=8<VLAN_MTU>
        ether 00:05:5d:71:8d:b8
        inet 10.0.0.3 netmask 0xffffff00 broadcast 10.0.0.255
        media: Ethernet autoselect
        status: active
        laggproto lacp
        laggport: fxp1 flags=1c<ACTIVE,COLLECTING,DISTRIBUTING>
        laggport: fxp0 flags=1c<ACTIVE,COLLECTING,DISTRIBUTING>
```

Порты, помеченные как `ACTIVE`, являются частью LAG, который был согласован с удалённым коммутатором. Через эти активные порты будет передаваться и приниматься трафик. Добавьте `-v` к приведённой выше команде, чтобы просмотреть идентификаторы LAG.

Чтобы проверить статус порта на коммутаторе Cisco®:

```
switch# show lacp neighbor
Flags:  S - Device is requesting Slow LACPDUs
        F - Device is requesting Fast LACPDUs
        A - Device is in Active mode       P - Device is in Passive mode

Channel group 1 neighbors

Partner's information:

                  LACP port                        Oper    Port     Port
Port      Flags   Priority  Dev ID          Age    Key     Number   State
Fa0/1     SA      32768     0005.5d71.8db8  29s    0x146   0x3      0x3D
Fa0/2     SA      32768     0005.5d71.8db8  29s    0x146   0x4      0x3D
```

Для получения более подробной информации введите `show lacp neighbor detail`.

Чтобы сохранить эту конфигурацию после перезагрузки, добавьте следующие записи в /etc/rc.conf на системе FreeBSD:

```
ifconfig_fxp0="up"
ifconfig_fxp1="up"
cloned_interfaces="lagg0"
ifconfig_lagg0="laggproto lacp laggport fxp0 laggport fxp1 10.0.0.3/24"
```

Пример 45. Режим отказоустойчивости

Режим отказоустойчивости может быть использован для переключения на резервный интерфейс, если соединение на основном интерфейсе потеряно. Для настройки

отказоустойчивости убедитесь, что физические интерфейсы активны, затем создайте интерфейс lagg(4). В этом примере *fxp0* — основной интерфейс, *fxp1* — резервный интерфейс, а виртуальному интерфейсу назначен IP-адрес *10.0.0.15/24*:

```
# ifconfig fxp0 up
# ifconfig fxp1 up
# ifconfig lagg0 create
# ifconfig lagg0 up laggproto failover laggport fxp0 laggport fxp1 10.0.0.15/24
```

Виртуальный интерфейс должен выглядеть примерно так:

```
# ifconfig lagg0
lagg0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=8<VLAN_MTU>
        ether 00:05:5d:71:8d:b8
        inet 10.0.0.15 netmask 0xffffff00 broadcast 10.0.0.255
        media: Ethernet autoselect
        status: active
        laggproto failover
        laggport: fxp1 flags=0<>
        laggport: fxp0 flags=5<MASTER,ACTIVE>
```

Трафик будет передаваться и приниматься через интерфейс *fxp0*. Если соединение на *fxp0* будет потеряно, активным станет интерфейс *fxp1*. Если соединение на основном интерфейсе восстановится, он снова станет активным.

Чтобы сохранить эту конфигурацию после перезагрузки, добавьте следующие записи в /etc/rc.conf:

```
ifconfig_fxp0="up"
ifconfig_fxp1="up"
cloned_interfaces="lagg0"
ifconfig_lagg0="laggproto failover laggport fxp0 laggport fxp1 10.0.0.15/24"
```

Пример 46. Режим автоматического переключения между Ethernet и беспроводным интерфейсом

Для пользователей ноутбуков обычно желательно настроить беспроводное устройство как вторичное, которое используется только при отсутствии Ethernet-подключения. С помощью lagg(4) можно настроить отказоустойчивость, отдавая предпочтение Ethernet-подключению как по соображениям производительности, так и безопасности, сохраняя при этом возможность передачи данных через беспроводное соединение.

Это достигается путем замены MAC-адреса Ethernet-интерфейса на MAC-адрес беспроводного интерфейса.

Теоретически, либо Ethernet, либо беспроводной MAC-адрес можно изменить, чтобы они были одинаковыми. Однако некоторые популярные беспроводные интерфейсы не поддерживают переопределение MAC-адреса. Поэтому мы рекомендуем переопределить Ethernet MAC-адрес для этой цели.

Если драйвер для беспроводного интерфейса не загружен в `GENERIC` или собственном ядре, и компьютер работает под FreeBSD 12.1, загрузите соответствующий .ko в /boot/loader.conf, добавив `driver_load="YES"` в этот файл и перезагрузив систему. Другой, более предпочтительный способ — загрузить драйвер в /etc/rc.conf, добавив его в `kld_list` (подробности см. в rc.conf(5)) в этом файле и перезагрузив систему. Это необходимо, потому что в противном случае драйвер ещё не загружен к моменту настройки интерфейса lagg(4).

В этом примере Ethernet-интерфейс *re0* является основным, а беспроводной интерфейс *wlan0* — резервным. Интерфейс *wlan0* был создан на основе физического беспроводного интерфейса *ath0*, а Ethernet-интерфейс будет настроен с MAC-адресом беспроводного интерфейса. Сначала включите беспроводной интерфейс (замените *FR* на код вашей страны из 2 букв), но не задавайте IP-адрес. Замените *wlan0* на имя беспроводного интерфейса вашей системы:

```
# ifconfig wlan0 create wlandev ath0 country FR ssid my_router up
```

Определите MAC-адрес беспроводного интерфейса, например, так:

```
# ifconfig wlan0
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
    ether b8:ee:65:5b:32:59
    groups: wlan
    ssid Bbox-A3BD2403 channel 6 (2437 MHz 11g ht/20) bssid 00:37:b7:56:4b:60
    regdomain ETSI country FR indoor ecm authmode WPA2/802.11i privacy ON
    deftxkey UNDEF AES-CCM 2:128-bit txpower 30 bmiss 7 scanvalid 60
    protmode CTS ampdulimit 64k ampdudensity 8 shortgi -stbctx stbcrx
    -ldpc wme burst roaming MANUAL
    media: IEEE 802.11 Wireless Ethernet MCS mode 11ng
    status: associated
    nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
```

Строка `ether` будет содержать MAC-адрес указанного интерфейса. Теперь измените MAC-адрес интерфейса Ethernet, чтобы он совпадал:

```
# ifconfig re0 ether b8:ee:65:5b:32:59
```

Убедитесь, что интерфейс *re0* включен, затем создайте интерфейс lagg(4) с *re0* в качестве основного с переключением на *wlan0* в случае отказа:

```
# ifconfig re0 up
# ifconfig lagg0 create
# ifconfig lagg0 up laggproto failover laggport re0 laggport wlan0
```

Виртуальный интерфейс должен выглядеть примерно так:

```
# ifconfig lagg0
lagg0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=8<VLAN_MTU>
        ether b8:ee:65:5b:32:59
        laggproto failover lagghash l2,l3,l4
        laggport: re0 flags=5<MASTER,ACTIVE>
        laggport: wlan0 flags=0<>
        groups: lagg
        media: Ethernet autoselect
        status: active
```

Затем запустите DHCP-клиент для получения IP-адреса:

```
# dhclient lagg0
```

Чтобы сохранить эту конфигурацию после перезагрузки, добавьте следующие записи в /etc/rc.conf:

```
ifconfig_re0="ether b8:ee:65:5b:32:59"
wlans_ath0="wlan0"
ifconfig_wlan0="WPA"
create_args_wlan0="country FR"
cloned_interfaces="lagg0"
ifconfig_lagg0="up laggproto failover laggport re0 laggport wlan0 DHCP"
```

35.10. Запуск системы по сети (PXE) без использования локальных накопителей

Встроенная среда исполнения Intel® Preboot eXecution Environment (PXE) позволяет операционной системе загружаться по сети. Например, система FreeBSD может загружаться по сети и работать без локального диска, используя файловые системы, смонтированные с NFS-сервера. Поддержка PXE обычно доступна в BIOS. Чтобы использовать PXE при запуске машины, выберите опцию `Загрузка по сети` в настройках BIOS или нажмите функциональную клавишу во время инициализации системы.

Для предоставления файлов, необходимых для загрузки операционной системы по сети, настройка PXE также требует правильно настроенных серверов DHCP, TFTP и NFS, где:

- Исходные параметры, такие как IP-адрес, имя загружаемого исполняемого файла и его расположение, имя сервера и корневой путь, получаются от сервера DHCP.
- Файл загрузчика операционной системы загружается с использованием TFTP.
- Файловые системы загружаются с использованием NFS.

Когда компьютер загружается по PXE, он получает информацию через DHCP о том, где получить начальный загрузочный файл. После того как хост-компьютер получает эту информацию, он загружает загрузчик через TFTP и затем выполняет его. В FreeBSD загрузочный файл называется /boot/pxeboot. После выполнения /boot/pxeboot загружается ядро FreeBSD, и продолжается остальная последовательность загрузки FreeBSD, как описано в Процесс загрузки FreeBSD.

Для загрузки по PXE на UEFI фактический файл загрузчика, который следует использовать, это /boot/loader.efi. См. раздел ниже Отладка проблем PXE о том, как использовать /boot/loader.efi.

Этот раздел описывает, как настроить эти службы в системе FreeBSD, чтобы другие системы могли загружаться по PXE в FreeBSD. Дополнительную информацию можно найти в diskless(8).

Как уже упоминалось, система, предоставляющая эти службы, является небезопасной. Она должна находиться в защищённой части сети и не доверяться другим хостам.

35.10.1. Настройка окружения PXE

В этом разделе показаны шаги по настройке встроенных серверов NFS и TFTP. В следующем разделе будет продемонстрировано, как установить и настроить сервер DHCP. В данном примере каталог, который будет содержать файлы, используемые пользователями PXE, называется /b/tftpboot/FreeBSD/install. Важно, чтобы этот каталог существовал и чтобы одно и то же имя каталога было указано как в /etc/inetd.conf, так и в /usr/local/etc/dhcpd.conf.

Примеры команд ниже предполагают использование оболочки sh(1). Пользователям csh(1) и tcsh(1) потребуется запустить оболочку sh(1) или адаптировать команды к синтаксису csh(1).

1. Создайте корневой каталог, который будет содержать установку FreeBSD для монтирования по NFS:

```
# export NFSROOTDIR=/b/tftpboot/FreeBSD/install
# mkdir -p ${NFSROOTDIR}
```

2. Включите сервер NFS, добавив следующую строку в /etc/rc.conf:

```
nfs_server_enable="YES"
```

3. Экспортируйте корневой каталог бездисковой системы через NFS, добавив следующее в /etc/exports:

```
/b -ro -alldirs -maproot=root
```

4. Запустите сервер NFS:

```
# service nfsd start
```

5. Включите inetd(8), добавив следующую строку в /etc/rc.conf:

```
inetd_enable="YES"
```

6. Раскомментируйте следующую строку в /etc/inetd.conf, убедившись, что она не начинается с символа #:

```
tftp dgram udp wait root /usr/libexec/tftpd tftpd blocksize 1468 -l -s /b/tftpboot
```

Указанный размер блока tftp, например, 1468 байт, заменяет стандартный размер 512 байт. Некоторые версии PXE требуют TCP-версию TFTP. В этом случае раскомментируйте вторую строку `tftp`, содержащую `stream tcp`.

7. Запустите inetd(8):

```
# service inetd start
```

8. Установите базовую систему в ${NFSROOTDIR}, либо распаковав официальные архивы, либо пересобрав ядро и пользовательское окружение FreeBSD (подробные инструкции можно найти в “Обновление FreeBSD из исходного кода”, но не забудьте добавить `DESTDIR=${NFSROOTDIR}` при выполнении команд `make installkernel` и `make installworld`).

9. Проверьте, что TFTP-сервер работает и может загрузить загрузчик, который будет получен через PXE:

```
# tftp localhost
tftp> get FreeBSD/install/boot/pxeboot
Received 264951 bytes in 0.1 seconds
```

10. Отредактируйте файл ${NFSROOTDIR}/etc/fstab и создайте запись для монтирования корневой файловой системы через NFS:

```
# Device                                        Mountpoint    FSType   Options
```

```
Dump Pass
myhost.example.com:/b/tftpboot/FreeBSD/install       /         nfs      ro       0
0
```

Замените *myhost.example.com* на имя хоста или IP-адрес сервера NFS. В этом примере корневая файловая система монтируется в режиме только для чтения, чтобы предотвратить возможное удаление содержимого корневой файловой системы клиентами NFS.

11. Установите пароль root в среде PXE для клиентских машин, загружающихся через PXE:

```
# chroot ${NFSROOTDIR}
# passwd
```

12. При необходимости разрешите вход root по ssh(1) для клиентских машин, загружающихся через PXE, отредактировав ${NFSROOTDIR}/etc/ssh/sshd_config и включив `PermitRootLogin`. Эта опция описана в sshd_config(5).

13. Выполните другие необходимые настройки PXE-окружения в ${NFSROOTDIR}. Эти настройки могут включать установку пакетов или редактирование файла паролей с помощью vipw(8).

При загрузке с корневого тома NFS, /etc/rc определяет загрузку по NFS и запускает /etc/rc.initdiskless. В этом случае /etc и /var должны быть файловыми системами в памяти, чтобы эти каталоги были доступны для записи, тогда как корневой каталог NFS доступен только для чтения:

```
# chroot ${NFSROOTDIR}
# mkdir -p conf/base
# tar -c -v -f conf/base/etc.cpio.gz --format cpio --gzip etc
# tar -c -v -f conf/base/var.cpio.gz --format cpio --gzip var
```

При загрузке системы будут созданы и смонтированы файловые системы в памяти для /etc и /var, а содержимое файлов cpio.gz будет скопировано в них. По умолчанию эти файловые системы имеют максимальный размер 5 мегабайт. Если архивы не помещаются, что обычно происходит с /var при установке бинарных пакетов, укажите больший размер, записав количество необходимых секторов по 512 байт (например, 5 мегабайт — это 10240 секторов) в файлы ${NFSROOTDIR}/conf/base/etc/md_size и ${NFSROOTDIR}/conf/base/var/md_size для файловых систем /etc и /var соответственно.

35.10.2. Настройка DHCP-сервера

Сервер DHCP не обязательно должен быть тем же самым компьютером, что и серверы TFTP и NFS, но он должен быть доступен в сети.

DHCP не является частью базовой системы FreeBSD, но может быть установлен с помощью порта net/isc-dhcp44-server или пакета.

После установки отредактируйте файл конфигурации /usr/local/etc/dhcpd.conf. Настройте параметры `next-server`, `filename` и `root-path`, как показано в этом примере:

```
subnet 192.168.0.0 netmask 255.255.255.0 {
   range 192.168.0.2 192.168.0.3 ;
   option subnet-mask 255.255.255.0 ;
   option routers 192.168.0.1 ;
   option broadcast-address 192.168.0.255 ;
   option domain-name-servers 192.168.35.35, 192.168.35.36 ;
   option domain-name "example.com";

   # IP address of TFTP server
   next-server 192.168.0.1 ;

   # path of boot loader obtained via tftp
   filename "FreeBSD/install/boot/pxeboot" ;

   # pxeboot boot loader will try to NFS mount this directory for root FS
   option root-path "192.168.0.1:/b/tftpboot/FreeBSD/install/" ;

}
```

Директива `next-server` используется для указания IP-адреса TFTP-сервера.

Директива `filename` определяет путь к /boot/pxeboot. Используется относительное имя файла, что означает, что /b/tftpboot не включен в путь.

Опция `root-path` определяет путь к корневой файловой системе NFS.

После сохранения изменений включите DHCP при загрузке, добавив следующую строку в /etc/rc.conf:

```
dhcpd_enable="YES"
```

Затем запустите службу DHCP:

```
# service isc-dhcpd start
```

35.10.3. Отладка проблем PXE

После настройки и запуска всех служб клиенты PXE должны автоматически загружать FreeBSD по сети. Если конкретный клиент не может подключиться, при загрузке этой машины войдите в меню конфигурации BIOS и убедитесь, что она настроена на загрузку с сети.

Этот раздел содержит несколько советов по устранению неполадок для выявления источника проблемы с конфигурацией, если клиенты не могут загрузиться через PXE.

1. Используйте пакет net/wireshark или порт для отладки сетевого трафика, задействованного в процессе загрузки по PXE, который показан на схеме ниже.

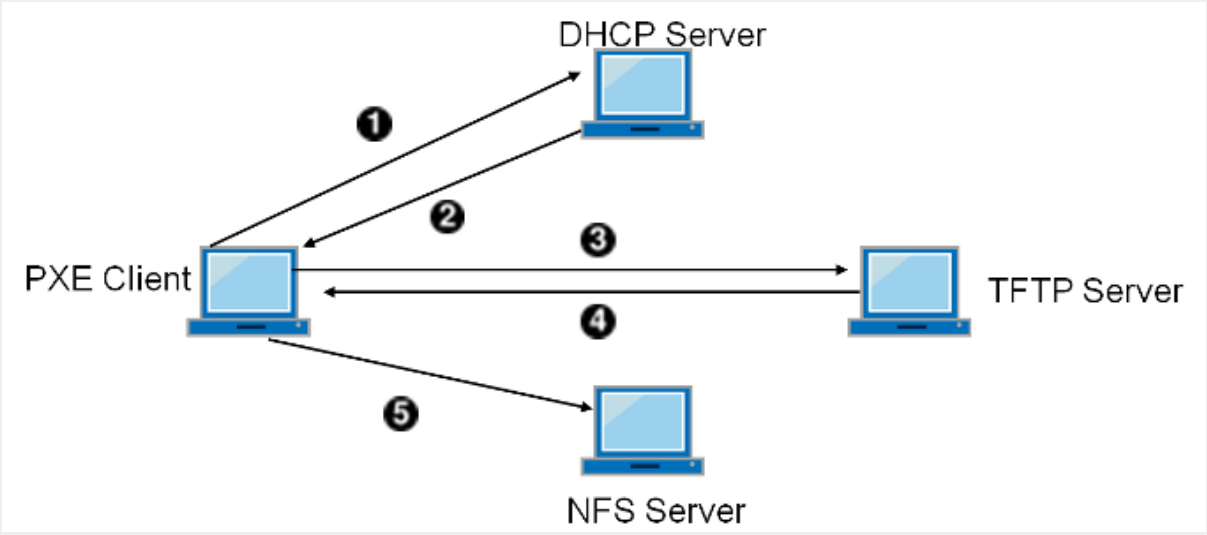


Рисунок 78. Процесс загрузки PXE с монтированием корневой файловой системы по NFS

   1. Клиент рассылает широковещательное сообщение DHCPDISCOVER.
   2. Сервер DHCP отвечает с указанием IP-адреса, next-server, filename и значений root-path.
   3. Клиент отправляет запрос TFTP на next-server, запрашивая получение файла filename.
   4. Сервер TFTP отвечает и отправляет имя файла клиенту.
   5. Клиент выполняет файл `pxeboot(8)`, который затем загружает ядро. При запуске ядра корневая файловая система, указанная в `root-path`, монтируется через NFS.

2. На TFTP-сервере прочитайте /var/log/xferlog, чтобы убедиться, что pxeboot загружается из правильного места. Для проверки конфигурации, сделанной для этого примера:

```
# tftp 192.168.0.1
tftp> get FreeBSD/install/boot/pxeboot
Received 264951 bytes in 0.1 seconds
```

   Разделы `BUGS` в tftpd(8) и tftp(1) описывают некоторые ограничения TFTP.

3. Убедитесь, что корневая файловая система может быть смонтирована через NFS. Для проверки конфигурации из этого примера:

```
# mount -t nfs 192.168.0.1:/b/tftpboot/FreeBSD/install /mnt
```

4. Для загрузки по PXE на UEFI замените файл boot/pxeboot на файл boot/loader.efi:

```
# chroot ${NFSROOTDIR}
```

```
# mv boot/pxeboot boot/pxeboot.original
# cp boot/loader.efi boot/pxeboot
```

35.11. Протокол общей избыточности адресов (CARP)

Протокол общей избыточности адресов (CARP) позволяет нескольким хостам использовать один и тот же IP-адрес и идентификатор виртуального хоста (VHID) для обеспечения *высокой доступности* одной или нескольких служб. Это означает, что при отказе одного или нескольких хостов остальные хосты прозрачно возьмут на себя их функции, чтобы пользователи не заметили сбоя в работе службы.

В дополнение к общему IP-адресу каждый узел имеет собственный IP-адрес для управления и настройки. Все машины, которые используют общий IP-адрес, имеют одинаковый VHID. для каждого виртуального IP-адреса VHID должен быть уникальным в пределах широковещательного домена сетевого интерфейса.

Высокая доступность с использованием CARP встроена в FreeBSD, хотя шаги по её настройке немного различаются в зависимости от версии FreeBSD. В этом разделе приведён одинаковый пример конфигурации для версий до и начиная с FreeBSD 10.

Этот пример настраивает поддержку отказоустойчивости с тремя хостами, каждый из которых имеет уникальный IP-адрес, но предоставляет одинаковое веб-содержимое. В нём есть два разных основных хоста с именами `hosta.example.org` и `hostb.example.org`, а также общий резервный хост с именем `hostc.example.org`.

Эти машины балансируют нагрузку с помощью конфигурации DNS Round Robin. Основная и резервная машины настроены идентично, за исключением их имён хостов и управляющих IP-адресов. Эти серверы должны иметь одинаковую конфигурацию и запускать одинаковые службы. При возникновении переключения, запросы к службе на общем IP-адресе могут быть корректно обработаны только если резервный сервер имеет доступ к тому же содержимому. Резервная машина имеет два дополнительных интерфейса CARP, по одному для каждого из IP-адресов основного сервера содержимого. При возникновении сбоя, резервный сервер возьмёт IP-адрес вышедшей из строя основной машины.

35.11.1. Использование CARP

Включите поддержку CARP при загрузке, добавив запись для модуля ядра carp.ko в /boot/loader.conf:

```
carp_load="YES"
```

Чтобы сейчас загрузить модуль без перезагрузки:

```
# kldload carp
```

Для пользователей, которые предпочитают использовать собственное ядро, добавьте следующую строку в файл конфигурации ядра и скомпилируйте его, как описано в Настройка ядра FreeBSD:

```
device  carp
```

Имя хоста, управляющий IP-адрес и маска подсети, общий IP-адрес и VHID задаются путем добавления записей в /etc/rc.conf. Этот пример для `hosta.example.org`:

```
hostname="hosta.example.org"
ifconfig_em0="inet 192.168.1.3 netmask 255.255.255.0"
ifconfig_em0_alias0="inet vhid 1 pass testpass alias 192.168.1.50/32"
```

Следующий набор записей предназначен для `hostb.example.org`. Поскольку он представляет второй мастер, используется другой общий IP-адрес и VHID. Однако пароли, указанные с помощью `pass`, должны быть идентичными, так как CARP будет принимать и обрабатывать объявления только от машин с правильным паролем.

```
hostname="hostb.example.org"
ifconfig_em0="inet 192.168.1.4 netmask 255.255.255.0"
ifconfig_em0_alias0="inet vhid 2 pass testpass alias 192.168.1.51/32"
```

Третья машина, `hostc.example.org`, настроена для обработки перехода на резервный сервер от любого из основных. Эта машина настроена с двумя CARPVHID, по одному для обработки виртуального IP-адреса каждого из основных хостов. Значение advskew (временной сдвиг анонсов CARP) обеспечивает задержку в отправке анонсов резервным сервером по сравнению с основным, поскольку `advskew` контролирует порядок приоритета при наличии нескольких резервных серверов.

```
hostname="hostc.example.org"
ifconfig_em0="inet 192.168.1.5 netmask 255.255.255.0"
ifconfig_em0_alias0="inet vhid 1 advskew 100 pass testpass alias 192.168.1.50/32"
ifconfig_em0_alias1="inet vhid 2 advskew 100 pass testpass alias 192.168.1.51/32"
```

Наличие двух настроенных CARPVHID означает, что `hostc.example.org` заметит, если один из главных серверов станет недоступен. Если главный сервер не отправит объявление раньше резервного сервера, резервный сервер возьмёт на себя общий IP-адрес до тех пор, пока главный сервер снова не станет доступен.

Если исходный главный сервер снова станет доступен, `hostc.example.org` не освободит виртуальный IP-адрес автоматически. Чтобы это произошло, необходимо включить вытеснение. Эта функция отключена по умолчанию и управляется через переменную sysctl(8) `net.inet.carp.preempt`. Администратор может принудительно вернуть IP-адрес главному серверу с резервного сервера:

```
# ifconfig em0 vhid 1 state backup
```

Как только настройка завершена, перезапустите сеть или перезагрузите каждую систему. Высокая доступность теперь включена.

Функциональность CARP может управляться с помощью нескольких переменных sysctl(8), описанных в carp(4). Другие действия могут быть запущены при событиях CARP с использованием devd(8).

35.12. Виртуальные сети VLAN

Виртуальные локальные сети (VLAN) — это способ виртуального разделения сети на множество различных подсетей, также называемый сегментированием. Каждый сегмент будет иметь свою собственную широковещательную область и быть изолированным от других VLAN.

На FreeBSD поддержка VLAN должна быть обеспечена драйвером сетевой карты. Чтобы узнать, какие драйверы поддерживают VLAN, обратитесь к странице руководства vlan(4).

При настройке VLAN необходимо знать несколько параметров. Во-первых, какой сетевой интерфейс? Во-вторых, какой тег VLAN?

Для настройки VLAN во время выполнения, с сетевой картой `em0` и тегом VLAN `5`, команда будет выглядеть следующим образом:

```
# ifconfig em0.5 create vlan 5 vlandev em0 inet 192.168.20.20/24
```

Видите, что имя интерфейса состоит из имени драйвера сетевой карты и тега VLAN, разделённых точкой? Это рекомендуемая практика для упрощения работы с конфигурациями VLAN, когда на машине присутствует множество VLAN.

При определении VLAN убедитесь, что родительский сетевой интерфейс также настроен и включен. Минимальная конфигурация для приведённого выше примера будет следующей:

```
# ifconfig em0 up
```

Для настройки VLAN при загрузке необходимо обновить файл /etc/rc.conf. Чтобы повторить приведённую выше конфигурацию, нужно добавить следующее:

```
vlans_em0="5"
ifconfig_em0_5="inet 192.168.20.20/24"
```

Дополнительные VLAN могут быть добавлены путём простого дополнения тега в поле `vlans_em0` и добавления дополнительной строки для настройки сети на интерфейсе с этим тегом VLAN.

При определении VLAN в /etc/rc.conf убедитесь, что родительский сетевой интерфейс также настроен и включен. Минимальная конфигурация для приведённого выше примера будет следующей:

```
ifconfig_em0="up"
```

Полезно присвоить интерфейсу символическое имя, чтобы при изменении связанного оборудования требовалось обновить лишь несколько переменных конфигурации. Например, камеры наблюдения должны работать через VLAN 1 на `em0`. Позже, если карта `em0` будет заменена на карту с драйвером ixgb(4), все упоминания `em0.1` не нужно будет изменять на `ixgb0.1`.

Для настройки VLAN `5` на сетевой карте `em0`, назначения интерфейсу имени `cameras` и присвоения интерфейсу IP-адреса `192.168.20.20` с `24`-битным префиксом используйте следующую команду:

```
# ifconfig em0.5 create vlan 5 vlandev em0 name cameras inet 192.168.20.20/24
```

Для интерфейса с именем `video` используйте следующее:

```
# ifconfig video.5 create vlan 5 vlandev video name cameras inet 192.168.20.20/24
```

Чтобы применить изменения при загрузке, добавьте следующие строки в /etc/rc.conf:

```
vlans_video="cameras"
create_args_cameras="vlan 5"
ifconfig_cameras="inet 192.168.20.20/24"
```

Часть V: Приложения

Приложение A: Получение FreeBSD

A.1. Зеркала

Официальные зеркала проекта FreeBSD состоят из множества машин, управляемых администраторами кластера проекта, и используют GeoDNS для направления пользователей к ближайшему доступному зеркалу. Текущие местоположения: Австралия, Бразилия, Япония (две площадки), Малайзия, Южная Африка, Швеция, Тайвань, Соединённые Штаты Америки (Калифорния — две площадки, Иллинойс — две площадки, Нью-Джерси и Вашингтон).

Официальный сервис зеркал:

Имя Сервиса	Протоколы	Больше информации
cgit.FreeBSD.org	https	Веб-интерфейс для Git-репозиториев FreeBSD.
docs.FreeBSD.org	https	Портал документации FreeBSD.
download.FreeBSD.org	https ftp	То же содержимое, что и на `ftp.FreeBSD.org`, `ftp` — устаревшее название; рекомендуется использовать `download.FreeBSD.org`.
git.FreeBSD.org	git по `https` и `ssh`	Подробнее в разделе использование git.
pkg.FreeBSD.org	pkg(8) через `http` и `https`	Официальные репозитории пакетов FreeBSD, используемые программой pkg(8).
vuxml.FreeBSD.org / **www.VuXML.org**	https	Страница проекта FreeBSD VuXML. `pkg audit` получает список уязвимостей из этой службы.
www.FreeBSD.org	https	Веб-сайт FreeBSD.

Все официальные зеркала поддерживают IPv4 и IPv6.

http://ftp-archive.FreeBSD.org не входит в инфраструктуру GeoDNS, размещается только в одном месте (США).

Проект ищет новые площадки; желающие выступить спонсорами, пожалуйста, свяжитесь с командой администраторов кластера для получения дополнительной информации.

CDN (сеть доставки контента) Fastly используется для обслуживания download.FreeBSD.org и pkg.FreeBSD.org, обеспечивая ускорение, кэширование и глобальное распределение пропускной способности для образов и пакетов FreeBSD.

Список зеркал, поддерживаемый сообществом и другими компаниями:

Страна	Имя сайта	Протоколы
Австралия ✉	ftp.au.FreeBSD.org	http http_v6 rsync rsync_v6

Страна	Имя сайта	Протоколы
	ftp3.au.FreeBSD.org	http ftp rsync
Бразилия	ftp2.br.FreeBSD.org	http rsync rsync_v6
	ftp3.br.FreeBSD.org	http ftp rsync
Болгария	ftp.bg.FreeBSD.org	ftp ftp_v6 rsync rsync_v6
Чехия	ftp.cz.FreeBSD.org	http http_v6 rsync rsync_v6
Дания	ftp.dk.FreeBSD.org	http http_v6 ftp ftp_v6 rsync rsync_v6
Финляндия	ftp.fi.FreeBSD.org	ftp
Франция	ftp.fr.FreeBSD.org	http http_v6 ftp ftp_v6 rsync rsync_v6
	ftp3.fr.FreeBSD.org	ftp
	ftp6.fr.FreeBSD.org	http ftp rsync
Германия	ftp.de.FreeBSD.org	http http_v6 https https_v6 ftp ftp_v6 rsync rsync_v6
	ftp1.de.FreeBSD.org	http http_v6 ftp ftp_v6 rsync rsync_v6
	ftp2.de.FreeBSD.org	http http_v6 ftp ftp_v6 rsync rsync_v6
	ftp5.de.FreeBSD.org	ftp ftp_v6
	ftp7.de.FreeBSD.org	http http_v6 ftp ftp_v6
Греция	ftp.gr.FreeBSD.org	http http_v6 ftp ftp_v6
	ftp2.gr.FreeBSD.org	http http_v6 ftp ftp_v6 rsync
Япония	ftp.jp.FreeBSD.org	http http_v6 ftp ftp_v6 rsync rsync_v6
	ftp2.jp.FreeBSD.org	ftp rsync rsync_v6
	ftp3.jp.FreeBSD.org	http rsync
	ftp4.jp.FreeBSD.org	ftp
	ftp6.jp.FreeBSD.org	http http_v6 ftp ftp_v6 rsync rsync_v6
Казахстан	mirror.ps.kz	http ftp

Страна	Имя сайта	Протоколы
	mirror.neolabs.kz	http ftp
Корея	ftp.kr.FreeBSD.org	http https ftp rsync
	ftp2.kr.FreeBSD.org	rsync
Латвия	ftp.lv.FreeBSD.org	http ftp
Нидерланды	ftp.nl.FreeBSD.org	http http_v6 ftp ftp_v6 rsync rsync_v6
	ftp2.nl.FreeBSD.org	http ftp rsync
	mirror.nl.altushost.com	https
Новая Зеландия	ftp.nz.FreeBSD.org	http ftp
Норвегия	ftp.no.FreeBSD.org	ftp ftp_v6 rsync rsync_v6
Польша	ftp.pl.FreeBSD.org	http http_v6 ftp rsync rsync_v6
Россия	ftp.ru.FreeBSD.org	http http_v6 ftp ftp_v6 rsync rsync_v6
	ftp2.ru.FreeBSD.org	https ftp rsync
Словения	ftp.si.FreeBSD.org	http http_v6 ftp ftp_v6
ЮАР	ftp2.za.FreeBSD.org	http http_v6 ftp_v6
	ftp4.za.FreeBSD.org	http ftp rsync
Швеция	ftp.se.FreeBSD.org	http http_v6 ftp ftp_v6 rsync rsync_v6
	mirror.se.altushost.com	https
Тайвань	ftp4.tw.FreeBSD.org	https ftp rsync
	ftp5.tw.FreeBSD.org	http ftp
Украина	ftp.ua.FreeBSD.org	http ftp ftp_v6 rsync rsync_v6
Великобритания	ftp.uk.FreeBSD.org	http http_v6 ftp ftp_v6 rsync rsync_v6
	ftp2.uk.FreeBSD.org	http http_v6 https https_v6 ftp ftp_v6
Соединение штаты Америки	ftp11.FreeBSD.org	http http_v6 ftp ftp_v6 rsync rsync_v6
	ftp14.FreeBSD.org	ftp rsync (Former official tier 1)

Страна	Имя сайта	Протоколы
	ftp5.FreeBSD.org	http http_v6 ftp ftp_v6

Текущий список протоколов, поддерживаемых общедоступными зеркалами, был последний раз обновлён 31 января 2022 года, и его актуальность не гарантируется.

A.2. Используя Git

A.2.1. Введение

Начиная с декабря 2020 года FreeBSD использует git в качестве основной системы контроля версий для хранения всего исходного кода и документации базовой системы. Начиная с апреля 2021 года FreeBSD использует git в качестве единственной системы контроля версий для хранения всей Коллекции портов FreeBSD.

Git обычно является инструментом разработчика. Пользователи могут предпочесть использование `freebsd-update` (“FreeBSD Update”) для обновления базовой системы FreeBSD.

В этом разделе показано, как установить Git в системе FreeBSD и использовать его для создания локальной копии репозитория исходного кода FreeBSD.

A.2.2. Установка

Git можно установить из Коллекции портов или в виде пакета:

```
# pkg install git
```

A.2.3. Запуск Git

Чтобы получить чистую копию исходников в локальный каталог, используйте `git clone`. Этот каталог с файлами называется *рабочим деревом*.

Git использует URL-адреса для указания репозитория. Существует три разных репозитория: `src` для исходного кода системы FreeBSD, `doc` для документации и `ports` для коллекции портов FreeBSD. Все три доступны по двум разным протоколам: HTTPS и SSH. Например, URL-адрес `https://git.FreeBSD.org/src.git` указывает на основную ветку репозитория `src`, используя протокол `https`.

Таблица 53. Таблица URL репозиториев Git FreeBSD

Элемент	URL Git
Репозиторий src через HTTPS (только для чтения)	`https://git.FreeBSD.org/src.git`
Репозиторий src через anon-ssh (только для чтения)	`ssh://anongit@git.FreeBSD.org/src.git`

Элемент	URL Git
Репозиторий документации через HTTPS (только для чтения)	`https://git.FreeBSD.org/doc.git`
Репозиторий документации через anon-ssh (только для чтения)	`ssh://anongit@git.FreeBSD.org/doc.git`
Репозиторий портов через HTTPS (только для чтения)	`https://git.FreeBSD.org/ports.git`
Репозиторий портов через anon-ssh (только для чтения)	`ssh://anongit@git.FreeBSD.org/ports.git`

Внешние зеркала, поддерживаемые участниками проекта, также доступны; дополнительную информацию можно найти в разделе Внешние зеркала.

Для клонирования копии репозитория исходного кода системы FreeBSD:

```
# git clone -o freebsd https://git.FreeBSD.org/src.git /usr/src
```

Опция `-o freebsd` указывает источник (origin); по соглашению в документации FreeBSD предполагается, что источником является `freebsd`. Поскольку первоначальное извлечение должно загрузить полную ветку репозитория с сервера, это может занять некоторое время. Пожалуйста, наберитесь терпения.

Изначально рабочее дерево содержит исходный код ветки `main`, которая соответствует CURRENT. Для переключения на 13-STABLE вместо этого:

```
# cd /usr/src
# git checkout stable/13
```

Рабочее дерево можно обновить с помощью `git pull`. Чтобы обновить /usr/src, созданный в примере выше, используйте:

```
# cd /usr/src
# git pull --rebase
```

Обновление происходит гораздо быстрее, чем извлечение, передавая только изменённые файлы.

A.2.4. Веб-интерфейс репозиториев

Проект FreeBSD использует cgit в качестве веб-браузера репозиториев: https://cgit.FreeBSD.org/.

A.2.5. Для разработчиков

Для получения информации о правах на запись в репозитории см. Руководство коммиттера.

A.2.6. Внешние зеркала

Эти зеркала не размещены на FreeBSD.org, но по-прежнему поддерживаются участниками проекта. Пользователи и разработчики могут клонировать или просматривать репозитории на этих зеркалах. Pull-запросы для репозиториев `doc` и `src` на GitHub принимаются; в остальном, рабочий процесс проекта с этими зеркалами всё ещё обсуждается.

Codeberg

- doc: https://codeberg.org/FreeBSD/freebsd-doc
- ports: https://codeberg.org/FreeBSD/freebsd-ports
- src: https://codeberg.org/FreeBSD/freebsd-src

GitHub

- doc: https://github.com/freebsd/freebsd-doc
- ports: https://github.com/freebsd/freebsd-ports
- src: https://github.com/freebsd/freebsd-src

GitLab

- doc: https://gitlab.com/FreeBSD/freebsd-doc
- ports: https://gitlab.com/FreeBSD/freebsd-ports
- src: https://gitlab.com/FreeBSD/freebsd-src

A.2.7. Списки рассылки

Основной список рассылки по общим вопросам использования git в проекте FreeBSD — freebsd-git. Подробнее, включая списки рассылки сообщений о коммитах, см. в главе Списки рассылки.

A.2.8. SSH ключи серверов

- Отпечатки ключей хоста gitrepo.FreeBSD.org:
  - Отпечаток ключа ECDSA — `SHA256:seWO5D27ySURcx4bknTNKlC1mgai0whP443PAKEvvZA`
  - Отпечаток ключа ED25519 — `SHA256:lNR6i4BEOaaUhmDHBA1WJsO7H3KtvjE2r5q4sOxtIWo`
  - Отпечаток ключа RSA — `SHA256:f453CUEFXEJAXlKeEHV+ajJfeEfx9MdKQUD7lIscnQI`
- Отпечатки ключей хоста git.FreeBSD.org:
  - Отпечаток ключа ECDSA — `SHA256:/UlirUAsGiitupxmtsn7f9b7zCWd0vCs4Yo/tpVWP9w`
  - Отпечаток ключа ED25519 — `SHA256:y1ljKrKMD3lDObRUG3xJ9gXwEIuqnh306tSyFd1tuZE`
  - Отпечаток ключа RSA — `SHA256:jBe6FQGoH4HjvrIVM23dcnLZk9kmpdezR/CvQzm7rJM`

Они также публикуются как записи SSHFP в DNS.

Приложение B: Библиография

Хотя справочные страницы (man-страницы) предоставляют точную информацию об отдельных компонентах операционной системы FreeBSD, они редко показывают, как объединить эти компоненты, чтобы обеспечить бесперебойную работу всей системы. В этом случае нет замены хорошей книге или руководству пользователя по администрированию UNIX®.

B.1. Библиография FreeBSD

- **Absolute FreeBSD: The Complete Guide To FreeBSD**, третье издание, издано No Starch Press, 2018. ISBN: 978-1593278922
- **FreeBSD Mastery: Storage Essentials**, издано Tilted Windmill Press, 2014. ISBN: 978-1642350098
- **FreeBSD Mastery: Specialty Filesystems**, издано Tilted Windmill Press, 2015. ISBN: 978-1642350111
- **FreeBSD Mastery: ZFS**, издано Tilted Windmill Press, 2015. ISBN: 978-1642350005
- **FreeBSD Mastery: Advanced ZFS**, издано Tilted Windmill Press, 2016. ISBN: 978-0692688687
- **FreeBSD Mastery: Jails**, издано Tilted Windmill Press, 2019. ISBN: 978-1642350241
- **FreeBSD Device Drivers: A Guide for the Intrepid**, издано No Starch Press, 2012. ISBN: 978-1593272043
- **The Design And Implementation Of The Freebsd Operating System**, второе издание, издано Pearson Education, Inc., 2014. ISBN: 978-0321968975
- **UNIX and Linux System Administration Handbook**, пятое издание, издано Pearson Education, Inc., 2017. ISBN: 978-0134277554
- **Designing BSD Rootkits**, издано No Starch Press, 2007. ISBN: 978-1593271428
- **FreeBSD Jails using VNETs**, опубликовано в gumroad

B.2. Издания о безопасности

- **The Book of PF: A No-Nonsense Guide to the OpenBSD Firewall**, третье издание, издано No Starch Press, 2014 год. ISBN: 978-1593275891
- **SSH Mastery: OpenSSH, PuTTY, Tunnels, and Keys**, второе издание, 2018 год. ISBN: 978-1642350029

B.3. История UNIX®

- Lion, John *Lion's Commentary on UNIX, 6-е изд. С исходным кодом*. ITP Media Group, 1996 год. ISBN 1573980137
- Raymond, Eric S.. *The New Hacker's Dictionary, 3rd edition*. MIT Press, 1996 год. ISBN 0-262-68092-0. Также известен как Jargon File

- Salus, Peter H. *A quarter century of UNIX*. Издано Addison-Wesley Publishing Company, Inc., 1994 год. ISBN 0-201-54777-5
- Simon Garfinkel, Daniel Weise, Steven Strassmann. *The UNIX-HATERS Handbook*. IDG Books Worldwide, Inc., 1994 год. ISBN 1-56884-203-1. Книга снята с печати, но доступна онлайн.
- Don Libes, Sandy Ressler *Life with UNIX* - специальное издание. Prentice-Hall, Inc., 1989 год. ISBN 0-13-536657-7
- *Генеалогическое древо BSD*. https://cgit.freebsd.org/src/tree/share/misc/bsd-family-tree или /usr/share/misc/bsd-family-tree на машине с FreeBSD.
- *Сетевая библиотека технических отчётов по компьютерным наукам*.
- *Старые выпуски BSD от группы исследования компьютерных систем (CSRG)*. http://www.mckusick.com/csrg/: Набор из 4 дисков включает все версии BSD от 1BSD до 4.4BSD и 4.4BSD-Lite2 (но, к сожалению, не 2.11BSD). На последнем диске также находятся окончательные исходные тексты вместе с файлами SCCS.
- Керниган, Брайан *Unix: История и мемуары*. Kindle Direct Publishing, 2020 год. ISBN 978-169597855-3

B.4. Периодические издания, журналы и газеты

- Admin Magazin (на немецком языке), издательство Medialinx AG. ISSN: 2190-1066
- BSD Now - Video Podcast, издано Jupiter Broadcasting LLC
- FreeBSD Journal, издаваемый S&W Publishing при спонсорской поддержке Фонда FreeBSD. ISBN: 978-0-615-88479-0

Приложение C: Ресурсы в Интернете

Разработка FreeBSD идёт слишком быстро, чтобы печатные издания могли эффективно информировать пользователей. Для отслеживания новостей лучше использовать электронные альтернативы печати.

Сообщество пользователей FreeBSD предоставляет обширную техническую поддержку — среди наиболее популярных и эффективных способов общения можно выделить форумы, чаты и электронную почту.

Ниже приведены наиболее важные точки взаимодействия. Раздел Сообщества на вики может содержать более актуальную информацию.

Пожалуйста, сообщите команде Список рассылки Проекта Документации FreeBSD о любом ресурсе, который является избыточным или ещё не указан ниже.

C.1. Вебсайты

- Форумы FreeBSD предоставляют веб-форум для обсуждения вопросов, связанных с FreeBSD, и технических дискуссий.
- Ссылка FreeBSD Wiki содержит различную информацию, которая ещё не вошла в Руководство.
- Портал документации предлагает гораздо больше, чем просто Руководство FreeBSD, здесь доступно более сорока книг и статей.
- FreeBSD Journal — это бесплатный профессионально редактируемый технический журнал, выпускаемый раз в два месяца Фондом FreeBSD.
- Канал BSDConferences на YouTube содержит коллекцию высококачественных видео с конференций BSD со всего мира. Это отличный способ увидеть выступления ключевых разработчиков о новых работах в FreeBSD.
- Отчёты о состоянии FreeBSD выпускаются раз в три месяца и отслеживают прогресс в разработке FreeBSD.
- Вот ссылка на группу в Reddit, посвящённую FreeBSD: r/freebsd.
- Super User и Server Fault, сервисы Stack Exchange для системных администраторов.
- FreeBSD Discord сервер — это платформа для общения и объединения сообщества, где участники сообщества FreeBSD могут общаться, получать поддержку или помогать другим, учиться, вносить свой вклад, сотрудничать и быть в курсе всего, что связано с FreeBSD.
- IRC-каналы — широко распространённый, технически зрелый, открытый стандарт текстового чата.

C.2. Списки рассылки

Почтовые рассылки — это самый прямой способ задать вопрос или начать техническое обсуждение среди заинтересованной аудитории FreeBSD. Существует множество рассылок,

посвящённых различным темам, связанным с FreeBSD. Отправка вопросов в наиболее подходящую рассылку, как правило, гарантирует более быстрый и точный ответ.

Технические обсуждения в списках рассылки должны оставаться техническими.

Все пользователи и разработчики FreeBSD должны подписаться на рассылку Список рассылки анонсов FreeBSD.

Для проверки возможностей списка рассылки FreeBSD направляйте сообщения в Тестовый список рассылки FreeBSD. Пожалуйста, не отправляйте тестовые сообщения в другие списки.

Если вы не уверены, в какой список отправить вопрос, обратитесь к разделу Как получить наилучшие результаты из списка рассылки FreeBSD-questions.

Перед отправкой сообщения в любой список, пожалуйста:

- Узнайте, как лучше всего использовать списки рассылки, например, как помочь избежать часто повторяющихся обсуждений, прочитав документ Часто задаваемые вопросы о списках рассылки (FAQ)
- поищите в архивах, чтобы убедиться, что кто-то уже не публиковал тоже самое.

Интерфейсы поиска в архивах включают:

- https://lists.freebsd.org/search (FreeBSD, экспериментальный интерфейс)
- https://www.freebsd.org/search/ (DuckDuckGo)

Обратите внимание, что это также означает, что сообщения, отправленные в списки рассылки FreeBSD, хранятся в архиве бессрочно. Если важна конфиденциальность, рекомендуется использовать одноразовый дополнительный адрес электронной почты и публиковать только общедоступную информацию.

Предоставляемые FreeBSD архивы:

- не отображают ссылки в виде ссылок
- не отображают встроенные изображения
- не отображают HTML-содержимое HTML-сообщений.

Списки публичных рассылок FreeBSD можно найти здесь.

C.2.1. Как подписаться или отписаться

На https://lists.freebsd.org нажмите на название списка, чтобы увидеть доступные опции.

Для отправки сообщения после подписки отправьте письмо на `listname@FreeBSD.org`. Сообщение будет распространено среди участников списка.

C.2.2. Основные правила списков

Все списки рассылки FreeBSD имеют определённые основные правила, которые должны соблюдаться всеми участниками. Несоблюдение этих правил приведет к двум (2) письменным предупреждениям от постмастера FreeBSD postmaster@FreeBSD.org, после чего, при третьем нарушении, отправитель будет удален из всех списков рассылки FreeBSD и заблокирован для дальнейших отправок. Мы сожалеем, что такие правила и меры вообще необходимы, но сегодняшний Интернет, кажется, довольно суровая среда, и многие не осознают, насколько хрупкими могут быть некоторые его механизмы.

Правила поведения:

- Тема любого сообщения должна соответствовать основному описанию списка рассылки, в который оно отправляется. Если список посвящён техническим вопросам, сообщение должно содержать техническое обсуждение. Постоянные не относящиеся к делу разговоры или флейм только снижают ценность списка рассылки для всех его участников и не допускаются. Для свободного обсуждения без определённой темы доступен список Список рассылки, посвящённый неформальным беседам о FreeBSD, который и следует использовать в таких случаях.
- Ни одно сообщение не должно отправляться более чем в 2 списки рассылки, и только в 2, если существует явная и очевидная необходимость отправить в оба списка. Для большинства списков уже существует значительное пересечение подписчиков, и, за исключением самых экзотических комбинаций (например, «-stable & -scsi»), нет никаких оснований отправлять сообщение более чем в один список одновременно. Если получено сообщение с несколькими списками рассылки в строке `Cc`, сократите строку `Cc` перед ответом. *Тот, кто отвечает, по-прежнему несёт ответственность за перекрёстную публикацию, независимо от того, кем был отправитель.*
- Личные оскорбления и ненормативная лексика (в контексте спора) не допускаются, и это касается как пользователей, так и разработчиков. Грубые нарушения сетевого этикета, такие как цитирование или перепубликация личной переписки без разрешения, когда такое разрешение не было и не могло быть получено, не приветствуются, но специально не пресекаются.
- Реклама продуктов или услуг, не связанных с FreeBSD, строго запрещена и приведёт к немедленному бану, если очевидно, что нарушитель занимается спамом.

C.2.3. Фильтрация в почтовых рассылках

Списки рассылки FreeBSD фильтруются несколькими способами, чтобы избежать распространения спама, вирусов и других нежелательных писем. Описанные в этом разделе действия по фильтрации не включают все методы, используемые для защиты списков рассылки.

На почтовых рассылках разрешены только определённые типы вложений. Все вложения с типом содержимого MIME, не указанным в списке ниже, будут удалены перед рассылкой письма.

- application/octet-stream

- application/pdf
- application/pgp-signature
- application/x-pkcs7-signature
- message/rfc822
- multipart/alternative
- multipart/related
- multipart/signed
- text/html
- text/plain
- text/x-diff
- text/x-patch

Некоторые из списков рассылки могут разрешать вложения с другими типами MIME-контента, но приведённый выше список должен быть применим для большинства списков рассылки.

Если многосоставное сообщение включает части text/plain и text/html:

- получатели получат обе части
- lists.freebsd.org будет отображать text/plain с возможностью просмотра исходного текста (исходник, включая сырой HTML среди частей).

Если text/plain не сопровождает text/html:

- Будет выполнено преобразование из HTML в простой текст.

C.3. Группы новостей Usenet

Помимо двух тематических групп новостей, посвящённых FreeBSD, существует множество других, где обсуждается FreeBSD или которые иным образом связаны с интересами пользователей FreeBSD.

C.3.1. Новостные группы, специфичные для BSD

- comp.unix.bsd.freebsd.announce
- comp.unix.bsd.freebsd.misc
- de.comp.os.unix.bsd (немецкий)
- fr.comp.os.bsd (французский)

C.3.2. Другие новостные группы UNIX®, представляющие интерес

- comp.unix
- comp.unix.questions

- comp.unix.admin
- comp.unix.programmer
- comp.unix.shell
- comp.unix.misc
- comp.unix.bsd

C.3.3. Система X Window

- comp.windows.x

Приложение D: Ключи OpenPGP

Здесь представлены открытые ключи OpenPGP должностных лиц FreeBSD.org. Эти ключи могут быть использованы для проверки подписи или отправки зашифрованной электронной почты одному из должностных лиц. Полный список OpenPGP ключей FreeBSD доступен в статье PGP ключи. Полный набор ключей можно загрузить по ссылке pgpkeys.txt.

D.1. Офицеры

D.1.1. Команда Офицера Безопасности <security-officer@FreeBSD.org>

```
pub   rsa4096/D9AD2A18057474CB 2022-12-11 [C] [expires: 2030-01-24]
      Key fingerprint = 0BE3 3275 D74C 953C 79F8  1107 D9AD 2A18 0574 74CB
uid                           FreeBSD Security Officer <security-officer@freebsd.org>
sub   rsa4096/6E58DE901F001AEF 2022-12-11 [S] [expires: 2028-01-24]
sub   rsa4096/46DB26D62F6039B7 2022-12-11 [E] [expires: 2028-01-24]
```

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGOVdeUBEADHF5VGg1iPbACB+7lomX6aDytUf0k2k2Yc/Kp6lfYv7JKU+1nr
TcNF7Gt1YkajPSeWRKNZw/X94g4w5TEOHbJ6QQWx9g+N7RjEq75actQ/r2N5zY4S
ujfFTepbvgR55mLTxlxGKFBmNrfNbpHRyh4GwFRgPlxf5Jy9SB+0m54yFS4QlSd0
pIzO0CLkjHUFy/8S93oSK2zUkgok5gLWruBXom+8VC3OtBElkWswPkE1pKZvMQCv
VyM+7BS+MCFXSdZczDZZoEzpQJGhUYFsdg0KqlLv6z1rP+HsgUYKTkRpcrumDQV0
MMuCE4ECU6nFDDTnbR8Wn3LF5oTt0GtwS0nWf+nZ1SFTDURcSPR4Lp/PKjuDAkOS
P8BaruCNx1ItHSwcnXw0gS4+h8FjtWNZpsawtzjjgApcl+m9KP6dkBcbN+i1DHm6
NG6YQVtVWyN8aOKmoC/FEm1CWh1bv+ri9XOkF2EqT/ktbjbT1hFoFGBkS9/35y1G
3KKyWtwKcyF4OXcArl6sQwGgiYnZEG3sUMaGrwQovRtMf7le3cAYsMkXyiAnEufa
deuabYLD8qp9L/eNo+9aZmhJqQg4EQb+ePH7bGPNDZ+M5oGUwReX857FoWaPhs4L
dAKQ1YwASxdKKh8wnaamjIeZSGP5TCjurH7pADAIaB3/D+ZNl2a7od+C1wARAQAB
tDdGcmVlQlNEIFNlY3VyaXR5IE9mZmljZXIgPHNlY3VyaXR5LW9mZmljZXJAZnJl
ZWJzZC5vcmc+iQJSBBMBCgA8AhsBBAsJCAcEFQoJCAUWAgMBAAIeBQIXgBYhBAvj
MnXXTJU8efgRB9mtKhgFdHTLBQJjlXeQBQkF3u+rAAoJENmtKhgFdHTLOVoQALS3
cj7rqYkHiV4zDYrgPEp9O1kAyGI8VdfGAMkDVTqr+wP4v/o7LIUrgwZl5qxesVFB
VknFr0Wp5g9h0iAjasoI5sDd6tH2SmumhBHXFVdftzDQhrugxH6fWRhHs0SaFYCk
Qt5nFbcpUfWgtQ35XTbsL8iENdYpjKXsSFQrJneGSwxIjWYTFn6ps/AI3gwR8+Bn
OffEFdYugJ049O6Vu6YBFJHrnMO7NbF4v95dVYuLtpMIaXWM+V9KITmhaBzFz5fM
Q7UOzcLlbxOYKNIWcp8QQk429mayKW5VUeUExUD1ZzBHn+P6ZG7QTMDu/RmBqiHo
ewCMVz4n9uXT5BiOngE4CvS0WQwHzK+k9MLpG2u/Bo9+LT0Ceh9Ou1rfU5+0tRwl
GyOFFjf3INS7I7gkcAwxQ7dzDItN/UQPZpg8y9mABU2x4enz0AvTnb61d/1dnTEr
tdNgU433he0ZnD1HurZCjBEWC656wv6iMdWcD8gjhMbmEpPmjvXcYlTO6zhEygSM
DiwdQCWK2W4++YJerA6ULBi3niNWBpofOFH8XylV56ruhjtHCo7+/3carcMoPOJv
lVZ1zCKxLro3TRBT15JTFBGqblRyTopFK3PuxW//GTnZOtpQEOV6yL4RAXcWeC1d
1hb5k/YxUmRF6XsDNEH4b08T8ZO8dV3dAV43Wh1oiQEzBBABCAAdFiEEuyjUCzYO
7pNq7RVv5fe8y6O93fgFAmObXVYACgkQ5fe8y6O93fiBlwf/W8y1XXJIx1ZA3n6u
f7aS70rbP9KFPr4U0dixwKE/gbtIQ9ckeNXrDDWz0v0NCz4qS+33IPiJg1WcY3vR
W90e7QgAueCo5TdZPImPbCs42vadpa5byMXS4Pw+xyT+d/yp2oLKYbj3En4bg1GM
```

```
w71DezIjvV+e01UR++u1t9yZ8LOWM5Kumz1zyQLZDZ8qIKt1bBfpa+E0cEqtNQWu
iGhQE3AHI8eWV+jBkg5y2zHRIevbWb1UPsj43lgkFtAGHk9rrM8Rmgr4AXr531iD
srBwauKZ/MElcF3MINuLH+gkPPaFHw/YIpLRLaZXZVsw3Xi1RNXI2n2ea29dvs/C
Lcf1vYkCMwQQAQgAHRYhBPwOh4rlr+eIAo1jVdOXkvSep+XCBQJjm14FAAoJENOX
kvSep+XC0DcP/1ZB7k9p1T+9QbbZZE1PJiHby3815ccH3XKexbNmmakHIn3L6Cet
F891Kqt9ssbhFRMNtyZ/k/8y8Hv5bKxVep5/HMyK+8aqfDFN0WMrqZh0/CiR6DJh
gnAmPNw/hAVHMHaYGII9kCrFfPFJ02FKoc81g9F08odb7TV+UlvRjkErhRxF+dGS
wQoO0RCbf0Z1cs7nd0Vb2z4IJh4XMxBjWc/uQ2Q9dH/0uRzwpAnR4YX+MG5YrX7Z
zBvDyR0r76iQwRSDKgioNgkr6R3rq1NZGdaj+8b0LzdOqtzKJ/eupDe3+H67e/EN
qymtreGjrubpiU9bKvYArisUqhE5KtguryvR6Qz9bj87nPg33DT3WWGVrwFRxBox
dbWzjQFv0wug8m4GAwVF7fPR5/eW7IHw8zvgn0vSPcZz7MZ4e6Y5jN4kA5/xWJYZ
Sps54qQWB+FA30unIXN68KqdIzONIbtaY3W4/JjJUCm4T+wEjKaH+wJX8w1DMjlg
mkTmGh/UrTyC1vXbPgk9Sy3cRTICR1T9z7W8UlmTtnKrUklrjlFR7SXzrEXzLGOX
Fm+NEHpHNXqzcm6c3QfzY/yQ9HSAQ/t7SUQ9caRePbDz3/msyPxtGFor9roQv6VN
wRXCyRgkH4Y5tPhJAQ8G/FxX+VXFb93QL0lfelb23/BBu6cUwW63SRn5iHUEExYI
AB0WIQQeB2Johg/5/ikUnJwDU9SVF1S13AUCZISO3wAKCRADU9SVF1S13NnqAP95
LA1Om9XSAKl76VtV+L3JPDdAwIdbNa0OsRT4Wm7U3wD/YoFrdHXVHHQFKwYeUUhj
XZcxnZLe9Ixo0/JP+RVFVw6JAlIEEwEKADwCGwEECwkIBwQVCgkIBRYCAwEAAh4F
AheAFiEEC+MydddMlTx5+BEH2a0qGAV0dMsFAmluhFoFCQ1lXf0ACgkQ2a0qGAV0
dMv8ixAAhYn3uxnjTCBZ5BJ6FMdwFXgGZMZVnL1qEoHA7HWB0ZoQlu29sFzBX97x
rkVZfWiU82V9/PYyHh4EY0hN9oOJMDDhSgi+DfdoEablIcIXR9uBBL/Snnak6uEF
gOD6VFH1T3o7iBmS/ixee2YFXq5QxAwEiANSAsxDm2rlyztPc5xM76fz5eEcgp1m
UDI7qFjoEkJVvRraJGVy3AUh4sr/94VAXPuCa7JnC6a8j9SD8XtzqIfcuqNBXXfA
m4ji4fN+KcDHtZPdkNLDBu1DEXQkOR/1S++NmXHYCEdQEuVOwLVsy6vQqgc+unju
Xh9sOrdiXUWylHh2hvgmwkjhMGOREiIRwNoA09m0OKZ2jo2dSlhNdVFRD5yhD92t
X9BarQbbjQtyCvoWnCJMFfw/vDnxfDdUdV3GxLhaOWvR8ItWQ7ts8tEZSjR/G76/
fHj1JRNuHRdhwype6ehieewJSMyzByhT0A3Njv3KLI4nwQj//MYzB5FJ9uOmKLcG
w7PB0yVjW4oRbG62+Qv6IQybCyrAL+12qgbUXbMsTqMwmsmdgtsqr/7OJNo3PBxX
2rUEio/mez8wuzNB0mJtlMynoMqXtbFX7AAXHk16fOGd6L6guW8+IhPedT3TvrjS
hqiQVvjOhM6rEYZ/YUfcIdkWJVpmE6+MRPVUufyMWeHaWqsKXoa5Ag0EY5V2yQEQ
AOqjzPpMUCGu8eElXnAd2PruC6hi+lc/yC90KqizxIuW6qLQBaAkTCWq7suYpDqo
ygn7YM3rL50S285WAECAXrcst/cVAqr0UH/e6p4iJCUIiXcfjd/wq20RnN/+VuvL
hjpCFLY5czfVS31D7Uh9MbC+zUTz8nVTiNCsAao0qSdfJDIzB4nSO+9xIsme/dLs
I5QlU5PdxOBV6HdEhCUXOoratJCbKAOlLxtPwyMKxmv4oZ7Mqlt10peKjhpBb97q
cIzJhHxujQZD0OmzIA6xoQ2eSCGdxCEDsZ09kr3Esw1AwKnQ51xmWpFWNFk6627M
1bo8+hzOz81CrTZhYrgE+1JXv6V1L2A9lMsimdE1BHNycDS+dBOpIB9qxXCwAab4
ykfvNxox/ZPDUrTy7v7mDI5uDNTNCYYsKCj1UidycOKzSziB9Oa2uvmMJ5XstgNB
f7Z8Cky1dtVd4ol6bU9L5nos9tbYeSXFi4bmcWB7AJiVCMq6N+LBbUKWGLglB4TU
1qhttpqv31X9V6ges5gARY/RuRTKsVyhwsn7SDcqmNKRy0im2AYakwEp7hT07ula
hOSLxjP+5hCf+nSJlwbxJ8ozwjjbzeN2yLlJSI00klkIFBNUDt3wzFRW/n6qlf+/
lepgzekfNrYMtfPB8AT07Z2A3U4xlgiV346dZymbY/EjABEBAAGJBHIEGAEKACYW
IQQL4zJ110yVPHn4EQfZrSoYBXR0ywUCY5V2yQIbAgUJAgIpAAJACRDZrSoYBXR0
y8F0IAQZAQoAHRYhBLYVJ36BCH33XIGDO25Y3pAfABrvBQJjlXbJAAoJEG5Y3pAf
ABrvuBsQAOlQFPXhx6whO4yw5ZizIS02YHhSVMVYKS2T9jPIKi1qxnEiEw9eKH0b
W0Oj0TEhZPyM2NJID7DRWK5r8+KsMu8jwm1fUmIrefAx6fCVfCWRECT1MlbL3jhh
6AcX/nK2e3Bn8vgExhzczO3JlvD6wPCc0FkpiY7yDB9ihu1+gbE5Hg6dvfttRXD
rbEdAifbNp9KYxDigxdlOb0S14hjCBysLWH5Su/khcIlkeuqZcI8TmDldnUb2OqT
CVpFhaNwsPSrHBzmb0s2sXo4FL03pLsOdwhi31W6kjk4KvW5FKrOpoEwUMKVNMf5
0DHdvonUoUHRSIc/cV5NqUWHwvc0T5031qk0CCRRa/+/iij/p2RG7c1mx7ZECj+j
ZfmvjSqT+WHJ1BFlNJMWyK4fdVRZWyCaoAecdbukwzDwUCUqHJFIWeFtbut7SOPx
cwg7sbnKNApAKdi491dvH75s/U/OwRYO/2P+ymHlqtyix2jq0ReSVYcQPXswQ8i2
```

```
ifX41F+xTSl4RWCBBeXB1Nxk3+HsV4Jnnp1zAJZ0KlKW/oJxbNFdI1TImkpr2p8i
oFf+aiePLvDkgeaG8vABgjoihPXWHVAMR8Z+GvBY/A6OdexpibkTvC/zDr0/Exs4
lsylZKDwvvFbctcpHVXBeCBQLX5vfLrsTkaCLWF/SV9OdMykvYKU7ZAP+gKEwhp+
HPFuOHZbOBhqFUdkfeCkdzX/QGdzTuz349roRhgz2vRfN7MtbtuzA6NWWhEWt5Dc
UgX/Y5I3Q4Z2bt3JiXQ6WJMgMMOXAr+XxtxyRRykc1HV3DQ/cq8OWYubNnIbgebP
NIFr2OIWKsR9yDaucZzpmLfzaMZUAu5hWmU9fIw5SIKGnQABBnNMhilfD+CkETp6
baTvjTK4rpaobjJdeCTrsWgfXRNC8x3hDvcrjPD70MylOGVQdx8GYChWJnCKXsLT
GX7KwdfxkjclTyzWvdcCemp0eLhamLGb9y1dtWdNIDcVCvZJy0lipHVUdFYYxb4i
LZJANL631tlPM6AA8sO1/L4mqEGnAIHVrUQd+2QkSiOl9mKlpgaR/fJz683BR5Qe
n9ywX0JPtBupqPW3t9VbO/uNxUqlHCeAhPi9NLOpujpYLfgW5QAfS3u0nkp5nrbk
CoQUua2q0Oj7J0mFmtWtcE1c9+THmFJVb8j2G9yQw3ADe3Qp9ALazP5nVDVri8NZ
BhHK1/KuBmRYZtcyfqXUnKoiiWAlm5rHaRiztW7e3wqm2oJu/RkEAagybutEuBWh
2Ej2+gDxjEKKtIKGu54lif4kqTwwjKTcN1ekGihwwgCMUkBSBeNXk1ClkzLFHwES
JCcFwdEgpVYQTKFsu0emYISyco3IpUajGzfUiQRyBBgBCgAmAhsCFiEEC+MydddM
lTx5+BEH2a0qGAV0dMsFAmWFy28FCQPyfaYCQMF0IAQZAQoAHRYhBLYVJ36BCH33
XIGDO25Y3pAfABrvBQJjlXbJAAoJEG5Y3pAfABrvuBsQAOlQFPXhx6whO4yw5Ziz
IS02YHhSVMVYKS2T9jPIKi1qxnEiEw9eKH0bW0Oj0TEhZPyM2NJID7DRWK5r8+Ks
Mu8jwm1fUmIrefAx6fCVfCWRECT1MlbL3jhh6AcX/nK2e3Bn8vgExhzczO3JlvD6
wPCc0FkpiY7yDB9ihu1+gbE5Hg6dvftttRXDrbEdAifbNp9KYxDigxdlOb0S14hj
CBysLWH5Su/khcIlkeuqZcI8TmDldnUb2OqTCVpFhaNwsPSrHBzmb0s2sXo4FL03
pLsOdwhi31W6kjk4KvW5FKrOpoEwUMKVNMf50DHdvonUoUHRSIc/cV5NqUWHwvc0
T5031qk0CCRRa/+/iij/p2RG7c1mx7ZECj+jZfmvjSqT+WHJ1BFlNJMWyK4fdVRZ
WyCaoAecdbukwzDwUCUqHJFIWeFtbut7SOPxcwg7sbnKNApAKdi491dvH75s/U/O
wRYO/2P+ymHlqtyix2jq0ReSVYcQPXswQ8i2ifX41F+xTSl4RWCBBeXB1Nxk3+Hs
V4Jnnp1zAJZ0KlKW/oJxbNFdI1TImkpr2p8ioFf+aiePLvDkgeaG8vABgjoihPXW
HVAMR8Z+GvBY/A6OdexpibkTvC/zDr0/Exs4lsylZKDwvvFbctcpHVXBeCBQLX5v
fLrsTkaCLWF/SV9OdMykvYKUCRDZrSoYBXR0yyOqEACitDvbkbfjaton6izr4T8Q
U2yvhJHkf4B6KeVDbKY1J4784OxXp2bJgPeF53SYBe8gm3YHjp8ULh4A/19U4hsw
yE8ymcm5nIs8OLyBdxkuBZJGEnzxH3woiyYqWH7991kzhEjUkuMgKLuTI1HiO0oL
MuPQNhUHOnWafSVPC0XO/tIL12OmoUuc7ligY9Z9AceFjTZOuHamixHAAc6hpxdI
W+yhC/qTpc2VK0niWeuQfq3453iRTf9MnR5Beztl3ZYRWcx7UiFuKGwZwBibNnNm
Us6GyQcJ5UTa1oeJcLqHi0Lf/r0jXo3wgJq7EZjjVyU+GI2ZVoDOaS6c4/OvLm62
XoeSlnn/dQxUcjUki+x8lb69IxSF1xAgsC/oNtFZYd5rHdlnqIBUYK0lLtSCXBkz
VeivSiQa0hL5on8LDu1nw2bXyW61yt/YxVb4FanMxAqdYVBhOfU0RaPNifH01rbb
4TwC9bTZN1LQ1KI/Swb/SruUE0RyT28fhYRtsReS2PnUODghJSFDJbwFbBZf6RKI
16q1xqKRRvxIWPm+lMOi1NLOKR9P+OKy9HmChMw0UJUcVl1cJ2xtRl3wi5t6AA6H
oNv/TrLeYVgMR9wYmKlpvjTQ5jTdrbHD1XP5jGsp8QsJMGja1m/7cryReCpcVxvI
meReaOdgz+zDmQqq3O5zuIkEcgQYAQoAJgIbAhYhBAvjMnXXTJU8efgRB9mtKhgF
dHTLBQJnhEEJBQkF0/JAAkDBdCAEGQEKAB0WIQS2FSd+gQh991yBgztuWN6QHwAa
7wUCY5V2yQAKCRBuWN6QHwAa77gbEADpUBT14cesITuMsOWYsyEtNmB4UlTFWCkt
k/YzyCotasZxIhMPXih9G1tDo9ExIWT8jNjSSA+w0Viua/PirDLvI8JtX1JiK3nw
MenwlXwlkRAk9TJWy944YegHF/5ytntwZ/L4BMYc3MztyZbw+sDwnNBZKYmO8gwf
YobtfoGxOR4Onb37bbUVw62xHQIn2zafSmMQ4oMXZTm9EteIYwgcrC1h+Urv5IXC
JZHrqmXCPE5g5XZ1G9jqkwlaRYWjcLD0qxwc5m9LNrF6OBS9N6S7DncIYt9VupI5
OCr1uRSqzqaBMFDClTTH+dAx3b6J1KFB0UiHP3FeTalFh8L3NE+dN9apNAgkUWv/
v4oo/6dkRu3NZse2RAo/o2X5r40qk/lhydQRZTSTFsiuH3VUWVsgmqAHnHW7pMMw
8FAlKhyRSFnhbW7re0jj8XMIO7G5yjQKQCnYuPdXbx++bP1PzsEWDv9j/sph5arc
osdo6tEXklWHED17MEPIton1+NRfsU0peEVggQXlwdTcZN/h7FeCZ56dcwCWdCpS
lv6CcWzRXSNUyJpKa9qfIqBX/monjy7w5IHmhvLwAYI6IoT11h1QDEfGfhrwWPwO
jnXsaYm5E7wv8w69PxMbOJbMpWSg8L7xW3LXKR1VwXggUC1+b3y67E5Ggi1hf0lf
TnTMpL2ClAkQ2a0qGAV0dMsNxRAAsuW1aLh+hgydW+iH6DmdQRMESsB1kE02kO14
```

```
62TAQaziIAvNoxw5h48xvyEnrDA8d+9IDMyxdrLmAbndUlSveMa9+EPiGHwr6VTy
FL8nA5F7DcFi4mjEyGKe18JcaAlYUtvHgWH6EjiX2iSXpsrJFEhtfFNolZ5sp9LF
I6hOBihSJxZK4sbMR7Q6IkDuAVpTFLiejBRlsXpFvTGL6040CtxbL5cqkVMYP38r
FMTuc3pGGJA4wb5EC1dGjUi6XjbYH7kuCAFyXqV9eQQP61x7K9W8qnXW+weCIMKf
SX7AcCtH1jXBAM6lqpPrh6amc+/rbg2eNA7DmgJnEY4apIcDB/b4khRMga2ozeGW
WyIvOaVvR2R7ALQ+Rgut85cM+4+Vl2PHmOzW/yYdHVb5REQItFR5COb/mGUqYhkC
tiV3nXo/K0uOQKu5SBbNzLuNvuwdn+Eimxjl18VnrGG7sjtUa0MLmtr62GiEVrhr
Dqa/biHp8LdWkAQjLZ4aTRh2XZiggaVFZHmkw3ILPyKKM21UXdM0YRk3TGVK8ODQ
y58ebPS4v9yYT9gUA9UDkDYeGcF2qjoDPVNvcG6H8jCSsPRl1KZwtqqITCOSAIAP
I4Nu97kO6nbOyQpYlwjd1MhvVXnP66mHSvmqaxbNGX1mF9B/yERkBkooNZrKuJSv
BTC2J1qJBHIEGAEKACYCGwIWIQQL4zJ110yVPHn4EQfZrSoYBXR0ywUCaW6EcgUJ
CaGkeQJAwXQgBBkBCgAdFiEEthUnfoEIffdcgYM7bljekB8AGu8FAmOVdskACgkQ
bljekB8AGu+4GxAA6VAU9eHHrCE7jLDlmLMhLTZgeFJUxVgpLZP2M8gqLWrGcSIT
D14ofRtbQ6PRMSFk/IzY0kgPsNFYrmvz4qwy7yPCbV9SYit58DHp8JV8JZEQJPUy
VsveOGHoBxf+crZ7cGfy+ATGHNzM7cmW8PrA8JzQWSmJjvIMH2KG7X6BsTkeDp29
+221FcOtsR0CJ9s2n0pjEOKDF2U5vRLXiGMIHKwtYflK7+SFwiWR66plwjxOYOV2
dRvY6pMJWkWFo3Cw9KscHOZvSzaxejgUvTekuw53CGLfVbqSOTgq9bkUqs6mgTBQ
wpU0x/nQMd2+idShQdFIhz9xXk2pRYfC9zRPnTfWqTQIJFFr/7+KKP+nZEbtzWbH
tkQKP6Nl+a+NKpP5YcnUEWU0kxbIrh91VFlbIJqgB5x1u6TDMPBQJSockUhZ4W1u
63tI4/FzCDuxuco0CkAp2Lj3V28fvmz9T87BFg7/Y/7KYeWq3KLHaOrRF5JVhxA9
ezBDyLaJ9fjUX7FNKXhFYIEF5cHU3GTf4exXgmeenXMAlnQqUpb+gnFs0V0jVMia
SmvanyKgV/5qJ48u8OSB5oby8AGCOiKE9dYdUAxHxn4a8Fj8Do517GmJuRO8L/MO
vT8TGziWzKVkoPC+8Vty1ykdVcF4IFAtfm98uuxORoItYX9JX050zKS9gpQJENmt
KhgFdHTLozcQAJCdYMPo1SPgxxRI40BCD4ekuA6ZMzdTuuso9XTTLf3moh+eu0Y+
Pv6ISOFI93g09V34H3XnuaOnA16DyApO6mCPZ58DunjRRYHv4wxbeff6vw9twPQb
bcdNgI2ZXSKjUef4FygkpUhy6075m6zoUfcsR/22Ggzd8Qw3chfoqiBZs7SlGwEH
sxkB+dJvgCMwYaLGqS9lNbHSEFnETb0ihFSHKDrK6pYrFnfsuArJN5owuTN+dzim
8g0agB3Pzs4A6+9Hpt9sN78VEY+ZzFhQD8knEgqSNbFIaNxsG5/L8/RcXuNyPvvz
5MIkhfacOX37YFfAhCORfN3PaVWYQ7oDnv8Nh69i9SkxXKUcwX0/iYaim8pwwM6a
HXhTlqBgKR5b5Hcx6Cwx1GTRUpdbJYThmz7LQgzjBIhmoGbxR3vyJjSyPGBZa3Ih
jAETYzMt7j0qGPiJUx7kcHeK7apAH1xB4DnfuPLSC1GjWptb31A22wjWy/75SJk8
0FoyYKPj5RqiQF+5CcSLRwTJEEi+9ggM8mUJiFNUOtfZgATgmCyEKG4fL1N4iyrj
fBha11/EO3LcH4t0rj5i8SkT+w8A0bgpOliLhnKpFQMy+cRewkzNIS2ynGlHqnKA
biuBEr1ji0iHNHczTMuqSYwraVHis+098D+1me5+9SaDp8yEIKgVuTUAuQINBGOV
dwcBEADKRVXM2aPWTzbIFu/dVFuaOsJ653dZyDvkuK3N+6tkxM/ChhwrsaOTcH32
drei59OASOeKW2qYNe2+2tIT5Gt+xxZC9vJ+FcjHcdArAJt0BH1LHSuscCldkrnG
HlXK1dbqXDeD3Vw+7LtSO/wOAllxSRtOO8ydGfPZSWYkBIgKu4IbxKIpwtLJvtGu
mqP6HKgoFgijNcgjzssyV5Xka79vUkZ2M4jEfqtlNMfcYPV5ehjMFJhcGyXPZzkV
byx+321Ov3bvBIKAGK4JRThWRv00qJUhlkUjImr89CUg/4vqxDeGiE+FnwXvi/Vs
tQJqzcYgDvsypsx0LtyUsQ53K/svoXHfltIKsNdBaCPY1i3C+5EfumtIjo7iUl0q
Katpdw2n4j8bQhZq+AhkIcwRzOnuP7/zTXdxdF6VWyekUKYe5ZH738/Ktpb1JR4X
sydKdcV5oX3/Qh43VTIZgJs+fopgQF5NpqLbTXym9KRRHp9cWNyv1DwOIPqD7Eo+
PeTLLyIPXiDAnAt/89EhFDFh/7Lcsicz6n6cnxsGXk3CI/b4NqPkZxr5nC1WAPxB
k6D5PdOZL/3Cu96MvG2cjgdMzIO0/T2loC1DtLKo6qNDMN5JxckoN6EV5XIpT0L3
z3V9Cb8C6XPQ1pa9ftRpAlydPJVroCXPegiULqSkW9SGwtKFFwARAQABiQI8BBgB
CgAmFiEEC+MydddMlTx5+BEH2a0qGAV0dMsFAmOVdwcCGwwFCQICKQAACgkQ2a0q
GAV0dMsMLQ/8CA5hHB3RZchUU6JlAY23itrs3GSIZJIC+fJXOt1D5XuYidhQb5mE
vtK0Eud7FRLHHJ+Sn5DsE68ipZb1m5dCM7oqn2sMQC3H1qX2SR80zEpwx34F7zjs
hTEK7qkQwBwK+zzQtAp209LBBdmRc2nfDPXy6rZF1MeJoBAjRieAlaoO+SmPLlM7
/wZkcBH+fOxU+f4gCHneTIxQdlRQ1/RgAXL002x8QmSMZ8Zd3LNeUsU9JQKH2drn
lOpFmFmUlxXmhDP+MpjycLNppldAJGyYIS1wp/TMd51+jU2WfR2q267vkc1t/x4N
```

```
w2XdCfb5KPD2nz9JdTVrYMl++Aqr04I808V056q0V83HMyu34XAtruItKaz7V13M
sou0sKaC4pGmQixRYVJjsfpkuU47YBchWQTcZzu4gMJM33E3AvbD6FmhZokH+quR
CGMxvlkKZ3I/NJsZdx1jj+qBY/4WSUWLHinjJLycy06PO6xXTOgrDKy8BE0envml
jNG7TxYpda11QAjTo7MtG8U1fDiRH8sZ2Jso75wbwdJHYmlRA3VYcYEqsywF+JoF
2uPf1ejP7szK/p9/wNSXZOFen1ZvEuXBSF8aFGrd3AeMRmk/Tlg96NTT6+nkvVaN
BLCZUZfNUhuvH0vkLZNn19u0+vGBuAGt8zH2cVLFbhS7U7kG0QvnyaSJAjwEGAEK
ACYCGwwWIQQL4zJ110yVPHn4EQfZrSoYBXR0ywUCZYXLqgUJA/J9aAAKCRDZrSoY
BXR0y6DtEACbExPX5JXm+2MzxLj/Rid48fDxnVpu5/vFDXa9uJEuoDl1VHz2a1mu
UqNtYXCMGW+D2Q6h0lUkJ0XIioUpxu4KZWDzRmePhVmy6/5gUWGq45FRS1Lgj3sU
iI8VdSkhiZ1rPWE1QCmQM++RqQjC+ow+m4gjkAkPOsKjnVr2SGUfGL7q68UeuAAQ
dK427GhvHDTlF9XMAH3S8LYX+uHUvtpt5ajRgx5D+XjS9ESMQuYF2BKdJb5pTB92
bjuVq3x+9z0w4PPW8V3roaxZkhMH+MVputN6IvP9qkL0ff+429KPre9TULlq1iBT
jWVloGQVBrKPrsk914GOGKihkd89kMyh2JyejO0r0mKaku0bXMXAlxdBxjXxEvHA
ILSlaRbW7UsF16fDvKC8o5lSDvzY1KWyzhnRSLGq7HsyxVfX9n6metyd+IclEoPC
62FVdXwOQyVKXm21+IS6XZNDDLYBuNwcCiBpSU6S+UZEP7kFN2FtpGoq8u6PtYpg
w9E0c0MbXY1TiOb55UzXHyWSCHy8eHawAQNdjr6V93Xz1etkfvAQJSRU+N9AhugC
EiSAaiURb7opJp2RlLxB5jlLPxS222l0z5CnO7UdliMJY0bzFfkJPOHFnF+mHITL
JTLogzNZrZUGYoN+FhxkhPswlp1wLJSB5UsKMe6n3TtCs9l53EmyAIkCPAQYAQoA
JgIbDBYhBAvjMnXXTJU8efgRB9mtKhgFdHTLBQJnhEEuBQkF0/ICAAoJENmtKhgF
dHTLJtUP/1dPcTrrUm1lOOCkbq3ZxirJOegxZal7YxmCagz6zg6V9zPCA39jLkyD
dsnzzL/fOGwGzgPUS+xGhlZUFFvUFj1ETLGShBUJY4X8USTvSaBg2CO7kQ50hIul
o1b4FJOToiIdChCOIxHGiws73iBHLMK1Q0gJ/u5xr3sD/Ea8e6jFDbywJbcYjMnx
l53mr0bN7H7rM3W/S002PJQlNMa7JWl6673wpGkW7JBYt0t9SPtWQf3sLgIR6n3N
d3kQYz8EDmpyRXqQMz+BheNX1UDfRrbLAEwg/nVejhYWI31iWziHD79u+8ArxiBb
nDPV0Y25AiGBUBOVvtTv2nLdzm49bAfdAvnFToAp4N+2Knzm2CxdfUROPmuzVAIf
xsR4/vE8VMr762llsnzakzA4xzEoTQ4xtKksxV+738lKZnwH8/umne/GpLPSzH/v
OohpLwYVxbO3M1b+D1lmFdflkoTVNmPcjGh986VlGkRHQdOxymy72uGNYnKllcfs
+sbNz77Km03ov5qF4mZO6IaWwZtIbSZZIttgEWKxl0iusOUULQmaAVtAxI+STQY5
Cul2wFt1RbgANSecC0ts9HiEeWnT/v0sUFG06vmRYjhnQT9YP+/N3u8xHcgNGHwN
ilPTnS87qVIsf8fiWHMZq2V53UjDm3dCIljocF5YGq2ySTEXjjkYiQI8BBgBCgAm
AhsMFiEEC+MydddMlTx5+BEH2a0qGAV0dMsFAmluhHIFCQmhpDsACgkQ2a0qGAV0
dMs5ag//V7mahuI9VikX1lG599GSjm1cXeDrZmGtamz9hYE7EzGlyxf5WQn83Sxu
2kNe9UbaLcIpKzZ9CWcb7NfC6odUIzJsO21qSKNuoP7FV9472JTNFMhaj1+o1P8b
v1nGOU4H9IH/y0Wqkuteef3t4le0iCtePMU3vXgfUFAmm/VF/tgmJQb2hLkXFNCQ
QwK//jjm9QHajaU2wEpvskarsRGfGBintBbUqUUxyPgirqAC8dFAWrjefmGePWEW
JfRLql39II+xbjEzADBB8yqnxLuaUQfVF7t2oAOfp09dP++bCua7FO1WfuFoa1Yw
um84/XszGgkxDRFkBxabpMCurPWgyhkGto4TqKdcWTMl4S1rTxt4Bm/6fqMn+04z
/AiXgU/iIbEKhvQUxsTxyNBLPFIRD7VmrqrOr3wkHgGHJgP0656ABq5J+T4mtFhX
uL0U9nGSm5lV9Hl1ana+Kcs4DqzgQO3QJk/SoulBDZ6at0pK2Y016JrvY51fYpx6
a/IOaXE2JRi8itVlpciDTjSOJs4eNlUx0C3b/LdzU5LZqHtBea1xWxlKPn0VFjYr
YNML6o5B6AL05AcySZj8kNZTWE2ZfOR2usLdiB8b+u2F7WGXZBiQYSnr75WsN94S
nYwO46v5SlzGqTwEN/UBfTkKPklVNgeOMn7fgjziEtMmtBlb7ws=
=ijsc
-----END PGP PUBLIC KEY BLOCK-----
```

D.1.2. Секретарь основной команды <core-secretary@FreeBSD.org>

```
pub   rsa4096/4D632518C3546B05 2024-02-17 [SC] [expires: 2028-02-08]
      Key fingerprint = 1A23 6A92 528D 00DD 7965  76FE 4D63 2518 C354 6B05
```

```
uid                           FreeBSD Core Team Secretary <core-
secretary@FreeBSD.org>
sub   rsa4096/CABFDE12CA516ED2 2024-02-17 [E] [expires: 2028-02-08]
```

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGXQ1o8BEAC+Rcg8cmVxuP17Vu+q5KgCx/XiulQuqKXAqqBlYCH2jqk6DINP
yFrREGBhzd/qNmlAYEahQ4Zgl0bUZNTrZVDyzicOvPP0jH+KSTQwRs7NOawEdlVO
cyHrwDCPEqf5ZzD4NhfTriEOw+j0pEH/onitUGvoQRtx15xWyaJQxDEBMTYMLewE
86D1bltwnTNczE3UZb7oQLJXkAX5hcLtou70XJGgZITvJkK+kp/xot2eFjnqRz/u
WeXnKhYAmC07EKwZ1uw047eHKwMMRBYqzApLwoQtfE430Kxf2q8de64x8zDbi6YM
1J4r8OAxOtHVyfJ0j7Q23DEZz0VVb4b1Tx5OG2Re/KSNvqI0awJO4TcRmOR88OyY
dzyXgnX6Sa7GVQY1FXvn7vtFuDAt7egZOzeomSHL9bdX07LTQ4UtM88EV9wm3q4q
smoatV9jsvPQ1zxCU3aQD/5eWTJH2/kz1LIuBL/Qi5XQpJn91lBtUWJrCgkHWPGu
f//rnnXmsG7DAChW+yZ7cFO8lfNa8sFhPqSxCYphWmJTrvadyQtDngB8JakWdnmK
pfGS6y5lel+181vw38ZZKt04AKM+nDY8051lBM7Q9Q6kTLI33UZeImndx5xYukVD
kV6aQ31HYfEark15c7iEz+OAcwFnM2ntXMt7kKGd40CqzusiPcQkPqPbAQARAQAB
tDhGcmVlQlNEIENvcmUgVGVhbSBTZWNyZXRhcnkgPGNvcmUtc2VjcmV0YXJ5QEZy
ZWVCU0Qub3JnPokCVwQTAQoAQQIbAwgLCQ0IDAcLAwUVCgkICwUWAwIBAAIeBQIX
gBYhBBojapJSjQDdeWV2/k1jJRjDVGsFBQJnp8PiBQkHeofTAAoJEE1jJRjDVGsF
GMIQALhj+mNpH8OmTFeihQ6t9P8un3llz6Wmqe/Q+ULWeqJvV/uC1J5T9fnoGhwF
MgECuguXYJtoYQ16KXnsSOs1tcqMOK9GtEtFJTGe2DtfLiBednwUvu9j3HmTlwLN
M+7rqkiC0HCg2qSjcMjxbVbA5BSgNkgfyTSO2YdjfaZ+ceiHwo/qa5xWE6i2dMR1
PLGMMHTeELdtdSH6VR9/3h0qt3qzwdMBRCVAQHbim7CqwjUH9jg+IOySXl81jNoB
xuVZR3pKshyY4Oxo1dK+W86Uiff/+c4jCAxokGWIR7C4MkZZWUQqV7920gkZFC/5
qzpT1A1/sFUg8HfFTr0vCoPSVFWn2+Tto3vr78DICVaAf02aYAFlyKK18BMCoHkS
hDDO+/JQZmvHOIgEYK+T2WN1c9gm9IDJzGZuH0X2C/vkREnJKkccJfB4pXuN10wt
fqyP9fn8h6+/t+sv3qNLm4d8fkmLXofulG3WB4i/F+Hip2rjPvvBCF7zl4xy6cJ2
xVY5HUOBTqmIwVhYwUpXaqNoWa/qJBLTuot3z7ciKmKX6Lq+Dze5dzhrPNl/CalC
HBf3miHRK3TZbYLooG3bcEWgxU2BnBi3vl5NpCoUOKkPYRlALXi2TyHmPxO7oT4e
mIzS6BgnXOqO+AXvbKKfayuSePdBqkNMK/SMC4Dylkf6Xj25iQIzBBABCgAdFiEE
EBpxaxYrAOVb7eoFrbv4YQo3ibcFAmbu7x4ACgkQrbv4YQo3ibcr5A//TIcbD/EW
YJz0zrUQdc9xG3UNfU6uHmQzAuUy5ginevyqv0TSso5qvSkOHvdxbi41rfMiB2RJ
V3q0n0PSvHFld89fOTZUMZXaPvozCiBYWScrt+KA/2pq3K8mUumHS+IFtpHLL2Tu
+gI8XCHUFxO9HUThM/rFlIyEdzoctgmqQ7IG4uZG+o2J3w09llhDAUe0vraJK1On
p9yFACnRrhqvl41JeUWcv9MH9JcwHUqtUo9WLTCIb+hkByTOyRfHBYpYw//bdXdf
6rkCwKVWpyMDbk61zq2VsS1kqbP9IH/A8CsBnA6mg+zPq2i7HIFw8Swj4OGJcIvG
a9ubUYJRjDhX/vBpNrtncANZ88FQmA+Maq0vu0LS5IIGyIKkvd1fKIsyvBDDa9kE
nfCW0XMkJA0Gf+kxdB+elXQHBwKO2sr5BiKnJT5lJq3Yu9fxxGBnf93yiN4E9bmF
gG7cZxpyb1Bp76TJhLcANyybOTjCtiNRrgqeaSx9/6hSPfPigGXIne0H2lmJO6oq
jUrsYmFiWvU9sc7AcPVw/eHG8FgW35TuwKX71z8w994iaahUPNcSVyXOUD+QNROv
HhGUXrc81t613rivh22N0NZpNubVatq43KV7+/bnPyWBIi1Awh3vIFsNNSQSrYxF
lUuQaAHQXTeZMZ/7npE4t86seMt0T7BGn765Ag0EZdDWjwEQAL3VwFifpnRCYzQ4
VZ1dAjp8w542iRprqeA+C3tvNbk2OvKpN3DIc49l2bgNZ/VI7/T58lEKrFsgLK4w
AWtv18OV7xuh0AuOblmq5MvcPrUelkPj5HA7M6Ng/rAubuHfwdP/IjIrzzY6+XDh
fP9N5KIv9VRJYT23BbvLPeit4J4tQEB/NpxL3L6zI75qq4R3T/EkHP6Y9Buup9Lr
isJlcxkSK+CyORCgTpEz6fWsXiTDgS7cTaQ969XCygBpj4wRQzwkBdokxo1wsiFt
4zLt07/PrPYjeHltTDkrF9XDNhLNJ5GliHnd01oHg1j5/n+9Osvh1maoFwuBxXTN
nTZ2P+7RKLBaVQVSkUa5KJbXoM3v7bMbXM5aLs2XjglrAzrZOV+Y7zOu5UYQdpXd
7x8XAEtEozRJzt7dosQIhKx9h4LlltFFulDUpf5VCvcUEoAzMbiX0aju9n5Rwsql
```

```
DHatU9Mm3W8OdFXj1foIXZD+VX2Jp9DgxPLoAJ0CWhPXJ8f+WFSJZCjWoPJEJKWY
0EOxiXyAuvAyniAZAC/eKZkfGckXmu7edRgYbRTTwPmZ/axa/k9aHsHLwmbxuZo/
xxQXzqU70EleHZ31A3mOqsC1epjN6dn4AFKgvVesP/K/fbvSsfsfiABS2A/68ne4
zJG73Tnk4L6J79vrXc2iMJbmdg3ZABEBAAGJAjwEGAEKACYCGwwWIQQaI2qSUo0A
3Xlldv5NYyUYw1RrBQUCZ6fEFQUJB3qIBgAKCRBNYyUYw1RrBQd9EACMVckxzy4w
aUGlERguJ+kslS8MkMjNqnfDPLRDVxbUxNvbbhw7/u9blM65DCGCeNLc2n4oiu5C
E3I095AKmvq/0dOa81mEEdZkC1CVc64bXWbEyz5AtSHUpgdxRso6C+YopndSiz1T
WcIagQRXfWaw3FBWPooA87gmibmSmCegCtqx+uyc5QxX2eFI8mRK7vlfnGpYKHs0
D1/yUSGQOwoNRJ5FYm+ynfDE3FzHEQL7lv1vpv1k3xNKfBziMMg4IMEBKNHV4VKN
qJpa8UCodeTGSWQdNnCeCWPsxz5oQjCcVH9Z3e8sMpWLHhRcBZzSwXUOws2GbRMH
xHwrfrpHJcrBhe64pgfGOvZLUJ9BDs+8egTHsqRFacipbtTR+hhVhuJEHDaQQWuM
8IRHj9HIuTAczET8JTDHTMIoo8DZdOtiW/YgqCDwYghkI77d12oNQqYoeJ2HiqbK
cGzwCpsR0A+p/iOAxJG13tsxqZV8TQ8iTokWG6ACtZ7sfeWEhxqMbUKUMgogZn0I
3n1kV+tUZC6BQrIYI7TiKg95wLZsIydeoIsQoNZwvyKAXfVmQ62YjIX8njZwN+07
8/ipUPJxCYa8zL8BZyDmoFJqa3y9z+11+vtiZ9t+aTwGvjpHDwyeCJco7go9cU3m
GRFZYciqIoG4n3tl8Pob15vFlVqk47rRqg==
=8TzT
-----END PGP PUBLIC KEY BLOCK-----
```

D.1.3. Секретарь Группы менеджеров дерева портов <portmgr-secretary@FreeBSD.org>

```
pub   ed25519/E3C401F60D709D59 2023-03-06 [SC] [expires: 2027-03-05]
      Key fingerprint = BED4 A1D3 6555 B681 2E9F  ABDA E3C4 01F6 0D70 9D59
uid                           FreeBSD Ports Management Team Secretary <portmgr-
secretary@FreeBSD.org>
sub   cv25519/2C92B55E27A641C3 2023-03-06 [E] [expires: 2027-03-05]
```

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mDMEZAXJvxYJKwYBBAHaRw8BAQdASFAC20WL3R1T6uNyGMZbfJCxDkcP4C5vi3Op
tcZ2fbq0R0ZyZWVCU0QgUG9ydHMgTWFuYWdlbWVudCBUZWFtIFNlY3JldGFyeSA8
cG9ydG1nci1zZWNyZXRhcnlARnJlZUJTRC5vcmc+iJYEExYKAD4WIQS+1KHTZVW2
gS6fq9rjxAH2DXCdWQUCZAXJvwIbAwUJB4TOAAULCQgHAwUVCgkICwUWAwIBAAIe
BQIXgAAKCRDjxAH2DXCdWYN1AP43TjyfZtZ3DLYT++g0+SuPsoO/3yWVybA+UmFL
zb8MngEA+LLNUfvEwCuXS/soh+ww5bpfmi3UUmeGiQEAXug3iA+JATMEEAEKAB0W
IQT7N0XIbxXo7ayBMvzYKU7Du8TX1QUCZAXLkwAKCRDYKU7Du8TX1XHMB/9R1MX4
6zMgpKqPPt76GOI+eGEdBK6bY8aJZjQGdqTh9f6VtXVoTGIG7cvhc9X8tDBoB0PT
2KZWheF51AV1+NHU4HwLAQ1BMebrFvWSfkw4xg4fBGwDhz9/GN85No+Js772V5ey
8lRiL6meRVWxMlLyWcxGd8JjcC5yX/iAUQ3SBGCLqW7unWjjg7CTd+AMBwcqPGrv
ax8q6eFVguJcHJAjMnKf6HAy4cpK3s+uMoUBCGnszSN12B3ysKfyC4pNO/pix5tA
Q5v8aRqTeFPh5zmNhWo0KGPzplTPqRQSHDl7GDQC8Ru3MhzFkeWzHsexjZVwS6W2
DPcYpuuAsA0XOZIZiQIzBBABCgAdFiEEEBpxaxYrAOVb7eoFrbv4YQo3ibcFAmQF
0u0ACgkQrbv4YQo3ibccwg/9F2Xuic3nhKxRbB3mJeDo6SYQETa/Gh1qQ34+8zlt
8UMazOx67gnYQfy+pXjro6eQ2up0a4eUYezcNOudqAQD21nRz3HA6EQVNcE/TzEA
xl5CJntTaLOt7S+EDXFW5BuQIvhhoMGgm8+WNVgA0EJ7tfL0OcYBSvr19fqwChEn
9c14cSk6mgHSsleP5NvskYN053pxHwy0LTSb8YBBv52th37t/CRFC1363rS5q+D7
JixFopd1O5pKpA5ipvE4gGgRjPtwjx0SjjepwK/3fuhEJQQyKzTIKlMfu2Dj/iR2
```

```
Li1Sfccau5LQXOj9fUITU3u1YG7yrm8VGzT7ao4d+KRwgMLjd2pLqiGIbbJwGBiP
FRmtilWQoeIlmSlFX4obAA517DOK0pW1mH8+eEn4EJd3SekT3yzFyKTASv0J48Z8
3F928xg+eZvHxVC0t1J+J5IG0gt3EEncuWKIPQGR7PiQbti6R3FQVTz6WfMWOebP
Qi0E9F/Aqakr6Vj2sKGrDq+ebpaF5G8Yw1YrUl2IDiPzkCegp3ZbI0wh11Xvzhi8
LXPQGK4jBQas4G8cegfitzmtdGRHYrbMv0R9I4mvaL+WlOuD2AvyVG28lguqVhnN
AZP+ohdquYyX2CNCVvbKWAtXo6Ur0vWG8BL8m6defAtEkIwVBALaOHQOSI3aNUz4
lwy4OARkBcm/EgorBgEEAZdVAQUBAQdAsefmSfxEOdOr02+K/6noYCuJ1FeAWVz6
jFYQ+9w6jggDAQgHiH4EGBYKACYWIQS+1KHTZVW2gS6fq9rjxAH2DXCdWQUCZAXJ
vwIbDAUJB4TOAAAKCRDjxAH2DXCdWRl4AP9h5ot212BK29S6ZcMBhHvmtF5PG1oD
c7LnZycSRmbFiwEAndCMpAGOhDW8iVgDd0wLQq/ZMPe+xccfG1b3zFH2EgE=
=iiAT
-----END PGP PUBLIC KEY BLOCK-----
```

D.1.4. doceng-secretary@FreeBSD.org

```
pub   rsa2048/E1C03580AEB45E58 2019-10-31 [SC] [expires: 2022-10-30]
      Key fingerprint = F24D 7B32 B864 625E 5541  A0E4 E1C0 3580 AEB4 5E58
uid                           FreeBSD Doceng Team Secretary <doceng-
secretary@freebsd.org>
sub   rsa2048/9EA8D713509472FC 2019-10-31 [E] [expires: 2022-10-30]
```

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQENBF27FFcBCADeoSsIgyQUY8vREwkTikwFFlNg31MVy5s/Nq1cNK1PRfRMnprS
yfB62KqbYuz16bmQKaA9zHN4FGfiTvR6tl66LVHm1s/5HPiLv8sP14GsruLro9zN
v72dO7a9i68bMw+jarPOnu9dGiDFEI0dACOkdCGEYKEUapQeNpmWRrQ46BeXyFwF
JcNx76bJJUkwk6fWC0W63D762e6lCEX6ndoaPjjLBnFvtx13heNGUc8RukBwe2mA
U5pSGHj47J05bdWiRSwZaXa8PcW+20zTWaP755w7zWe4h60GANY7OsT9nuOqsioJ
QonxTrJuZweKRV8fNQ1EfDws3HZr7/7iXvO3ABEBAAG0PEZyZWVCU0QgRG9jZW5n
IFRlYW0gU2VjcmV0YXJ5IDxkb2Nlbmctc2VjcmV0YXJ5QGZyZWVic2Qub3JnPokB
VAQTAQoAPhYhBPJNezK4ZGJeVUGg5OHANYCutF5YBQJduxRXAhsDBQkFo5qABQsJ
CAcDBRUKCQgLBRYDAgEAAh4BAheAAAoJEOHANYCutF5YB2IIALw+EPYmOz9qlqIn
oTFmk/5MrcdzC5iLEfxubbF6TopDWsWPiOh5mAuvfEmROSGf6ctvdYe9UtQV3VNY
KeeyskeFrIBOFo2KG/dFqKPAWef6IfhbW3HWDWo5uOBg01jHzQ/pB1n6SMKiXfsM
idL9wN+UQKxF3Y7S/bVrZTV0isRUolO9+8kQeSYT/NMojVM0H2fWrTP/TaNEW4fY
JBDAl5hsktzdl8sdbNqdC0GiX3xb4GvgVzGGQELagsxjfuXk6PfOyn6Wx2d+yRcI
FrKojmhihBp5VGFQkntBIXQkaW0xhW+WBGxwXdaAl0drQlZ3W+edgdOl705x73kf
Uw3Fh2a5AQ0EXbsUVwEIANEPAsltM4vFj2pi5xEuHEcZIrIX/ZJhoaBtZkqvkB+H
4pu3/eQHK5hg0Dw12ugffPMz8mi57iGNI9TXd8ZYMJxAdvEZSDHCKZTX9G+FcxWa
/AzKNiG25uSISzz7rMB/lV1gofCdGtpHFRFTiNxFcoacugTdlYDiscgJZMJSg/hC
GXBdEKXR5WRAgAGandcL8llCToOt1lZEOkd5vJM861w6evgDhAZ2HGhRuG8/NDxG
r4UtlnYGUCFof/Q4oPNbDJzmZXF+8OQyTNcEpVD3leEOWG1Uv5XWS2XKVHcHZZ++
ISo/B5Q6Oi3SJFCVV9f+g09YF+PgfP/mVMBgif2fT20AEQEAAYkBPAQYAQoAJhYh
BPJNezK4ZGJeVUGg5OHANYCutF5YBQJduxRXAhsMBQkFo5qAAAoJEOHANYCutF5Y
kecIAMTh2VHQqjXHTszQMsy3NjiTVVITI3z+pzY0u2EYmLytXQ2pZMzLHMcklmub
5po0X4EvL6bZiJcLMI2mSrOs0Gp8P3hyMI40IkqoLMp7VA2LFlPgIJ7K5W4oVwf8
khY6lw7qg2l69APm/MM3xAyiL4p6MU8tpvWg5AncZ6lxyy27rxVflzEtCrKQuG/a
oVaOlMjH3uxvOK6IIxlhvWD0nKs/e2h2HIAZ+ILE6ytS5ZEg2GXuigoQZdEnv71L
```

```
xyvE9JANwGZLkDxnS5pgN2ikfkQYlFpJEkrNTQleCOHIIIp8vgJngEaP51xOIbQM
CiG/y3cmKQ/ZfH7BBvlZVtZKQsI=
=MQKT
-----END PGP PUBLIC KEY BLOCK-----
```

Глоссарий FreeBSD

Этот глоссарий содержит термины и аббревиатуры, используемые в сообществе FreeBSD и документации.

A

ACL

См. Access Control List.

ACPI

См. Advanced Configuration and Power Interface.

AMD

См. Automatic Mount Daemon.

AML

См. ACPI Machine Language.

API

См. Application Programming Interface.

APIC

См. Advanced Programmable Interrupt Controller.

APM

См. Advanced Power Management.

APOP

См. Authenticated Post Office Protocol.

ASL

См. ACPI Source Language.

ATA

См. Advanced Technology Attachment.

ATM

См. Asynchronous Transfer Mode.

ACPI Machine Language

Псевдокод, интерпретируемый виртуальной машиной в рамках соответствующей спецификации ACPI операционной системы, обеспечивающий прослойку между базовым оборудованием и документированным интерфейсом, предоставляемым ОС.

ACPI Source Language

Язык программирования, на котором написан AML.

Access Control List

Список разрешений, связанных с объектом, обычно файлом или сетевым устройством.

Advanced Configuration and Power Interface

Спецификация, которая предоставляет абстракцию интерфейса, представляемого оборудованием операционной системе, так что операционная система не должна знать ничего о базовом оборудовании для его наиболее эффективного использования. ACPI развивает и заменяет функциональность, ранее предоставляемую APM, PNPBIOS и другими технологиями, а также предоставляет средства для управления энергопотреблением, приостановкой работы машины, включением и отключением устройств и т.д.

Application Programming Interface

Набор процедур, протоколов и инструментов, определяющих каноническое взаимодействие одной или нескольких частей программы; как, когда и почему они работают вместе, а также какие данные они используют или обрабатывают.

Advanced Power Management

API, позволяющий операционной системе работать совместно с BIOS для управления питанием. APM был заменён более универсальным и мощным стандартом ACPI для большинства применений.

Advanced Programmable Interrupt Controller

Advanced Technology Attachment

Asynchronous Transfer Mode

Authenticated Post Office Protocol

Automatic Mount Daemon

Демон, автоматически монтирующий файловую систему при обращении к файлу или каталогу в этой файловой системе.

B

BAR

См. Base Address Register.

BIND

См. Berkeley Internet Name Domain.

BIOS

См. Basic Input/Output System.

BSD

См. Berkeley Software Distribution.

Base Address Register

Регистры, определяющие диапазон адресов, на которые будет отвечать устройство PCI.

Basic Input/Output System

Определение BIOS немного зависит от контекста. Некоторые называют так микросхему ПЗУ с базовым набором подпрограмм, обеспечивающих интерфейс между программным обеспечением и оборудованием. Другие подразумевают под BIOS набор подпрограмм, содержащихся в микросхеме, которые помогают в загрузке системы. Некоторые могут также называть BIOS экран, используемый для настройки процесса загрузки. BIOS специфичен для PC, но в других системах есть нечто подобное.

Berkeley Internet Name Domain

Реализация протоколов DNS.

Berkeley Software Distribution

Это название, которое Группа исследования компьютерных систем (CSRG) из http://www.berkeley.edu [Университета Калифорнии в Беркли] дала своим улучшениям и модификациям 32V UNIX® от AT&T. FreeBSD является наследником работ CSRG.

Bikeshed Building

Явление, при котором множество людей высказывают своё мнение по простой теме, в то время как сложная тема получает мало или вообще никакого обсуждения. Подробнее о происхождении термина см. в FAQ.

C

CD

См. Carrier Detect.

CHAP

См. Challenge Handshake Authentication Protocol.

CLIP

См. Classical IP over ATM.

COFF

См. Common Object File Format.

CPU

См. Central Processing Unit.

CTS

См. Clear To Send.

Carrier Detect

Сигнал RS232C, указывающий на обнаружение несущей.

Central Processing Unit

Также известен как процессор. Это мозг компьютера, где происходят все вычисления. Существует множество различных архитектур с различными наборами инструкций. Среди наиболее известных — Intel-x86 и его производные, Arm и PowerPC.

Challenge Handshake Authentication Protocol

Метод аутентификации пользователя, основанный на секрете, разделяемом между клиентом и сервером.

Classical IP over ATM

Clear To Send

Сигнал RS232C, разрешающий удалённой системе передавать данные.

См. Also Request To Send.

Common Object File Format

D

DAC

См. Discretionary Access Control.

DDB

См. Debugger.

DES

См. Data Encryption Standard.

DHCP

См. Dynamic Host Configuration Protocol.

DNS

См. Domain Name System.

DRM

См. Direct Rendering Manager.

DSDT

См. Differentiated System Description Table.

DSR

См. Data Set Ready.

DTR

См. Data Terminal Ready.

DVMRP

См. Distance-Vector Multicast Routing Protocol.

Discretionary Access Control

Data Encryption Standard

Метод шифрования информации, традиционно используемый для шифрования паролей в UNIX® и функции crypt(3).

Готовность терминального оборудования (Data Set Ready)

Сигнал RS232C, передаваемый от модема к компьютеру или терминалу, указывающий на готовность к отправке и приему данных.

См. Also Data Terminal Ready.

Готовность терминального оборудования (Data Terminal Ready)

Сигнал RS232C, передаваемый от компьютера или терминала к модему, указывающий на готовность к отправке и приёму данных.

Debugger

Интерактивное средство, поддерживаемое ядром системы, для проверки состояния системы, часто используемое после аварии системы для выяснения обстоятельств сбоя.

Differentiated System Description Table

Таблица ACPI, предоставляющая базовую информацию о конфигурации основной системы.

Distance-Vector Multicast Routing Protocol

Domain Name System

Система, преобразующая удобочитаемые имена хостов (например, mail.example.net) в интернет-адреса и обратно.

Direct Rendering Manager

Модуль ядра drm(7) предоставляет клиентским приложениям прямой доступ к графическому оборудованию через инфраструктуру Direct Rendering (DRI).

Dynamic Host Configuration Protocol

Протокол, который динамически назначает IP-адреса компьютеру (хосту), когда он запрашивает их у сервера. Назначение адреса называется "арендой".

E

ECOFF

См. Extended COFF.

ELF

См. Executable and Linking Format.

ESP

См. Encapsulated Security Payload.

Encapsulated Security Payload

Executable and Linking Format

Extended COFF

F

FADT

См. Fixed ACPI Description Table.

FAT

См. File Allocation Table.

FAT16

См. File Allocation Table (16-bit).

FTP

См. File Transfer Protocol.

File Allocation Table

File Allocation Table (16-bit)

File Transfer Protocol

Член семейства высокоуровневых протоколов, реализованных поверх TCP, который может использоваться для передачи файлов через сеть TCP/IP.

Fixed ACPI Description Table

G

GUI

См. Graphical User Interface.

Giant

Имя механизма взаимного исключения (спящий `mutex`), который защищает большой набор ресурсов ядра. Хотя простой механизм блокировки был достаточен во времена, когда машина могла иметь всего несколько десятков процессов, одну сетевую карту и, конечно, только один процессор, в настоящее время это неприемлемое узкое место производительности. Разработчики FreeBSD активно работают над его заменой на блокировки, защищающие отдельные ресурсы, что позволит достичь значительно большей степени параллелизма как для однопроцессорных, так и для многопроцессорных машин.

Graphical User Interface

Система, в которой пользователь и компьютер взаимодействуют с помощью графики.

H

HTML

См. HyperText Markup Language.

HUP

См. HangUp.

HangUp

HyperText Markup Language

Язык разметки, используемый для создания веб-страниц.

I

I/O

См. Input/Output.

IASL

См. Intel's ASL compiler.

IMAP

См. Internet Message Access Protocol.

IP

См. Internet Protocol.

IPFW

См. IP Firewall.

IPP

См. Internet Printing Protocol.

IPv4

См. IP Version 4.

IPv6

См. IP Version 6.

ISP

См. Internet Service Provider.

IP Firewall

IP Version 4

Протокол IP версии 4, использующий 32-битную адресацию. Эта версия до сих пор наиболее широко используется, но постепенно заменяется на IPv6.

См. Also IP Version 6.

IP Version 6

Новый IP-протокол. Создан из-за того, что адресное пространство в IPv4 заканчивается. Использует 128 бит для адресации.

Input/Output

Intel's ASL compiler

Компилятор Intel для преобразования ASL в AML.

Internet Message Access Protocol

Протокол для доступа к электронным письмам на почтовом сервере, характеризующийся тем, что сообщения обычно хранятся на сервере, а не загружаются в почтовый клиент.

Смотрите также: Post Office Protocol версии 3.

Internet Printing Protocol

Internet Protocol

Протокол передачи пакетов, который является основным протоколом в Интернете. Первоначально разработан в Министерстве обороны США и является чрезвычайно важной частью стека TCP/IP. Без Интернет-протокола Интернет не стал бы тем, чем он

является сегодня. Для получения дополнительной информации см. RFC 791.

Internet Service Provider

Компания, предоставляющая доступ к Интернету.

K

KAME

Японское слово, означающее "черепаха", термин KAME используется в компьютерных кругах для обозначения проекта KAME Project, который занимается реализацией IPv6.

KDC

См. Key Distribution Center.

KLD

См. Kernel ld(1).

KMS

См. Kernel Mode Setting.

KSE

См. Kernel Scheduler Entities.

KVA

См. Kernel Virtual Address.

Kbps

См. Kilo Bits Per Second.

Установка режима дисплея в пространстве ядра.

Kernel ld(1)

Метод динамической загрузки функциональности в ядро FreeBSD без перезагрузки системы.

Kernel Scheduler Entities

Поддерживаемая ядром система потоков. Дополнительные сведения см. на домашней странице проекта KSE.

Kernel Virtual Address

Key Distribution Center

Kilo Bits Per Second

Используется для измерения пропускной способности (сколько данных может пройти через заданную точку за указанное время). Альтернативы приставке "Кило" включают

"Мега", "Гига", "Тера" и так далее.

L

LAN

См. Local Area Network.

LOR

См. Lock Order Reversal.

LPD

См. Line Printer Daemon.

Line Printer Daemon

Local Area Network

Сеть, используемая в локальной зоне, например, в офисе, дома или подобных местах.

Lock Order Reversal

Ядро FreeBSD использует ряд блокировок ресурсов для разрешения конкуренции за эти ресурсы. Диагностическая система блокировок, работающая в режиме реального времени и присутствующая в ядрах FreeBSD-CURRENT (но удаляемая для релизов), называемая witness(4), обнаруживает возможность взаимоблокировок из-за ошибок блокировок. (witness(4) на самом деле немного консервативна, поэтому возможны ложные срабатывания.) Истинное срабатывание означает, что «если вам не повезло, здесь могла бы произойти взаимоблокировка».

Истинные положительные LOR обычно быстро исправляются, поэтому перед отправкой в списки рассылки проверьте https://lists.FreeBSD.org/subscription/freebsd-current и страницу LOR увиденные.

M

MAC

См. Mandatory Access Control.

MADT

См. Multiple APIC Description Table.

MFC

См. Merge From Current.

MFH

См. Merge From Head.

MFS

См. Merge From Stable.

MFV

См. Merge From Vendor.

MIT

См. Massachusetts Institute of Technology.

MLS

См. Multi-Level Security.

MOTD

См. Message Of The Day.

MTA

См. Mail Transfer Agent.

MUA

См. Mail User Agent.

Mail Transfer Agent

Приложение, используемое для передачи электронной почты. Традиционно MTA входил в базовую систему BSD. В настоящее время Sendmail включен в базовую систему, но существует множество других MTA, таких как postfix, qmail и Exim.

Mail User Agent

Приложение, используемое пользователями для отображения и написания электронной почты.

Принудительный контроль доступа (MAC)

Massachusetts Institute of Technology

Merge From Current

Для объединения функциональности или патча из ветки -CURRENT в другую, чаще всего -STABLE.

Merge From Head

Объединить функциональность или патч из репозитория HEAD с более ранней веткой.

Merge From Stable

В обычном ходе разработки FreeBSD изменения сначала вносятся в ветку -CURRENT для тестирования, а затем переносятся в -STABLE. В редких случаях изменения сначала попадают в -STABLE, а затем переносятся в -CURRENT.

Этот термин также используется, когда исправление переносится из ветки -STABLE в ветку безопасности.

См. Also Merge From Current.

Merge From Vendor

Message Of The Day

Сообщение, обычно отображаемое при входе в систему, часто используемое для распространения информации среди пользователей системы.

Multi-Level Security

Multiple APIC Description Table

N

NAT

См. Network Address Translation.

NDISulator

См. Project Evil.

NFS

См. Network File System.

NTFS

См. New Technology File System.

NTP

См. Network Time Protocol.

Network Address Translation

Метод, при котором IP-пакеты переписываются при прохождении через шлюз, позволяя многим машинам за шлюзом эффективно использовать один IP-адрес.

Network File System

New Technology File System

Файловая система, разработанная Microsoft и доступная в её операционных системах «New Technology», таких как Windows® 2000, Windows NT® и Windows® XP.

Network Time Protocol

Средство синхронизации часов по сети.

O

OBE

См. Overtaken By Events.

ODMR

См. On-Demand Mail Relay.

OS

См. Operating System.

On-Demand Mail Relay

Operating System

Набор программ, библиотек и инструментов, предоставляющих доступ к аппаратным ресурсам компьютера. Современные операционные системы варьируются от простейших конструкций, поддерживающих только одну программу, работающую в данный момент времени и имеющую доступ только к одному устройству, до полностью многопользовательских, многозадачных и многопроцессных систем, способных обслуживать тысячи пользователей одновременно, каждый из которых запускает десятки различных приложений.

Overtaken By Events

Указывает на предложенное изменение (например, отчёт о проблеме или запрос функции), которое больше не актуально или не применимо из-за таких факторов, как последующие изменения в FreeBSD, изменения в сетевых стандартах, устаревание затронутого оборудования и т. д.

P

PAE

См. Physical Address Extensions.

PAM

См. Pluggable Authentication Modules.

PAP

См. Password Authentication Protocol.

PC

См. Personal Computer.

PCNSFD

См. Personal Computer Network File System Daemon.

PDF

См. Portable Document Format.

PID

См. Process ID.

POLA

См. Principle Of Least Astonishment.

POP

См. Post Office Protocol.

POP3

См. Post Office Protocol Version 3.

PPD

См. PostScript Printer Description.

PPP

См. Point-to-Point Protocol.

PPPoA

См. PPP over ATM.

PPPoE

См. PPP over Ethernet.

PPP через ATM

PPP через Ethernet

PR

См. Problem Report.

PXE

См. Preboot eXecution Environment.

Password Authentication Protocol

Personal Computer

Personal Computer Network File System Daemon

Physical Address Extensions

Метод обеспечения доступа к объёму оперативной памяти до 64 ГБ на системах с физически 32-разрядным адресным пространством (что без PAE ограничило бы объём 4 ГБ).

Подключаемые модули аутентификации (PAM)

Point-to-Point Protocol

Pointy Hat

Мифический головной убор, напоминающий колпак для двоечников, который вручается любому коммиттеру FreeBSD, сломавшему сборку, заставившему номера ревизий идти в обратном порядке или создавшему любой другой хаос в исходной базе. Любой коммиттер, стоящий своего звания, быстро накопит большую коллекцию таких. Использование (почти всегда?) шутливое.

Portable Document Format

Post Office Protocol

Смотрите также: Post Office Protocol версии 3.

Post Office Protocol Version 3

Протокол для доступа к электронным письмам на почтовом сервере, характеризующийся тем, что сообщения обычно загружаются с сервера на клиент, в отличие от хранения на сервере.

См. Also Internet Message Access Protocol.

PostScript Printer Description

Preboot eXecution Environment

Principle Of Least Astonishment

По мере развития FreeBSD изменения, видимые пользователю, должны оставаться как можно более предсказуемыми. Например, произвольное изменение порядка переменных запуска системы в /etc/defaults/rc.conf нарушает POLA. Разработчики учитывают POLA при внесении изменений в систему, которые видны пользователю.

PRIME

Метод совместного использования нескольких физических графических сопроцессоров путём разделения их буферов прямого доступа к памяти (DMA).

Problem Report

Описание проблемы, обнаруженной в исходном коде FreeBSD или документации. См. Написание отчётов о проблемах FreeBSD.

Process ID

Число, уникальное для конкретного процесса в системе, которое идентифицирует его и позволяет выполнять действия над ним.

Project Evil

Рабочее название NDISulator, написанного Биллом Полом, который назвал его, ссылаясь на то, насколько ужасна сама необходимость в подобном решении (с философской точки зрения). NDISulator — это специальный модуль совместимости, позволяющий использовать мини-портовые сетевые драйверы Microsoft Windows™ NDIS с FreeBSD/i386. Обычно это единственный способ использовать карты, драйверы которых закрыты. См. src/sys/compat/ndis/subr_ndis.c.

R

RA

См. Router Advertisement.

RAID

См. Redundant Array of Inexpensive Disks.

RAM

См. Random Access Memory.

RD

См. Received Data.

RFC

См. Request For Comments.

RISC

См. Reduced Instruction Set Computer.

RPC

См. Remote Procedure Call.

RS232C

См. Recommended Standard 232C.

RTS

См. Request To Send.

Random Access Memory

Revision Control System

Система контроля версий (RCS) — один из старейших наборов программ, реализующих «контроль версий» для обычных файлов. Она позволяет хранить, извлекать, архивировать, вести журнал изменений, идентифицировать и объединять несколько версий для каждого файла. RCS состоит из множества небольших инструментов, работающих вместе. В ней отсутствуют некоторые функции, присутствующие в более современных системах контроля версий, таких как Git, но она очень проста в установке, настройке и начале использования для небольшого набора файлов.

См. Also Subversion.

Received Data

Контакт или провод кабеля RS232C, на который принимаются данные.

См. Also Transmitted Data.

Recommended Standard 232C

Стандарт для связи между последовательными устройствами.

Reduced Instruction Set Computer

Подход к проектированию процессоров, при котором операции, которые может выполнять аппаратное обеспечение, упрощаются, но делаются максимально универсальными. Это может привести к снижению энергопотребления, уменьшению количества транзисторов и в некоторых случаях к повышению производительности и увеличению плотности кода. Примерами RISC-процессоров являются Alpha, SPARC®, ARM® и PowerPC®.

Redundant Array of Inexpensive Disks

Remote Procedure Call

Request For Comments

Набор документов, определяющих стандарты Интернета, протоколы и так далее. См. www.rfc-editor.org.

Также используется как общий термин, когда у кого-то есть предложение по изменению и он хочет получить обратную связь.

Request To Send

Сигнал RS232C, запрашивающий начало передачи данных от удалённой системы.

См. Also Clear To Send.

Router Advertisement

S

SCI

См. System Control Interrupt.

SCSI

См. Small Computer System Interface.

SG

См. Signal Ground.

SLAAC

См. StateLess Address AutoConfiguration.

SMB

См. Server Message Block.

SMP

См. Symmetric MultiProcessor.

SMTP

См. Simple Mail Transfer Protocol.

SMTP AUTH

См. SMTP Authentication.

SSH

См. Secure Shell.

STR

См. Suspend To RAM.

SVN

См. Subversion.

StateLess Address AutoConfiguration

SMTP Authentication

Server Message Block

Сигнальная земля (Signal Ground)

Вывод RS232 или провод, который является опорным заземлением для сигнала.

Simple Mail Transfer Protocol

Secure Shell (SSH)

Small Computer System Interface

Subversion

Subversion — это система контроля версий, в настоящее время используемая проектом FreeBSD.

Suspend To RAM

Symmetric MultiProcessor

System Control Interrupt

T

TCP

См. Transmission Control Protocol.

TCP/IP

См. Transmission Control Protocol/Internet Protocol.

TD

См. Transmitted Data.

TFTP

См. Trivial FTP.

TGT

См. Ticket-Granting Ticket.

TSC

См. Time Stamp Counter.

Ticket-Granting Ticket

Time Stamp Counter

Счетчик профилирования, внутренний для современных процессоров Pentium®, который подсчитывает тактовые импульсы частоты ядра.

Transmission Control Protocol

Протокол, который работает поверх (например) IP-протокола и гарантирует надёжную и упорядоченную доставку пакетов.

Transmission Control Protocol/Internet Protocol

Термин, обозначающий комбинацию протокола TCP, работающего поверх протокола IP. Большая часть Интернета работает на TCP/IP.

Transmitted Data

Контакт или провод RS232C, по которому передаются данные.

См. Also Received Data.

Trivial FTP

U

UDP

См. User Datagram Protocol.

UFS1

См. Unix File System Version 1.

UFS2

См. Unix File System Version 2.

UID

См. User ID.

URL

См. Uniform Resource Locator.

USB

См. Universal Serial Bus.

Uniform Resource Locator

Метод определения местоположения ресурса, такого как документ в Интернете, и способ идентификации этого ресурса.

Unix File System Version 1

Исходная файловая система UNIX®, иногда называемая Berkeley Fast File System.

Unix File System Version 2

Расширение UFS1, представленное в FreeBSD 5-CURRENT. UFS2 добавляет 64-битные указатели блоков (преодолевая барьер в 1T), поддержку расширенного хранения файлов и другие возможности.

Universal Serial Bus

Аппаратный стандарт, используемый для подключения широкого спектра компьютерных периферийных устройств к универсальному интерфейсу.

User ID

Уникальный номер, присваиваемый каждому пользователю компьютера, по которому можно идентифицировать ресурсы и разрешения, назначенные этому пользователю.

User Datagram Protocol

Простой, ненадёжный протокол датаграмм, используемый для обмена данными в сети TCP/IP. В отличие от TCP, UDP не обеспечивает проверку и исправление ошибок.

V

VPN

См. Virtual Private Network.

Virtual Private Network

Способ использования общедоступной телекоммуникационной сети, такой как Интернет, для обеспечения удалённого доступа к локальной сети, например, корпоративной LAN.

Сведения об издании

Эта книга — результат совместной работы сотен участников «The FreeBSD Documentation Project». Текст написан в формате AsciiDoc.