FreeBSD ハンドブック

概要

FreeBSD へようこそ! このハンドブックは *FreeBSD 13.1-RELEASE* および *FreeBSD 12.4-RELEASE* のインストールと日常での使い方について記述したものです。
本ハンドブックはさまざまな人々による編集の成果で、現在も改編作業中です。
いま存在するセクションの中には情報が古くなってしまっているものがあります。
もし、この文書を新しくしたり、 新しい情報の追加に協力したいとお考えなら、 FreeBSD documentation project メーリングリスト まで電子メールを (英語で) 送ってください。

このハンドブックの最新バージョンは FreeBSD ウェブサイト から入手できます。 以前のバージョンは https://docs.FreeBSD.org/doc/ から入手できます。 この文書の他の文書形式や圧縮形式は FreeBSD ダウンロードサーバ や ミラーサイト からダウンロードできます。
ハンドブックやその他の文書は、検索ページ で検索できます。

目次

前書き . . . 8
想定している読者 . . . 8
第 3 版からの変更 . . . 8
第 2 版 (2004) からの変更 . . . 8
第 1 版 (2001) からの変更 . . . 9
この文書の構成 . . . 10
この文書で用いられている表記法 . . . 12
謝辞 . . . 13
I: 導入 . . . 14
1. はじめに . . . 15
1.1. この章では . . . 15
1.2. FreeBSD へようこそ! . . . 15
1.3. FreeBSD プロジェクトについて . . . 20
2. FreeBSD のインストール . . . 26
2.1. この章では . . . 26
2.2. 最小ハードウェア要件 . . . 26
2.3. インストール前に行う作業 . . . 27
2.4. インストールの開始 . . . 31
2.5. bsdinstall の使用 . . . 34
2.6. ディスク領域の割り当て . . . 39
2.7. 配布ファイルのダウンロード . . . 56
2.8. ネットワークインターフェース、アカウント、タイムゾーン、サービスおよびセキュリティオプションの設定 . . . 58
2.9. トラブルシューティング . . . 81
2.10. Live CD を使う . . . 82
3. FreeBSD の基礎知識 . . . 83
3.1. この章では . . . 83
3.2. 仮想コンソールと端末 . . . 83
3.3. ユーザと基本的なアカウント管理 . . . 86
3.4. 許可属性 . . . 95
3.5. ディレクトリ構造 . . . 100
3.6. ディスク構成 . . . 102
3.7. ファイルシステムのマウントとアンマウント . . . 109
3.8. プロセスおよびデーモン . . . 112
3.9. シェル . . . 114
3.10. テキストエディタ . . . 118
3.11. デバイスとデバイスノード . . . 118
3.12. マニュアルページ . . . 118
4. アプリケーションのインストール - packages と ports . . . 121

4.1. この章では . . . 121
4.2. ソフトウェアのインストール . . . 121
4.3. ソフトウェアの探し方 . . . 123
4.4. pkg によるバイナリ package の管理 . . . 125
4.5. Ports Collection の利用 . . . 132
4.6. Poudriere を用いた package の構築 . . . 142
4.7. インストール後の作業 . . . 145
4.8. うまく動作しない ports に遭遇した場合には . . . 145
5. X Window System . . . 147
5.1. この章では . . . 147
5.2. 用語の説明 . . . 147
5.3. Xorg のインストール . . . 149
5.4. Xorg の設定 . . . 149
5.5. Xorg でのフォントの使用 . . . 159
5.6. X ディスプレイマネージャ . . . 164
5.7. デスクトップ環境 . . . 166
5.8. Compiz Fusion のインストール . . . 169
5.9. トラブルシューティング . . . 173
II: 日々の生活 . . . 178
6. デスクトップアプリケーション . . . 179
6.1. この章では . . . 179
6.2. ブラウザ . . . 179
6.3. 生産的なアプリケーション . . . 181
6.4. ドキュメントビューア . . . 185
6.5. 財務管理ソフトウェア . . . 187
7. マルチメディア . . . 189
7.1. この章では . . . 189
7.2. サウンドカードの設定 . . . 189
7.3. MP3 オーディオ . . . 195
7.4. ビデオ再生 . . . 197
7.5. TV カードの設定 . . . 204
7.6. MythTV . . . 205
7.7. 画像スキャナ . . . 206
8. FreeBSD カーネルのコンフィグレーション . . . 212
8.1. この章では . . . 212
8.2. なぜカスタムカーネルを作るか? . . . 212
8.3. システムのハードウェアについて知る . . . 213
8.4. コンフィグレーションファイル . . . 214
8.5. カスタムカーネルの構築とインストール . . . 216
8.6. 問題が起きた場合には . . . 217
9. プリンタの利用 . . . 219

9.1. この章では . . . 219
9.2. はじめに . . . 219
9.3. 基本的な設定 . . . 220
9.4. プリンタ設定上級編 . . . 235
9.5. プリンタを使う . . . 269
9.6. 標準スプーラの代替品 . . . 278
9.7. トラブルシューティング . . . 279
10. Linux® バイナリ互換機能 . . . 284
10.1. この章では . . . 284
10.2. Linux® バイナリ互換機能の設定 . . . 284
10.3. 高度なトピックス . . . 287
III: システム管理 . . . 290
11. 設定とチューニング . . . 291
11.1. この章では . . . 291
11.2. 中核となる設定 . . . 291
11.3. アプリケーションの設定 . . . 292
11.4. サービスの起動 . . . 293
11.5. バーチャルホスト . . . 294
11.6. 設定ファイル . . . 295
11.7. sysctl によるチューニング . . . 299
11.8. ディスクのチューニング . . . 299
11.9. Kernel 制限のチューニング . . . 301
12. FreeBSD の起動のプロセス . . . 302
12.1. この章では . . . 302
12.2. FreeBSD の起動プロセス . . . 302
12.3. Device Hints . . . 309
12.4. シャットダウン動作 . . . 310
13. セキュリティ . . . 311
13.1. この章では . . . 311
13.2. はじめに . . . 311
13.3. ワンタイムパスワード . . . 318
13.4. TCP Wrappers . . . 322
13.5. Kerberos5 . . . 324
13.6. OpenSSL . . . 333
13.7. VPN over IPsec . . . 336
13.8. OpenSSH . . . 342
13.9. ファイルシステムアクセス制御リスト (ACL) . . . 348
13.10. サードパーティ製ソフトウェアのセキュリティ問題を監視する . . . 350
13.11. FreeBSD セキュリティ勧告 . . . 351
13.12. プロセスアカウンティング . . . 356
13.13. リソースの制限 . . . 357

14. ストレージ . . . 361
14.1. この章では . . . 361
14.2. デバイス名 . . . 361
14.3. ディスクの追加 . . . 362
14.4. RAID . . . 364
14.5. 光メディア (CD & DVD) の作成と使用 . . . 370
14.6. フロッピーディスクの作成と使用 . . . 377
14.7. データテープの作成と使用 . . . 379
14.8. フロッピーディスクへのバックアップ . . . 382
14.9. バックアップの基本 . . . 383
14.10. ネットワーク、メモリ、そしてファイルベースのファイルシステム . . . 391
14.11. ファイルシステムのスナップショット . . . 396
14.12. ファイルシステムクォータ . . . 397
14.13. ディスクパーティションの暗号化 . . . 401
15. 地域化 (localization) - i18n/L10n の利用と設定 . . . 406
15.1. この章では . . . 406
15.2. 地域化の利用 . . . 406
15.3. 国際化 (i18n) に対応したアプリケーションを見つける . . . 413
15.4. 特定の言語にロケールを設定する . . . 413
16. FreeBSD のアップデートとアップグレード . . . 416
16.1. この章では . . . 416
16.2. FreeBSD Update . . . 416
16.3. ブートコードのアップデート . . . 424
16.4. ドキュメントのアップデート . . . 424
16.5. 開発ブランチを追いかける . . . 425
16.6. ソースを用いた FreeBSD のアップデート . . . 429
16.7. 複数のマシンで追いかける . . . 437
IV: ネットワーク通信 . . . 439
17. シリアル通信 . . . 440
17.1. この章では . . . 440
17.2. はじめに . . . 440
17.3. シリアル端末 . . . 445
17.4. ダイアルインサービス . . . 450
17.5. ダイアルアウトサービス . . . 458
17.6. シリアルコンソールの設定 . . . 462
18. PPP と SLIP . . . 472
18.1. この章では . . . 472
18.2. ユーザ ppp の利用 . . . 472
18.3. カーネル PPP の利用 . . . 487
18.4. PPP オーバイーサネット (PPPoE) の利用 . . . 495
18.5. SLIP の利用 . . . 497

19. 電子メール . . . 509
19.1. この章では . . . 509
19.2. 電子メールを使う . . . 509
19.3. sendmail の設定 . . . 510
19.4. MTA の変更 . . . 513
19.5. トラブルシュート . . . 516
19.6. 先進的なトピックス . . . 520
19.7. UUCP とともに SMTP を使う . . . 522
19.8. ダイアルアップ接続でメールを使う . . . 524
19.9. SMTP 認証 . . . 526
20. 高度なネットワーク . . . 528
20.1. この章では . . . 528
20.2. ゲートウェイと経路 . . . 528
20.3. 無線ネットワーク . . . 535
20.4. Bluetooth . . . 542
20.5. ブリッジ . . . 551
20.6. NFS . . . 554
20.7. ディスクレス稼働 . . . 561
20.8. ISDN . . . 567
20.9. NIS/YP . . . 571
20.10. DHCP . . . 589
20.11. DNS . . . 593
20.12. NTP . . . 606
20.13. ネットワークアドレス変換 (NAT) . . . 609
20.14. inetd"スーパサーバ" . . . 613
20.15. パラレルライン IP (PLIP) . . . 617
20.16. IPv6 . . . 620
V: 付録 . . . 625
付録 A: FreeBSD の入手方法 . . . 626
A.1. Mirrors . . . 626
A.2. Git の利用 . . . 628
A.3. Subversion の利用 . . . 631
A.4. CD および DVD セット . . . 634
付録 B: 参考図書 . . . 635
B.1. FreeBSD 専門の書籍 . . . 635
B.2. 利用者向けのガイド . . . 636
B.3. 管理者向けのガイド . . . 636
B.4. プログラマ向けのガイド . . . 636
B.5. オペレーティングシステム内部 . . . 637
B.6. セキュリティの参考資料 . . . 638
B.7. ハードウェアの参考資料 . . . 638

B.8. UNIX® の歴史 . . . 639
B.9. 定期刊行物、雑誌およびジャーナル . . . 639
付録 C: インターネット上のリソース . . . 640
C.1. ウェブサイト . . . 640
C.2. メーリングリスト . . . 640
C.3. Usenet ニュースグループ . . . 662
C.4. オフィシャルミラー . . . 662
付録 D: PGP 公開鍵 . . . 665
D.1. オフィサ . . . 665

前書き

想定している読者

最初の部分は FreeBSD を使い始めた人向けで、FreeBSD のインストールの過程を手引きし、UNIX® の基礎となっている概念や慣習を丁寧に紹介します。
この部分に取り組むために必要なのは、探究心と、紹介された新たな概念を理解する能力だけです。

その次の、ハンドブックのはるかに大きな部分では、FreeBSD
システム管理者が興味を抱くあらゆる種類の話題が分かりやすく言及されています。
一部の章は、その章の前に読んでおくべきことが推奨されており、各章の始めの概要で述べられています。

さらなる情報源の一覧は、参考図書 をご覧ください。

第 3 版からの変更

オンライン版のハンドブックは、FreeBSD ドキュメンテーションプロジェクトの献身的なメンバーによる 10 年以上に渡る作業の頂点に立つものです。 2004 年に出版された 2 巻組の第 3 版からの主な変更は、次のようなものです。

- Windows® アプリケーションを FreeBSD 上で実行するための WINE に関する情報が追加されました。
- 強力なパフォーマンス解析ツール DTrace に関する情報が追加されました。
- ファイルシステム対応の章が追加されました。 Sun™ の ZFS のような FreeBSD ネイティブではないファイルシステムへの対応について説明しています。
- セキュリティ監査の章が追加されました。 FreeBSD における新しい監査のケイパビリティおよびその使用方法について説明しています。
- 仮想化の章が追加されました。os; を仮想化ソフトへインストールする方法などを取り上げています。
- 新しいインストールユーティリティの bsdinstall を用いた FreeBSD のインストール方法を説明する FreeBSD のインストール という章が追加されました。

第 2 版 (2004) からの変更

第 3 版は、FreeBSD ドキュメンテーションプロジェクトの献身的なメンバーによる 2 年以上に渡る作業の頂点に立つものです。 サイズが大きくなったため、印刷版は、2 巻での出版となりました。 この新たな版における主な変更は、次のようなものです。

- 設定とチューニング に、ACPI 電源管理、`cron` システムユーティリティ、およびカーネルチューニングオプションに関するより多くの情報が追加されました。
- セキュリティ に、 Virtual Private Network (VPN)、 ファイルシステムアクセスコントロールリスト (ACL)、 およびセキュリティ勧告に関する情報が追加されました。
- Mandatory Access Control (MAC) の章がこの版で追加されました。 MAC がどのようなもので、このメカニズムがどのように FreeBSD システムを安全にするかについて説明しています。
- ストレージ に、 USB ストレージデバイス、ファイルシステムスナップショット、ファイルシステムクォータ、 ファイルおよびネットワークベースのファイルシステム、

暗号化されたディスクパーティションに関する情報が追加されました。

- PPP と SLIP に、トラブルシューティングの節が追加されました。
- 電子メール, に、他のメール転送エージェント、SMTP 認証、UUCP, fetchmail, procmail や他の高度な話題についての情報が追加されました。
- ネットワークサービスの章が、この版で新しく追加されました。この章では、Apache HTTP サーバ、fptd および Samba を用いて Microsoft® Windows® クライアント用にサーバを設定する方法などを取り上げています。
  再構成によりいくつかの節が、高度なネットワーク から移動してきました。
- 高度なネットワーク に、FreeBSD での Bluetooth® デバイスの使用、ワイヤレスネットワークの設定、Asynchronous Transfer Mode (ATM) ネットワークに関する情報が追加されました。
- 本書で使われている専門用語の定義をまとめた用語集が追加されました。
- 本書を通じて表および図の表現において数多くの改良がおこなわれました。

第 1 版 (2001) からの変更

第 2 版は、FreeBSD ドキュメンテーションプロジェクトの献身的なメンバーによる 2 年以上に渡る作業の頂点に立つものでした。この新たな版における主な変更は、次のようなものでした。

- 完備した索引が追加されました。
- ASCII キャラクタによる図はすべて画像に置き換えられました (訳注: 日本語版は作業中です)。
- 各章に、章に記載されている内容と、読者に期待される予備知識がすぐに分かるように、一定の内容の概要が付け加えられました。
- 内容は、"始めに"、"システム管理"、"付録" の 3 つの論理的な部分に再構成されました。
- UNIX の基礎知識 には、プロセス、デーモン、シグナルに関する情報が追加されました。
- アプリケーションのインストール - packages と ports には、バイナリパッケージの管理に関する情報が追加されました。
- X Window System は、XFree86™ 4.X 上で KDE や GNOME のような近代的なデスクトップテクノロジーを利用することに力点をおいて、完全に書き直されました。
- FreeBSD の起動のプロセス が拡張されました。
- ストレージ は、"ディスク" と "バックアップ" の 2 つの章に分かれていたものをまとめて書き直されました。私たちは、この話題は 1 つの章にまとめて示した方が分かりやすいと感じています。RAID (ハードウェアとソフトウェアの両方) に関する節も追加されました。
- シリアル通信 は FreeBSD 4.X/5.X 向けに一から再構成されました。
- PPP と SLIP は大幅に更新されました。
- 高度なネットワーク に、多くの新しい節が追加されました。
- 電子メール に、sendmail の設定についてより多くの情報が加えられました。
- Linux® バイナリ互換機能 には、Oracle® や Mathematica® のインストール情報が加えられました (訳注: 日本語版は作業中です)。

- この第 2 版では、以下の新たな話題が扱われています。
  - 設定とチューニング.
  - マルチメディア.

この文書の構成

この文書は 5 部構成になっています。 第 1 部 導入 では、FreeBSD のインストールと基本的な使い方を扱います。
各章は順に読むことを想定していますが、馴染み深い話題を扱った章は飛ばしてもよいでしょう。 第 2 部 日々の生活 では、FreeBSD で良く使われる機能について説明します。 この章とそれに続く章は、順不同に読むことができます。
各章の始めにはその章が何を扱っていて、読者にどんな予備知識が期待されるかを簡潔に述べた概要がおかれています。 第 3 部 システム管理 は、システム管理に関する話題を扱っています。 第 4 部 ネットワーク通信 では、ネットワークおよびサーバに関する話題を扱っています。 第 5 部は参考情報からなる 付録 です。

はじめに

新規ユーザに FreeBSD を紹介します。ここでは、FreeBSD プロジェクトの歴史、目標と開発モデルについて述べています。

FreeBSD のインストール

bsdinstall を用いた FreeBSD 9.*x* 以降のシステムのインストール過程を一通りユーザに案内しています。

UNIX の基礎知識

FreeBSD オペレーティングシステムの基本的なコマンドや機能を扱っています。 Linux® やその他の UNIX® 風のものに馴染んでいたら、 この章を飛ばしても構わないでしょう。

アプリケーションのインストール - *packages* と *ports*

FreeBSD の革新的な "Ports Collection" および標準的なバイナリパッケージによるサードパーティアプリケーションのインストールについて説明しています。

X Window System

X Window System 全般と、特に FreeBSD 上での X11 の利用について述べています。 また、KDE や GNOME のような一般的なデスクトップ環境にも触れています。

デスクトップアプリケーション

Web ブラウザや生産性向上ツールのような一般的なデスクトップアプリケーションをいくつか挙げ、FreeBSD におけるインストール方法を説明しています。

マルチメディア

システムを音声やビデオ再生に対応させるためにどう設定するかを説明します。
また、音声やビデオアプリケーションも例示しています。

FreeBSD カーネルのコンフィグレーション

どのような場合に新たにカーネルを構成する必要があるかを説明し、

カスタムカーネルのコンフィグレーション、構築、 インストールについて詳しく説明しています。

プリンタの利用

FreeBSD におけるプリンタの取り扱いを説明しています。たとえば、バナーページ、プリンターの課金、初期設定といったことです。

Linux® バイナリ互換機能

FreeBSD の Linux® バイナリ互換機能を説明しています。また、 Oracle®, Mathematica® といった人気の高い Linux® アプリケーションのインストールを詳しく説明しています。

設定とチューニング

システム管理者が FreeBSD システムを調整して最適な性能を引き出すのに利用できるパラメータについて述べています。
また、FreeBSD で利用されている様な設定ファイルとそのありかも解説しています。

FreeBSD の起動のプロセス

FreeBSD の起動プロセスを解説し、このプロセスを設定オプションで制御する方法を説明しています。

セキュリティ

FreeBSD システムを安全に保つために役立つ Kerberos, IPsec および OpenSSH といった利用可能なさまざまなツールについて説明しています。

ストレージ

FreeBSD でストレージメディアやファイルシステムをどう扱うかを説明しています。
対象は、物理ディスク、RAID アレイ、 光学およびテープメディア、メモリベースのディスク、ネットワークファイルシステムなどです。

地域化 *(localization) - i18n/L10n* の利用と設定

FreeBSD を英語以外の言語で使う方法を説明しています。
システムとアプリケーション両方のレベルの地域化を扱っています。

FreeBSD のアップデートとアップグレード

FreeBSD-STABLE, FreeBSD-CURRENT と FreeBSD のリリースの違いを説明します。
どんなユーザにとって開発システムを追随するのが有用かを述べ、
その方法の概要をまとめています。
システムを最新のセキュリティリリースへアップデートする方法についても説明しています。

シリアル通信

FreeBSD システムに端末やモデムを、ダイヤルインまたはダイヤルアウト用に接続する方法を説明しています。

PPP と *SLIP*

FreeBSD で、PPP を使ってリモートシステムに接続する方法を説明しています。

電子メール

電子メールサーバの構成要素をそれぞれ説明し、
最もよく使われているメールサーバソフトウェアである sendmail について、

単純な設定をとりあげています。

高度なネットワーク

LAN 上の他のコンピュータとインターネット接続の共有、高度なルーティングに関するトピックス、ワイヤレスネットワーク、 Bluetooth®, ATM, IPv6 等々、ネットワークに関するさまざまな話題を取り扱っています。

FreeBSD の入手方法

FreeBSD を収録した CDROM や DVD の様々な入手先や、FreeBSD をダウンロードしてインストールできるインターネット上のサイトを挙げています。

参考図書

この文書は、 もっと詳しい説明が欲しくなるかもしれないさまざまな題目について触れています。参考図書には、このハンドブックで参照している、 多くの素晴らしい本が挙げられています。

インターネット上のリソース

FreeBSD ユーザが FreeBSD について質問したり、 技術的な議論に参加できる、多くの公開された場について説明しています。

PGP 公開鍵

多くの FreeBSD 開発者の PGP fingerprint を載せています。

この文書で用いられている表記法

一貫して読みやすい文章を提供するために、 この文書全体では以下の表記法が用いられています。

書体による表記

イタリック体

イタリック体 のフォントは、ファイル名、URL, 強調表現、技術用語の最初の使用を表すのに使われています。

等幅

等幅フォントは、エラーメッセージ、 コマンド、環境変数、ports の名称、ホスト名、ユーザ名、グループ名、デバイスの名称、変数、 コードの断片を表すのに使われています。

太字

太字のフォントは、アプリケーション、コマンド、キーを表すのに使われています。

ユーザー入力

文章の他の部分と区別するため、キーは太字で示されています。
同時に押すことを意図したキーの組み合わせは、キーの間に + を入れて表されます。 たとえば

`Ctrl` + `Alt` + `Del`

は、ユーザーが `Ctrl`, `Alt` それから `Del` キーを同時に押すことを意図しています。

順に押すことを意図したキーは、カンマで区切って表されます。 たとえば

Ctrl + X, Ctrl + S

は、ユーザーが Ctrl キーと X キーを同時に押してから、Ctrl キーと S キーを同時に押すことを意図しています。

例

C:\> で始まる例は、MS-DOS® コマンドを表しています。特に注釈がなければ、それらのコマンドは最近の Microsoft® Windows® の "コマンドプロンプト" 環境でも実行できます。

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

\# で始まる例は、FreeBSD 上でスーパーユーザ権限で実行しなければならないコマンドを示しています。そのコマンドを入力するには、root としてログインするか、通常のアカウントでログインして、スーパーユーザ権限を取得するために su(1) を使います。

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

% で始まる例は、通常のユーザアカウントで実行するべきコマンドを示しています。特に断りのない限り、環境変数の設定やその他のシェルコマンドには C シェルの文法が使われています。

```
% top
```

謝辞

あなたが手にしている文書は、世界中の何百人もの人々の努力の賜物です。
誤字脱字の修正を送ったのか、文章を丸々投稿したのかによらず、すべての貢献が役に立ちました。

多くの会社が、著者らを雇用してフルタイムでこの文書に取り掛かれるようにしたり、出版費用を出したりして、この文書を作り上げるのを援助してくれました。 特に、BSDi (その後 Wind River Systems に買収されました) は、フルタイムでこの文書の改善作業をするように FreeBSD ドキュメンテーションプロジェクトのメンバーを雇用し、それが 2000 年 3 月の最初の出版 (ISBN 1-57176-241-8) につながりました。 その後、Wind River Systems は、印刷出力の仕組みを整備し、章を追加するために著者を何名か追加で雇用してくれました。
この作業は、2001 年 11 月の第 2 版の出版 (ISBN 1-57176-303-1) に結実しました。 2003-2004 年には、ハンドブック第 3 版の出版準備のために FreeBSD Mall, Inc が貢献者を雇用してくれました。 第 3 版は 2 巻組となりました。 各巻は、The FreeBSD Handbook 3rd Edition Volume 1: User Guide (ISBN 1-57176-327-9) および The FreeBSD Handbook 3rd Edition Volume 2: Administrators Guide (ISBN 1-57176-328-7) として出版されています。

Part I: 導入

ハンドブックの第 1 部はユーザと FreeBSD が初めての管理者向けです。各章の内容は以下のとおりです。

- FreeBSD の紹介
- インストールの手順の解説
- UNIX® の基礎
- FreeBSD で利用できる豊富なサードパーティ製のアプリケーションのインストール方法
- UNIX® におけるウィンドウシステムの X、およびプロダクティブなデスクトップ環境の設定の詳細の紹介

頻繁にページを飛すことなく各章を前から後へとスムーズに読み進めるように、後方への参照を極力抑えるようにしています。

Chapter 1. はじめに

1.1. この章では

FreeBSD に興味を持っていただきありがとうございます! この章では FreeBSD の歴史、目標、開発モデルなど、 FreeBSD プロジェクトに関するさまざまな事柄を扱います。

この章に書かれている話題は、次のようなものです。

- FreeBSD とその他のオペレーティングシステムとの違い
- FreeBSD プロジェクトの歴史
- FreeBSD プロジェクトの目標
- FreeBSD オープンソース開発モデルの基本的な考え方
- そして、"FreeBSD" という名前の由来について

1.2. FreeBSD へようこそ!

FreeBSD は、標準に準拠した Unix-like なオープンソースのオペレーティングシステムで、 x86 (32 および 64 ビットの両方), ARM®, AArch64, RISC-V®, MIPS®, POWER®, PowerPC® および Sun UltraSPARC® コンピュータに対応しています。 FreeBSD は、プリエンプティブなマルチタスク、メモリ保護、仮想メモリ、マルチユーザシステム、SMP 対応、さまざまな言語やフレームワーク用のすべてのオープンソースの開発ツール、 X ウィンドウシステム、KDE や GNOME を中心としたデスクトップ機能といった、今日では標準となっている機能をすべて提供しています。 注目すべき機能は以下の通りです。

- 自由なオープンソースライセンス。 ソースコードを自由に変更し、配布することができます。
  潜在的なライセンスの互換性の問題を避け、
  コピーレフトライセンスに典型的な制限を課すことなく、
  オープンソースプロジェクトおよびクローズな製品の両方に組み込むことが可能です。
- 堅固な *TCP/IP* ネットワーク - FreeBSD は、
  かってないほどの性能とスケーラビリティを兼ね備えた業界標準プロトコルを実装しています。
  サーバおよびルータ/ファイアウォールルールの両方と相性が良く、
  実際に多くの会社やベンダがまさにこの目的で採用しています。
- 完全に統合された *OpenZFS* への対応。 これには root-on-ZFS, ZFS ブート環境、障害管理、委任管理、jails への対応、FreeBSD 固有の文書、
  そしてシステムのインストーラによる対応が含まれます。
- Capsicum
  ケーパビリティおよびサンドボックスメカニズムに対する強制アクセスコントロールフレームワークによる 拡張されたセキュリティ機能。
- 対応しているすべてのアーキテクチャで利用可能な *3* 万を超えるコンパイル済みの *packages*。
  そして、あなた自身のカスタマイズされたソフトウェアの構築を容易にする Ports Collection。
- ドキュメント - システム管理からカーネル内部にまで渡る内容に関する、
  さまざまな著者によるハンドブックやブックに加え、 man(1) ページが用意されています。
  ユーザ空間のデーモン、 ユーティリティおよびコンフィグレーションファイルだけではなく、

カーネルドライバの API (セクション 9) および個々のドライバ (セクション 4) も用意されています。

- 分かりやすく首尾一貫したリポジトリ構造とビルドシステム - FreeBSD は、カーネルおよびユーザ空間の両方について、
  すべての構成要素をひとつのリポジトリで管理しています。
  統一されカスタマイズが容易なビルドシステムおよび綿密に考えられた開発プロセスが、
  あなた自身の製品のビルドインフラストラクチャに FreeBSD を統合することを容易にします。
- *Unix* の哲学に忠実であり続けます。 ハードコードされたモノリシックな "オールインワン" デーモンより、 要素から構成することを好みます。
- Linux との バイナリ互換。 仮想化の必要なしに多くの Linux バイナリを実行できます。

FreeBSD はカリフォルニア大学バークレイ校の Computer Systems Research Group (CSRG) による 4.4BSD-Lite リリースを基にしており、 BSD システムの開発の優れた伝統を守り続けています。 CSRG による素晴らしい活動に加えて、 FreeBSD プロジェクトは何千時間もの時間を注ぎ込んで、実際の使用の場において最大の性能と信頼性を発揮するためにシステムのチューニングをおこなっています。 FreeBSD は、商用のオペレーティングシステムと同等の性能、信頼性を、他では実現されていない数多くの最新の機能と共に提供しています。

1.2.1. FreeBSD で何ができるの?

あなたの思いつく限りのアプリケーションは、何でも FreeBSD で実行できます。ソフトウェア開発からファクトリオートメーション、
在庫制御から遠く離れた人工衛星のアンテナの方向調整まで; 商用 UNIX® 製品でできることは、FreeBSD でも十分にできるのです! また、FreeBSD は世界中の研究センターや大学によって開発される文字通り何千もの高品質で、
たいていはほとんど無料で利用できるアプリケーションによる恩恵を得ることができます。

FreeBSD のソースコードは自由に提供されているので、システムも特別なアプリケーションやプロジェクトに合わせて、
いくらでもカスタマイズすることができます。
これは有名な商業ベンダから出ているほとんどのオペレーティング
システムでは不可能なことです。以下に現在 FreeBSD を
使っている人々のアプリケーションの例をいくつか上げます:

- インターネットサービス*:* FreeBSD に組み込まれている 頑強な TCP/IP ネットワーキング機能は次のようなさまざまな
  インターネットサービスの理想的なプラットフォームになります:
  - ウェブサーバ
  - IPv4 および IPv6 ルーティング
  - ファイアウォールと NAT ("IP マスカレード") ゲートウェイ
  - FTP サーバ
  - メールサーバ
  - さらにいろいろ...
- 教育*:* あなたは、計算機科学または関連分野の工学を専攻する学生さんですか? オペレーティングシステムやコンピュータアーキテクチャ、 ネットワークについて学習するなら、
  実際に FreeBSD のソースコードを読んで、

それがどのように動作するのかを学ぶのが一番よい方法です。 また、無料で利用できる CAD や数学、 グラフィックデザインのパッケージがいくつもあるので、コンピュータに関わる主要な目的が、 他 のことをすることにある方にも、 大いに役立ちます。

- 研究: システム全体のソースコードが利用できるため、 FreeBSD はオペレーティングシステムの研究だけでなく、計算機科学の他の部門においても優れたプラットフォームです。 自由に利用できる FreeBSD の特長は、オープンフォーラムで議論される特別なライセンスの同意や制限について心配することなく、離れたグループでもアイディアや開発の共有による共同研究を可能にします。
- ネットワーキング: 新しいルータが必要? ネームサーバ (DNS) は? 内部のネットワークを人々から守るファイアウォールは? FreeBSD はすみに眠っている使われていない PC を簡単に洗練されたパケットフィルタリング機能を持つ高級なルータに 変えることができます。
- 組み込み: FreeBSD は、 組み込みシステムを構築する優れたプラットフォームとなります。 ARM®, MIPS® および PowerPC® プラットフォームへのサポートとともに、強固なネットワークスタック、最新の機能および寛容なBSD ライセンス により、FreeBSD は、組み込みルータ、ファイアウォールおよび他のデバイスを構築する優れた基盤となります。
- デスクトップ: FreeBSD は、自由に利用できる X11 サーバを使うことによって、安価なデスクトップとなります。 FreeBSD では、標準的な GNOME および KDE グラフィカルユーザインタフェースを含む、数多くのオープンソースのデスクトップ環境を選択できます。 FreeBSD は、 中央のサーバから "ディスクレス"でもブート可能であり、 個々のワークステーションを安価で、容易に管理することさえ可能にします。
- ソフトウェア開発: 基本的な FreeBSD システムには、完全な C/C++ コンパイラやデバッガスイートを含む完全な開発ツールがついてきます。他の多くの言語へのサポートも ports および package コレクションから利用できます。

FreeBSD は、無料でダウンロードできます。 また、CD-ROM または DVD でも入手可能です。 詳しくは FreeBSD の入手方法 をご覧ください。

1.2.2. FreeBSD はどこに使われていますか?

FreeBSD は、ウェブサービスの能力で知られています。 FreeBSD が利用されている代表的なサイトには Hacker News, Netcraft, NetEase, Netflix, Sina, Sony Japan, Rambler, Yahoo! および Yandex があります。

FreeBSD は、 先進的な機能、高いセキュリティ、および定期的なリリースサイクル、そして寛容なライセンスにより、 多くの商用およびオープンソースのアプライアンス、デバイスおよび製品を構築するプラットフォームとして利用されています。 世界最大規模の多くの IT 会社が FreeBSD を使っています。

- Apache - Apache ソフトウェア財団は、 1.4 百万回を超えるコミットというおそらく世界で最も大規模な SVN リポジトリを含む、数多くの公式のインフラストラクチャで FreeBSD を使っています。
- Apple - Apple により提供されている最近のオペレーションシステムは、FreeBSD

からプロセスモデル、ネットワークスタック、仮想ファイルシステム、ライブラリ、マニュアルページ、そしてコマンドラインユーティリティについてのコードを取り入れています。

- Cisco - IronPort ネットワークセキュリティおよびアンチスパムアプライアンスは、 改造された FreeBSD カーネルで動いています。
- Citrix - NetScaler の一連のセキュリティアプライアンスは、 FreeBSD シェルとともに 4-7 レイヤのロードバランス、 コンテントキャシュ、アプリケーションファイアウォール、 セキュリティ VPN およびモバイルクライド・ネットワークアクセスを提供します。
- Dell EMC Isilon - Isilon 社のエンタープライズストレージアプライアンスは、FreeBSD ベースです。寛大な FreeBSD ライセンスのおかげで、Isilon は、彼らの知的財産物をカーネルに統合することができるため、 オペレーティングシステムではなく、製品そのものに焦点を当てた開発が可能となっています。
- Quest KACE - KACE システム管理アプライアンスでは、 FreeBSD が用いられています。信頼性、スケーラビリティおよび継続的な開発をサポートしているコミュニティが評価され採用されています。
- iXsystems - 統合ストレージアプライアンスの TrueNAS シリーズは FreeBSD ベースです。
- Juniper - Juniper のすべてのネットワークギア (ルータ、スイッチ、セキュリティおよびネットワークアプライアンス) を動かしている JunOS オペレーティングシステムは、 FreeBSD ベースです。 Juniper は、FreeBSD プロジェクトと商用製品を提供しているベンダとの間で協力関係が成功している数多くのベンダのひとつです。 将来 FreeBSD の新しい機能を JunOS へと統合する際の複雑さを減らすため、 Juniper で作成された改良点は、FreeBSD に取り込まれています。
- McAfee - Sidewinder などの McAfee エンタープライズファイアウォール製品のベースである SecurOS は FreeBSD ベースです。
- NetApp - ストレージアプライアンスの Data ONTAP GX シリーズは、FreeBSD ベースです。 NetApp は、新しい BSD ライセンスのハイパーバイザである bhyve などの数多くの機能を FreeBSD プロジェクトに還元しています。
- Netflix - Netflix が顧客へのストリームムービーに使用している OpenConnect アプライアンスは、FreeBSD ベースです。 Netflix は、コードベースに対し多大な貢献を行っており、 FreeBSD のメインラインからの差分がゼロになるように作業を行っています。 Netflix OpenConnect アプライアンスは、 北米の全インターネットトラフィックの 32% の配送を受け持っています。
- Sandvine - Sandvine は、ハイパフォーマンスでリアルタイムのネットワークプロセッシングプラットフォームのベースに FreeBSD を使用しています。このプラットフォームは、彼らのインテリジェントネットワークポリシーコントロール製品を構成しています。
- Sony - PlayStation Vita, PlayStation 4 および PlayStation 5 のゲームコンソールは、 FreeBSD の改良版が動いています。
- Sophos - Sophos Email アプライアンス製品は、強化された FreeBSD がベースです。インバウンドメールに対してスパムやウィルススキャンを行う一方で、アウトバウンドメールがマルウェアではないか、また、機密情報がアクシデントで漏洩してしまわないようにモニタします。
- Spectra Logic - アーカイブグレードストレージアプライアンスの nTier シリーズは、FreeBSD および OpenZFS が動いています。

- Stormshield - Stormshield ネットワークセキュリティアプライアンスは、 強化された FreeBSD がベースです。 BSD ライセンスが、彼らの知的財産のシステムへの統合を可能にする一方で、コミュニティに非常に多くの興味深い開発結果をもたらしてくれます。
- The Weather Channel - 各ローカルケーブルプロバイダのヘッドエンドにインストールされていて、ローカルの天気予報をケーブル TV ネットワークプログラムに送る IntelliStar アプライアンスでは FreeBSD が動いています。
- Verisign - Verisign は .com および .net ルートドメインレジストリおよび関連する DNS インフラストラクチャの運用に責任を持っています。
  彼らのインフラストラクチャに一般的な障害点がないように、FreeBSD
  を含むさまざまなネットワークオペレーティングシステムに信頼を寄せています。
- Voxer - Voxer のモバイルボイスメッセージのプラットフォームでは、 ZFS が FreeBSD 上で動いています。 Voxer は、Solaris から派生したオペレーティングシステムから、 FreeBSD へと移行しました。優れた文書、 幅広く活動的なコミュニティ、
  そして開発者にとって好意的な環境がその理由です。 ZFS および DTrace といった決定的な機能に加え、 FreeBSD では、 ZFS が TRIM に対応しています。
- Fudo セキュリティ - FUDO セキュリティアプライアンスは、
  エンタープライズおよびシステムの管理者に対し、 モニタ、コントロール、レコードおよび audit コントラクタを提供します。 ZFS, GELI, Capsicum, HAST および auditdistd といった FreeBSD の最良なセキュリティ機能がベースとなっています。

また、FreeBSD は関連したオープンソースプロジェクトを数多く生み出しています。

- BSD Router - 広く使われているエンタープライズルータの置き換えとなるような FreeBSD ベースのルータで、標準的な PC ハードウェアで動作するように設計されています。
- TrueNAS は、ネットワークアタッチトストレージ (NAS) ソフトウェアです。
  データの共有およびランサムウェアやマルウェアといった現代の脅威からデータを保護します。
  TrueNAS
  を使うことで、ユーザおよびクライアントデバイスは、仮想化および共有プロトコルを通して共有データに容易にアクセスできます。
- GhostBSD は、FreeBSD から派生しており、GTK 環境を使用して美しい見た目や使い勝手の良さを現代の BSD
  プラットフォームに実現し、自然でネイティブな UNIX® 環境を提供します。
- mfsBSD - メモリから完全に実行可能な FreeBSD システムのイメージを構築するためのツールキットです。
- XigmaNAS - PHP によるウェブインタフェースを搭載した FreeBSD ベースのファイルサーバのディストリビューションです。
- OPNSense は、オープンソースの使いやすく構築が簡単な FreeBSD ベースのファイアウォールおよびルータのプラットフォームです。 OPNsense は、
  高価な商用のファイアウォールや標準で利用可能なほとんどの機能を持っています。
  オープンで検証可能なソースと共に、 商品が提供している豊富な機能のセットを提供します。
- MidnightBSD は、BSD から派生したオペレーティングシステムで、デスクトップユーザを念頭において開発されています。
  このオペレーティングシステムには、メール、ウェブブラウザ、ワードプロセッサ、ゲームといった、日々の生活で必要と思われるすべてのソフトウェアが含まれています。

- NomadBSD は、FreeBSD ベースの USB フラッシュドライブのための永続的な live システムです。ハードウェアを自動的に認識してセットアップを行い、すぐにデスクトップシステムとして使えるように設定します。 データリカバリ、教育および FreeBSD のハードウェア互換性の試験にも使用できます。
- pfSense - 数多くの機能および拡張 IPv6 サポートを持つ FreeBSD ベースのファイアウォールディストリビューションです。
- ZRouter - FreeBSD ベースの組み込みデバイス用のオープンソースのファームウェアです。いつでも購入できるようなルータ上のプロプリエタリのファームウェアの置き換えとなるように設計されています。

FreeBSD Foundation のウェブサイトでは、FreeBSD を製品やサービスのベースに利用している会社の声が紹介されています。 Wikipedia にも FreeBSD ベースの製品のリスト がまとめられています。

1.3. FreeBSD プロジェクトについて

以下の節では簡単な歴史やプロジェクトの目標、
開発モデルなど、普段は表にでない話題を提供しています。

1.3.1. FreeBSD 小史

FreeBSD プロジェクトは 1993 年の始めに Unofficial 386BSD Patchkit の最後の 3 人のまとめ役によって、部分的に patchkit から派生する形で開始されました。ここでの 3 人のまとめ役というのは、Nate Williams, Rod Grimes と、 Jordan Hubbard です。

このプロジェクトのもともとの目標は、patchkit
という仕組みではもう十分に解決できなくなってしまった 386BSD の数多くの問題を修正するための、386BSD の暫定的なスナップショットを作成することでした。こういった経緯を経ているので、 このプロジェクトの初期の頃の名前は 386BSD 0.5 や 386BSD 暫定版 (Interim) でした。

386BSD は、Bill Jolitz が (訳注: バークレイ Net/2 テープを基に) 作成したオペレーティングシステムです。当時の 386BSD は、ほぼ一年にわたって放っておかれていた (訳注: 作者がバグの報告を受けても何もしなかった) というひどい状況に苦しんでいました。作者の代わりに問題を修正し続けていた patchkit は日を追うごとに不快なまでに膨張してしまっていました。 このような状況に対して、彼らは暫定的な "クリーンアップ" スナップショットを作成することで Bill を手助けしようと決めました。しかし、この計画は唐突に終了してしまいました。Bill Jolitz が、
このプロジェクトに対する受け入れ支持を取り下げることを突然決意し、
なおかつこのプロジェクトの代わりに何をするのかを一切言明しなかったのです。

たとえ Bill が支持してくれないとしても、 彼ら 3 人の目標には依然としてやる価値があると考えていたため、 David Greenman が考案した名称 "FreeBSD" をプロジェクトの名前に採用し、新たなスタートを切りました。この時点でのプロジェクトの初期目標は、すでにこのシステム (訳注: 386BSD + Patchkit) を使っていた利用者たちと相談して決められました。
プロジェクトが実現に向けて軌道に乗ってきたことが明確になった時点で、 Jordan は Walnut Creek CDROM 社に連絡してみました。CD-ROM を使って FreeBSD を配布することによって、インターネットに容易に接続できない多くの人々が FreeBSD

を簡単に入手できるようになると考えたからです。Walnut Creek CDROM 社は FreeBSD を CD で配布するというアイデアを採用してくれたばかりか、
作業するためのマシンと高速なインターネット回線をプロジェクトに提供してくれました。
当時は海のものとも山のものともわからなかったこのプロジェクトに対して、Walnut Creek CDROM 社が信じられないほどの信頼を寄せてくれたおかげで、 FreeBSD は短期間のうちにここまで大きく成長したのです。

CD-ROM による最初の配布 (そしてネットでの、 ベータ版ではない最初の一般向け配布) は FreeBSD 1.0 で、1993 年 12 月に公開されました。これはカリフォルニア大学バークレイ校の 4.3BSD-Lite ("Net/2") を基とし、386BSD や Free Software Foundation からも多くの部分を取り入れたものです。これは初めて公開したものとしては十分に成功しました。続けて 1994 年 5 月に FreeBSD 1.1 を公開し、非常に大きな成功を収めました。

この時期、 あまり予想していなかった嵐が遠くから接近してきていました。 バークレイ Net/2 テープの法的な位置づけについて、Novell 社とカリフォルニア大学バークレイ校との間の長期にわたる法廷論争において和解が成立したのです。和解の内容は、Net/2 のかなりの部分が "権利つき (encumbered)" コードであり、それは Novell 社の所有物である、というバークレイ校側が譲歩したものでした。なお、Novell 社はこれらの権利を裁判が始まる少し前に AT&T 社から買収していました。 和解における譲歩の見返りにバークレイ校が得たのは、 4.4BSD-Lite が最終的に発表された時点で、 4.4BSD-Lite は権利つきではないと公式に宣言されること、そしてすべての既存の Net/2 の利用者が 4.4BSD-Lite の利用へと移行することが強く奨励されること、という Novell 社からの "ありがたき天からの恵み" でした (訳注: 4.4BSD-Lite はその後 Novell 社のチェックを受けてから公開された)。FreeBSD も Net/2 を利用していましたから、1994 年の 7 月の終わりまでに Net/2 ベースの FreeBSD の出荷を停止するように言われました。ただし、 このときの合意によって、私たちは締め切りまでに一回だけ最後の公開をすることを許されました。 そしてそれは FreeBSD 1.1.5.1 となりました。

それから FreeBSD プロジェクトは、まっさらでかなり不完全な 4.4BSD-Lite を基に、文字どおり一から再度作り直すという、 難しくて大変な作業の準備を始めました。"Lite" バージョンは、部分的には本当に軽くて、中身がなかったのです。 起動し、動作できるシステムを実際に作り上げるために必要となるプログラムコードのかなりの部分がバークレイ校の CSRG (訳注: BSDを作っているグループ) によって (いろいろな法的要求のせいで) 削除されてしまっていたということと、4.4BSD の Intel アーキテクチャ対応が元々かなり不完全であったということがその理由です。 この移行作業は結局 1994 年の 11 月までかかりました。 そして 12 月に FreeBSD 2.0 として公開されました。これは、かなり粗削りなところが残っていたにもかかわらず、
かなりの成功を収めました。そしてその後に、より信頼性が高く、 そしてインストールが簡単になった FreeBSD 2.0.5 が 1995 年の 6 月に公開されました。

これ以降、FreeBSD の安定性、速さや機能は改善され、 リリースが行われてきました。

長期的な開発プロジェクトは 16.0-CURRENT 開発ブランチ (main) で続けられ、 16.0 のスナップショットリリースは、開発の進行状況に応じて スナップショットサーバ より継続して入手できます。

1.3.2. FreeBSD プロジェクトの目標

FreeBSD プロジェクトの目的は、いかなる用途にも使用でき、何ら制限のないソフトウェアを供給することです。 私たちの多くは、 コード (そしてプロジェクト)

に対してかなりの投資をしてきており、
これからも多少の無駄はあっても投資を続けて行くつもりです。ただ、
他の人達にも同じような負担をするように主張しているわけではありません。 FreeBSD
に興味を持っている一人の残らず全ての人々に、 目的を限定しないでコードを提供すること。これが、
私たちの最初のそして最大の "任務"
であると信じています。そうすれば、コードは可能な限り広く使われ、
最大の恩恵をもたらすことができるでしょう。これが、
私たちが熱烈に支持しているフリーソフトウェアの最も基本的な目的であると、 私は信じています。

私たちのソースツリーに含まれるソースのうち、 GNU 一般公有使用許諾 (GPL) または GNU
ライブラリー一般公有使用許諾 (LGPL)
に従っているものについては、多少制限が課せられています。ただし、
ソースコードへのアクセスの保証という、 一般の制限とはいわば逆の制限 (訳注1) です。 GPL
ソフトウェアの商利用には、そのライセンスにある 複雑な側面が影響してくることがあります。
ですから私たちは、そうすることが合理的であると判断されたときには、 より制限の少ない、BSD
ライセンスを採用しているソフトウェアを選択するようにしています。

(訳注1) GPL では、「ソースコードを実際に受け取るか、
あるいは、希望しさえすればそれを入手することが可能であること」を求めています。

1.3.3. FreeBSD の開発モデル

FreeBSD は 非常に開かれた、柔軟性のあるプロセス により開発されています。 貢献者リスト
を見ていただければわかるとおり、FreeBSD
は文字通り世界中の何千という人々の努力によって開発されています。 FreeBSD
の開発環境は、この何千という開発者がインターネット経由で共同作業できるようになっているのです。
新しいボランティアはいつでも大歓迎です。 また、より密接に関わりたいと考える方は FreeBSD
への貢献 という文書をご覧ください。

あと、FreeBSD プロジェクトとその開発プロセスについて、
どなたにも知っていていただきたいのは以下のようなことです。

Git リポジトリ

長年にわたり FreeBSD のソースツリーは、 ソースコード管理用のフリーソフトウェアである CVS (Concurrent Versions System) によってメンテナンスされてきました。 2008 年 6 月、プロジェクトはソースコード管理のシステムを SVN (Subversion) に移行しました。 ソースツリーの急速な増加や、これまでに蓄積された膨大な量の履歴によって、CVS の持つ技術的な限界が明かになってきたためです。 ドキュメンテーションプロジェクトと Ports Collection リポジトリも、それぞれ 2012 年 5 月と 7 月に CVS から SVN へと移行しました。 そして 2020年 12 月、プロジェクトは ソースおよびドキュメンテーションのリポジトリ を Git へ移行し、2021 年 4 月に Ports を移行しました。 FreeBSD `src/` リポジトリを取得するための情報は ソースコードの入手 の章を、 FreeBSD Ports Collection を取得するための詳細については Ports Collection の利用 の章をご覧ください。

ソースツリー管理者

コミッター *(committers)* は Git リポジトリへの *push* 権限 を持っている人、FreeBSD のソースに変更を加えることができる人です
(リポジトリに変更を加えるには、ソースをコントロールする `commit`
というコマンドを使うので、これらの人々は英語では "committers" と呼ばれます)。

もしバグを見つけたのであれば、障害報告データベース に提出してください。 FreeBSD メーリングリスト、IRC
チャネルまたはフォーラムは、その問題がバグかどうかを確認する助けとなりますので、障害報告を提出する前に、これらを使って確認してください。

FreeBSD コアチーム

FreeBSD コアチーム は FreeBSD プロジェクトが会社だとすると取締役会にあたるものです。
コアチームとして一番重要な役割は FreeBSD プロジェクトが全体としてよい方向に向かっていることを確認することです。
責任感あふれる開発者を上記のソースツリー管理者として招くこと、
また仕事上の都合などでコアチームをやめた人たちの後任を見つけることもコアチームの役割です。
現在のコアチームは FreeBSD 開発者 (committer) の中から 2022 年 5 月に選挙によって選出されました。 コアチームを選出するための選挙は、2 年ごとに行なわれています。

忘れてほしくないのは、多くの開発者同様に、 コアチームのほとんどは FreeBSD に対してボランティアの立場であり、 FreeBSD プロジェクトからは何ら金銭的な支援を受けていない、
ということです。ですから、 ここでの"責任"は "保証されたサポート"ではありません。 そういう意味で、上記の"取締役会"という例えはあまりよくないかもしれません。むしろ、FreeBSD のために人生を棒に振ってしまった人の集まりといった方が正しいかも...。

The FreeBSD Foundation

The FreeBSD Foundation は、FreeBSD プロジェクトおよびコミュニティを全世界的にサポートしたり促進することを目的としたアメリカ合衆国における 501(c)(3) に認定された非営利団体です。 この Foundation は、ソフトウェア開発に対するプロジェクトの助成を通じて資金を提供したり、緊急の事態に対する迅速な対応や新しい特徴や機能の実装に対して、スタッフを提供します。 FreeBSD のインフラストラクチャの改善および維持する目的でハードウェアを購入したり、テストカバレッジ、継続的インテグレーションおよび自動化の改善のためにスタッフを雇用しています。
世界中で開催されている技術的な会議やイベントにおいて、FreeBSD をプロモーションすることで FreeBSD を宣伝しています。
また、ワークショップ、教育的な教材やプレゼンテーションを提供して、より多くのユーザや FreeBSD の貢献者をリクルートしています。
さらに、契約の締結、ライセンス契約、およびその他の法的主体必要となる協定において FreeBSD プロジェクトを代表しています。

その他のコントリビュータ

最後になりますが、
もっとも重要で多数をしめる開発者はフィードバックやバグフィクスをどんどん送ってくれるユーザ自身です。 FreeBSD のベースシステムの開発に関わっていきたいという人は、 議論の場である FreeBSD technical discussions メーリングリスト に参加するとよいでしょう。
サードパーティ製のアプリケーションの移植に関わる議論は、 FreeBSD ports メーリングリストで行われています。 FreeBSD のメーリングリストに関する詳細は、 インターネット上のリソースをご覧ください。

FreeBSD への貢献者リスト は日に日に長くなっています。 あなたも今日、FreeBSD

へ何か送ることからはじめてみませんか?
もちろん、コードを書くほかにもいろいろな方法があります!

ひとことで言うと、FreeBSD の開発組織はゆるやかな同心円状になっています。ともすると中央集権的に見えがちなこの組織は、FreeBSD のユーザがきちんと管理されたコードベースを容易に追いかけられるようにデザインされているもので、貢献したいという人を締め出す意図は全くありません!
私たちの目標は安定したオペレーティングシステムと簡単にインストールして使うことのできるアプリケーションを提供することです。この方法は、それを達成するために非常にうまくはたらきます。

これから FreeBSD の開発にたずさわろうという人に、私たちが望むことはただ一つです。FreeBSD の成功を継続的なものにするために、現在の開発者と同じような情熱を持って接してください!

1.3.4. サードパーティ製プログラム

FreeBSD では基本配布セットに加え、移植されたソフトウェア集として数千の人気の高いプログラムを提供しています。ports には HTTP サーバから、ゲーム、言語、エディタまでありとあらゆるものが含まれています。37000 以上の ports (移植ソフトウェア) が存在します。Ports Collection 全体でも 3 GB 程度にしかなりません。ports をコンパイルするには、インストールしたいと思っているプログラムのディレクトリに移動し、`make install` とすると、あとはすべてシステムがやってくれます。どの ports もオリジナルの配布セットを動的に取ってくるので、ディスクは構築したいと思っている ports の分だけを準備しておけば十分です。ほとんどの ports は、すでにコンパイルされた状態で "package" として提供されており、ソースコードからコンパイルしたくない場合、これを使うと (`pkg install` というコマンドで) 簡単にインストールできます。package と ports に関する詳細は、アプリケーションのインストール - packages と ports をご覧ください。

1.3.5. ドキュメント

サポートが行われているすべての FreeBSD では、システムのセットアップ時にインストーラを使って、ドキュメントを /usr/local/share/doc/freebsd 以下にインストールできます。システムをインストールした後は、package を使ってドキュメントをインストールできます。

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

各言語に翻訳されたドキュメントをインストールするには、"en" の部分を使用する言語のコードに置き換えてください。
翻訳されたドキュメントの中には、古い情報のままの文書があり、現在では正確でなかったり関係ない内容が含まれている可能性があることに注意してください。
ローカルにインストールされたドキュメントは、HTML ブラウザを使って以下の URL から参照できます。

FreeBSD ハンドブック **(**英文オリジナル**)**

/usr/local/share/doc/freebsd/en/books/handbook/book.html

FreeBSD に関する **FAQ (**英文オリジナル**)**

/usr/local/share/doc/freebsd/en/books/faq/book.html

最新版の文書は常に https://docs.FreeBSD.org/ にありますので、こちらも参照してください。

Chapter 2. FreeBSD のインストール

2.1. この章では

FreeBSD は、amd64, ARM®, RISC-V®, および PowerPC® 等のさまざまなアーキテクチャに対応しています。

インストールまたは直接 FreeBSD を実行するためのイメージは、アーキテクチャおよびプラットフォームごとに ダウンロード できます。

利用できるイメージのタイプは以下の通りです。

- qcow2, vmdk, vhd および raw デバイスイメージといった仮想マシンのディスクイメージ。これらはインストール用のイメージではなく、FreeBSD がすでにインストールされたインスタンスで、すぐに起動して、インストール後の作業を行うことができます。仮想マシンのイメージは、クラウド環境でも使われます。
- Raspberry Pi のような組み込みシステム用の SD カードイメージ。これらのファイルをダウンロードしたら、展開し、ボードが起動するように raw イメージとして SD カードに書き込んでください。
- ISO または USB デバイスから起動して、FreeBSD を通常のデスクトップ、ラップトップ、サーバシステムのドライブ上にインストールするためのインストールイメージ。

この章では、3 番目のケースについて、bsdinstall と呼ばれるテキストベースのインストールプログラムを用いた FreeBSD のインストール方法について説明します。

この章では、以下について説明します。

- FreeBSD イメージの入手方法および FreeBSD インストールメディアの作り方。
- bsdinstall の起動方法。
- bsdinstall が聞いてくる質問がどのような意味であり、 またどのように答えれば良いか。
- インストールに失敗した時の問題の解決方法。
- インストールを確定する前に、 FreeBSD の live 版へアクセスする方法。

2.2. 最小ハードウェア要件

FreeBSD をインストールするために必要なハードウェア要件は、アーキテクチャおよびバージョンごとに異なります。 FreeBSD の各リリースが対応しているハードウェアアーキテクチャおよびデバイスの一覧は、FreeBSD リリース情報 のページにまとめられています。アーキテクチャごとに推奨される適切なイメージの選択に関しては、 FreeBSD ダウンロードページ でも説明されています。

2.3. インストール前に行う作業

システムが FreeBSD のインストールのための最小ハードウェア要件を満たしていることを確認したら、インストールファイルをダウンロードして、インストール用のメディアを用意してください。
その前に、以下のチェックリストを確認して、システムをインストールする準備ができていることを確認してください。

1. 重要なデータのバックアップ

   オペレーティングシステムをインストールする前に、常に価値のあるすべてのデータを最初にバックアップしてください。
   インストールしようとしているシステムにはバックアップを保存しないでください。
   そのかわり、USB ドライブ、ネットワーク上の他のシステム、もしくはオンラインのバックアップサービスといったリムーバルディスクにデータを保存してください。
   インストールを始める前に、バックアップを調べて、必要なすべてのファイルがバックアップに含まれていることを確認してください。
   インストーラがシステムのディスクをフォーマットしてしまうと、ディスクに保存されていたすべてのデータは失われます。

2. FreeBSD をインストールする場所の決定

   インストールするオペレーティングシステムが FreeBSD のみであれば、このステップは飛ばすことができます。しかし、ディスクに FreeBSD と他のオペレーティングシステムを共存させる必要がある場合には、FreeBSD が利用するディスクおよびパーティションを決める必要があります。

   i386 および amd64 アーキテクチャでは、二つのパーティションスキームのどちらかを使って、ハードディスクを複数の塊に分割することができます。伝統的な *Master Boot Record* (MBR) では、ディスク 1 台あたり最大 4 つのプライマリパーティションをパーティションテーブルに持つことができます。歴史的な理由により、FreeBSD では、これらのパーティションのことを スライス と呼びます。プライマリパーティションの 1 つに、複数の 論理パーティション を含む 拡張パーティション を作成できます。*GUID Partition Table* (GPT) は、ディスクをパーティションに分ける簡単で新しい方法です。一般的な GPT の実装では、1 つのディスクに 128 個までのパーティションの作成が可能であり、論理パーティションは必要ありません。

   FreeBSD のブートローダは、プライマリまたは GPT パーティションのどちらかを必要とします。ディスク上のプライマリ、もしくは GPT パーティションがすべて使われているのであれば、そのひとつを FreeBSD のために開放してください。
   ディスクにあるデータを消去せずにパーティションを作成するには、パーティションサイズを変更するツールを使って今あるパーティションのサイズを小さくし、空いたスペースに新しいパーティションを作成してください。

   パーティションサイズを変更するためのフリーまたは商用のツールは、List of disk partitioning software wikipedia entry にまとめられています。GParted Live は、GParted パーティションエディタを含むフリーのライブ CD です。

   ディスクパーティションを縮小するユーティリティは、

適切に用いるとパーティション用の空き容量を新しく安全に作成できます。
すでにあるパーティションを間違って選択してしまう可能性があるので、
ディスクのパーティションを変更する前に、
必ず重要なデータのバックアップをとり、
バックアップが正しくとれていることを検証してください。

ディスクパーティションごとに異なるオペレーティングシステムをインストールすることで、一つのコンピュータに複数のオペレーティングシステムをインストールできます。仮想化技術を用いると、ディスクパーティションを変更することなく、複数のオペレーティングシステムを同時に起動できます。

3. ネットワーク情報の収集

   FreeBSD のインストール方法によっては、ネットワークに接続し、インストールファイルをダウンロードする必要があります。
   インストールする方法に関わらず、インストール後に、
   インストーラはシステムのネットワークインタフェースの設定をする機会を提供します。

   ネットワークに DHCP サーバがあると、自動的にネットワークの設定情報を取得できます。DHCP を利用できない環境では、システムの以下のネットワーク情報について、システム管理者かプロバイダにネットワーク情報を問い合わせる必要があります。

   a. IP アドレス
   b. サブネットマスク
   c. デフォルトゲートウェイの IP アドレス
   d. ネットワークのドメイン名
   e. ネットワークの DNS サーバの IP アドレス

4. FreeBSD Errata の確認

   FreeBSD プロジェクトでは FreeBSD の各リリースができる限り安定するよう努力していますが、
   時々バグが発生してしまうことがあります。
   極まれに、発生したバグがインストールプロセスに影響を与えることがあります。
   これらの問題は発見され解決されると、FreeBSD の各バージョンごとに Errata ページに記載されます。
   インストールに影響するような既知の問題が無いことを、インストールする前に Errata で確認してください。

   すべてのリリースに関する情報や Errata は、リリース情報 のページで確認できます。

2.3.1. インストールメディアの準備

FreeBSD のインストーラは、他のオペレーティングシステムで実行できるようなプログラムではありません。そのかわり、FreeBSD インストールファイルをダウンロードしたら、ファイルタイプやサイズに合わせてメディア (CD, DVD または USB) に焼いてください。そして、挿入したメディアからインストールするように、システムを起動してください。

FreeBSD のインストールファイルは FreeBSD ダウンロードページ から入手できます。

各インストールファイルの名前は、FreeBSD
のリリースバージョンおよびアーキテクチャ、ファイルタイプから構成されます。

インストールファイルは、さまざまな形式で、xz(1)
により圧縮されたファイルまたは圧縮されていないファイルが用意されています。
用意されているフォーマットは、
コンピュータのアーキテクチャやメディアのタイプによって異なります。

インストールファイルの形式

- `-bootonly.iso`: インストーラのみを含む最小のインストールファイルです。
  インストールを行う間、インストーラは FreeBSD
  をインストールするために必要なファイルをダウンロードするため、ネットワーク接続が必要です。
  このファイルは、光学メディアに書き込む必要があります。
- `-disc1.iso`: FreeBSD のインストールに必要となる、ソースおよび Ports Collection
  といったすべてのファイルが含まれています。
  このファイルは、光学メディアに書き込む必要があります。
- `-dvd1.iso`: FreeBSD のインストールに必要となる、ソースおよび Ports Collection
  といったすべてのファイルが含まれています。
  インターネットに接続することなく、メディアのみでシステムのインストールを完了できるように、
  良く使われるウィンドウマネージャおよびアプリケーションをインストールするためのバイナリ
  package も含まれています。 このファイルは、光学メディアに書き込む必要があります。
- `-memstick.img`: FreeBSD のインストールに必要となる、ソースおよび Ports Collection
  といったすべてのファイルが含まれています。 イメージファイルを USB に書き込む
  で説明されている手順に従って、このファイルを USB スティックに書き込んでください。
- `-mini-memstick.img`: `-bootonly.iso`
  と同じく、インストールファイルは含まれていないため、必要に応じてダウンロードする必要があります。 インストールを行う間、ネットワーク接続が必要です。 イメージファイルを USB に書き込む
  の説明に従って、USB スティックに書き込んでください。

イメージファイルをダウンロードしたら、同じディレクトリから少なくとも一つの チェックサム
ファイルをダウンロードしてください。 2 つの チェックサム ファイルが利用可能です。
これのファイル名にはリリース番号とアーキテクチャ名がついています。 たとえば `CHECKSUM.SHA256-FreeBSD-13.1-RELEASE-amd64` および `CHECKSUM.SHA512-FreeBSD-13.1-RELEASE-amd64`
という名前がつけられます。

どちらかの (もしくは両方の) ファイルをダウンロードしたら、イメージファイルの チェックサム
を計算し、 チェックサム ファイルに示されている値と比較してください。 計算した チェックサム は、2
つの異なるアルゴリズム (SHA256 および SHA512) に対応する適切なファイルと比較してください。
FreeBSD では、チェックサム のために sha256(1) および sha512(1) を提供しています。
他のオペレーティングシステムでも同じようなプログラムを利用できます。

FreeBSD での チェックサム の検証は、以下のように sha256sum(1) (または sha512sum(1))
を使用して自動的に行うことができます。

```
% sha256sum -c CHECKSUM.SHA256-FreeBSD-13.1-RELEASE-amd64 --ignore-missing
FreeBSD-13.1-RELEASE-amd64-dvd1.iso: OK
```

チェックサムは完全に一致している必要があります。 もしチェックサムが一致しなければ、イメージファイルが壊れている可能性があるので、もう一度ダウンロードしてください。

イメージファイルを USB に書き込む

`*memstick.img` ファイルは、完全なメモリスティックの内容のイメージです。これは、通常のファイルのように対象のデバイスにコピーすることはできません。 USB スティックへ `*.img` を書き込むためのアプリケーションは複数あります。この節ではこのうちの二つのユーティリティについて説明します。

先に進む前に、USB スティックに存在する重要なデータをバックアップしてください。以下の手順を実行すると、 スティックに存在するデータは削除されます。

Procedure. `dd` を使ってイメージを書き込む

この例では、イメージの書き込み先のターゲットデバイスとして、`/dev/da0` が使われています。 ここで使われるコマンドは、指定したターゲットデバイスに存在しているデータを破壊してしまうので、正しいデバイスが指定されていることに 細心の注意を払ってください。

1. dd(1) コマンドユーティリティは、 BSD, Linux®, および Mac OS® システムで利用できます。`dd` を使ってイメージを焼くには、USB スティックを挿入して、デバイス名を確定してください。 その後、ダウンロードしたインストールファイルおよび、 USB スティックのデバイス名を指定してください。 この例では、amd64 インストールイメージを FreeBSD システムの最初の USB デバイスに書き込みます。

```
# dd if=FreeBSD-13.0-RELEASE-amd64-memstick.img of=/dev/da0 bs=1M conv=sync
```

もし上記のコマンドに失敗するようでしたら、USB スティックがマウントされているかどうか、デバイス名が (パーティションではなく) ディスクに対して指定されていることを確認してください。

オペレーティングシステムによっては、このコマンドを sudo(8) で実行することが求められる場合があります。 dd(1) の書式は、プラットフォームによって少し変わります。 たとえば Mac OS® では、小文字の `bs=1m` を使う必要があります。 Linux® のようなシステムでは、書き込みをバッファします。すべての書き込みを完了させるには、sync(8) を使用してください。

Procedure. Windows® を使ってイメージを書き込む

適切なドライブレターを出力先に設定していることを十分に確認してください。さもなければ、現在あるデータは上書きされ、 破壊されてしまうでしょう。

1. **Image Writer for Windows®** を入手する

Image Writer for Windows® は、イメージファイルをメモリスティックに正しく書き込むことのできるフリーのアプリケーション

です。 win32diskimager ホームページ からダウンロードして、フォルダに展開してください。

2. イメージライタを使ってイメージを書き込む

   Win32DiskImager アイコンをダブルクリックして、プログラムを起動してください。 Device の下に表示されるデバイスレターが、
   メモリスティックのドライブであることを確認してください。
   フォルダのアイコンをクリックして、
   メモリスティックに書き込むイメージファイルを選択します。 **[Save]** をクリックして、
   イメージファイルの名前を確定してください。 すべてが正しく行われたかどうか、また、
   他のウィンドウでメモリスティックのフォルダが開かれていないことを確認してください。
   準備ができたら、**[Write]** をクリックして、
   メモリスティックにイメージファイルを書き込んでください。

2.4. インストールの開始

デフォルトでは、次のメッセージが表示されるまで
インストーラはディスクに何の変更も加えません。

```
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?
```

この警告の前であれば、いつでもインストールを中断できます。
もし、何かを間違って設定してしまったことが心配ならば、
最後の警告の前に単にコンピュータをオフにしてください。
システムのハードディスクを変更せずに済みます。

この章では、インストールメディアの準備
で説明されている手順によって準備されたインストールメディアから、システムを起動する方法について説明します。 起動可能な USB スティックを使用する場合には、コンピュータを立ち上げる前に、USB スティックを挿入してください。 CD もしくは DVD から起動する場合には、コンピュータを立ち上げ、すぐにメディアを挿入してください。
挿入したメディアからシステムを起動するように設定する方法は、アーキテクチャ毎に異なります。

2.4.1. FreeBSD ブートメニュー

インストールメディアからシステムが起動すると、 以下のようなメニューが表示されます。

図 *1. FreeBSD* ブートローダメニュー

デフォルトでは、メニューは、FreeBSD インストーラが起動するまで (FreeBSD がインストールされているシステムでは、FreeBSD が起動するまで)、ユーザからの入力を 10 秒間受け付けます。タイマーを停止してオプションを確認には、`Space` を押してください。オプションを選択するには、
ハイライトされている番号、文字、もしくはキーを押してください。以下のオプションが利用可能です。

- `Boot Multi User`: FreeBSD の起動プロセスを続けます。ブートタイマが停止しているのであれば `1`、大文字もしくは小文字の `B` または、`Enter` を押してください。
- `Boot Single User`: このモードは、すでにインストールされている FreeBSD を修復するために利用できます。シングルユーザモードについては、「シングルユーザモード」で説明されています。`2` もしくは、小文字もしくは、大文字の `S` を押すとこのモードに入ることができます。
- `Escape to loader prompt`: 制限された低レベルのコマンドのみが利用可能な修復用プロンプトでシステムを起動します。このプロンプトについては、「起動ステージ 3」で説明されています。`3` または `Esc` を押すとこのプロンプトで起動します。
- `Reboot`: システムを再起動します。
- `Cons: video, serial, Dual (serial primary)` または `Dual (Video primary)` でインストールを続けます。
- `Kernel`: 別のカーネルを読み込みます。
- `Boot Options`: FreeBSD ブートオプションメニュー で示されるメニューを開きます。

図 2. *FreeBSD* ブートオプションメニュー

この起動オプションメニューは、2つのセクションから構成されています。
最初のセクションは、メインのブートメニューに戻ったり、
オプションをデフォルト値に戻すために利用できます。

次のセクションでは、変更可能なオプションついて、選択されている番号や文字を、On や Off に変更できます。システムは、これらのオプションが変更されない限り、常に変更されたオプションで起動します。 このメニューで変更可能なオプションは以下の通りです。

- ACPI Support: 起動中にシステムが固まるようでしたら、このオプションを Off にしてください。
- Safe Mode: 上記のオプションの対応を行ってもシステムが起動時に固まるようでしたら、ACPI Support を Off にし、このオプションを On に設定してください。
- Single User: シングルユーザモードでインストールされている FreeBSD を修復には、On にしてください。シングルユーザモードについては、「シングルユーザモード」で説明されています。 問題が修正された後は、Off に戻してください。
- Verbose: 起動プロセスの表示をより詳細に表示したい場合には、このオプションを On にしてください。 ハードウェアの問題を解決する際には有効です。

設定が終わったら、1 または Backspace を押してメインブートメニューに戻り、Enter を押して FreeBSD の起動を続けてください。FreeBSD がハードウェアの検出を行い、インストールプログラムをロードしている間、ブートメッセージが表示されます。
起動後、ウェルカムメニュー が表示されます。

図 3. ウェルカムメニュー

Enter を押して、デフォルトの **[Install]** を選択すると、インストール作業が始まります。
この章の残りの部分では、このインストーラの使い方について説明します。
メニュー項目を選択するには、左右の矢印キーを使ったり、色付けされた文字を使ってください。
[Shell] を選択すると、インストールの前に、FreeBSD シェルからコマンドラインユーティリティでディスクを準備できます。 **[Live CD]** オプションを選択すると、 インストール前に FreeBSD を試すことができます。 live 版については、Live CD を使う で説明されています。

ハードウェアの検出などのブートメッセージを見るには 大文字または小文字の S を押してください。 その後、Enter を押して、 シェルにアクセスしてください。 シェルプロンプトから、more /var/run/dmesg.boot を入力してください。 メッセージのスクロールには、スペースバーを使ってってください。 終わったら、exit を押して、 ウェルカムメニューに戻ってください。

2.5. bsdinstall の使用

この章では、 bsdinstall メニューの順番と、 システムがインストールされる前に、尋ねられる情報の形式について紹介します。 メニューオプションの選択には、矢印キーを使い、メニューの項目の選択や解除する場合には、Space キーを使ってください。 設定が終わったら、Enter を押して設定を保存し、次の画面へ移動してください。

2.5.1. キー配列メニューの選択

このプロセスが始まると、 bsdinstall は キーマップの読み込み のようにキーマップファイルを読み込みます。

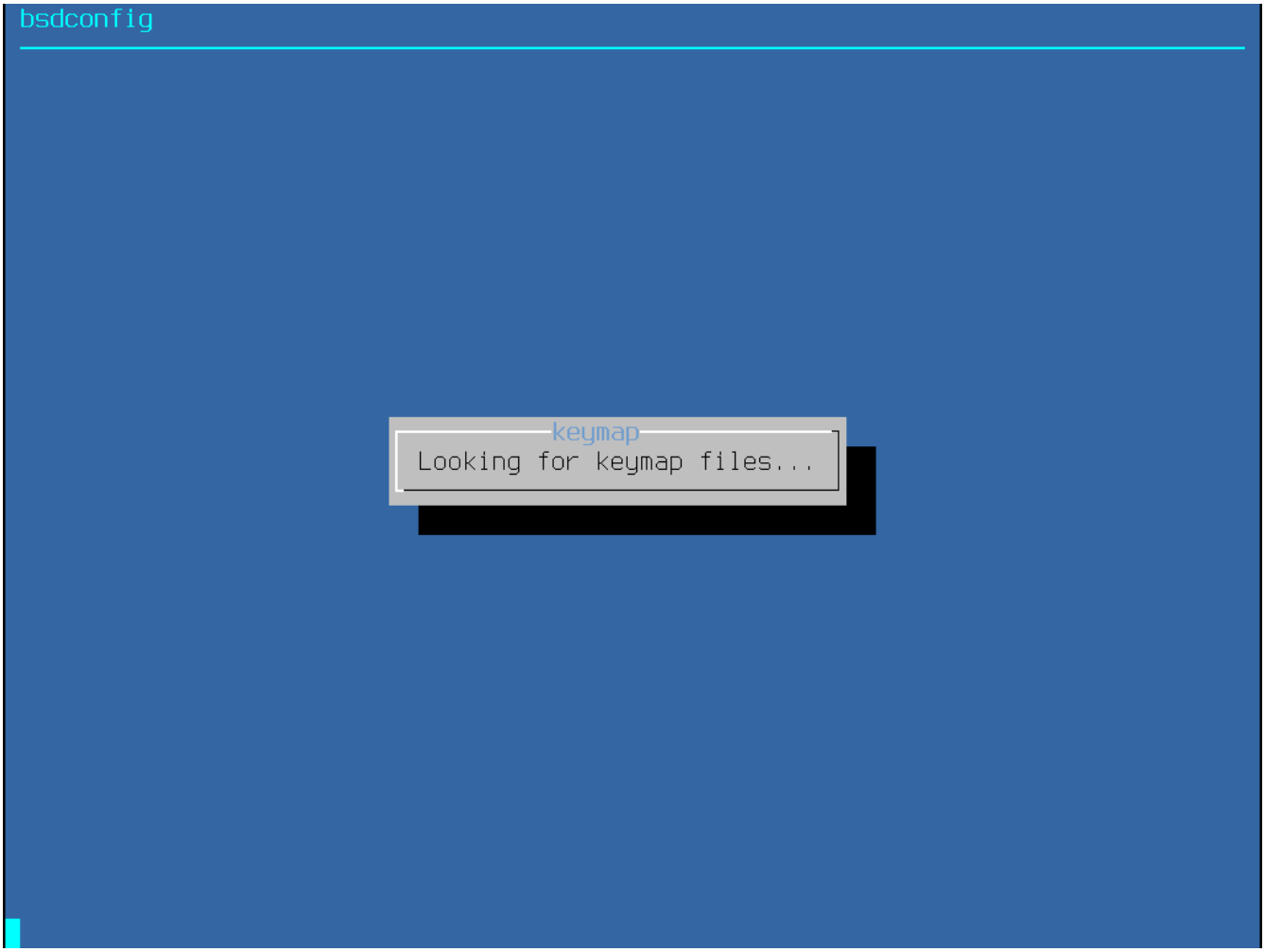


図 *4.* キーマップの読み込み

キーマップが読み込まれると、bsdinstall は キーマップ選択メニュー を表示します。
上下の矢印キーを使って、システムのキーボードに最も近いキーマップを選択してください。
選択を保存するには、`Enter` キーを押してください。

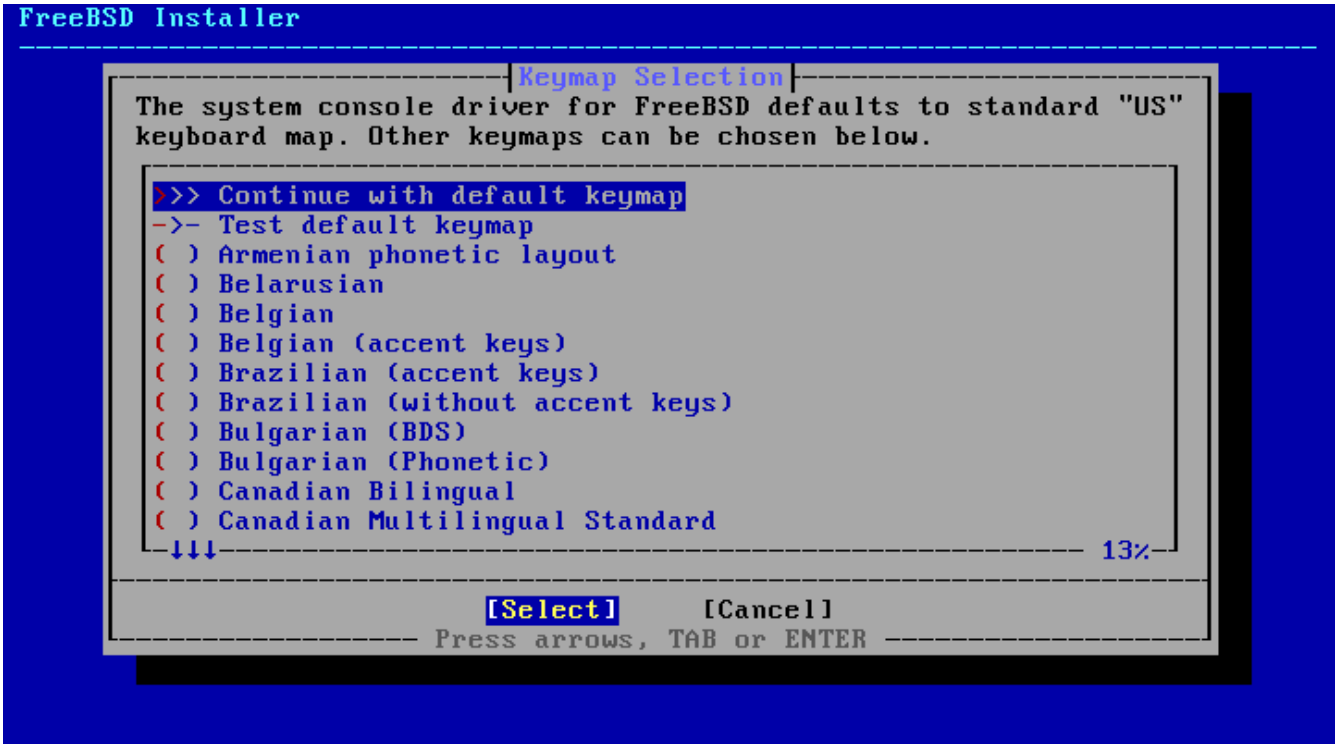


図 *5.* キーマップ選択メニュー

Esc を押すと、メニューは終了し、デフォルトのキーボードマップを使うようになります。
どのキーボードマップを選べばよいかわからない場合は、 United States of America ISO-8859-1 を選ぶとよいでしょう。

デフォルトとは異なるキーマップを選択した場合には、 キーマップテストメニュー でキーマップのテストを行い、 インストールを先に進む前に正しく動くかどうかを確認できます。

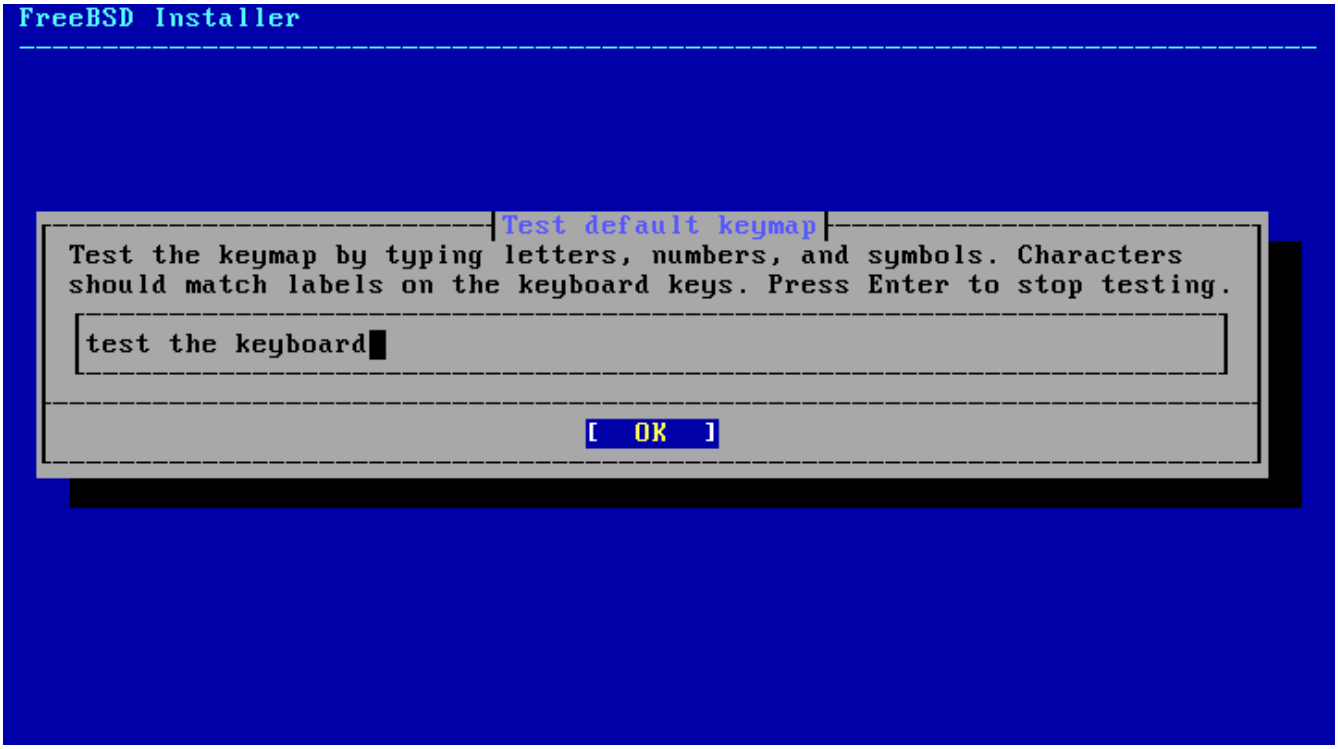


図 *6.* キーマップテストメニュー

2.5.2. ホスト名の設定

次の bsdinstall のメニューでは、 新しくインストールするシステムに与えるホスト名を設定します。

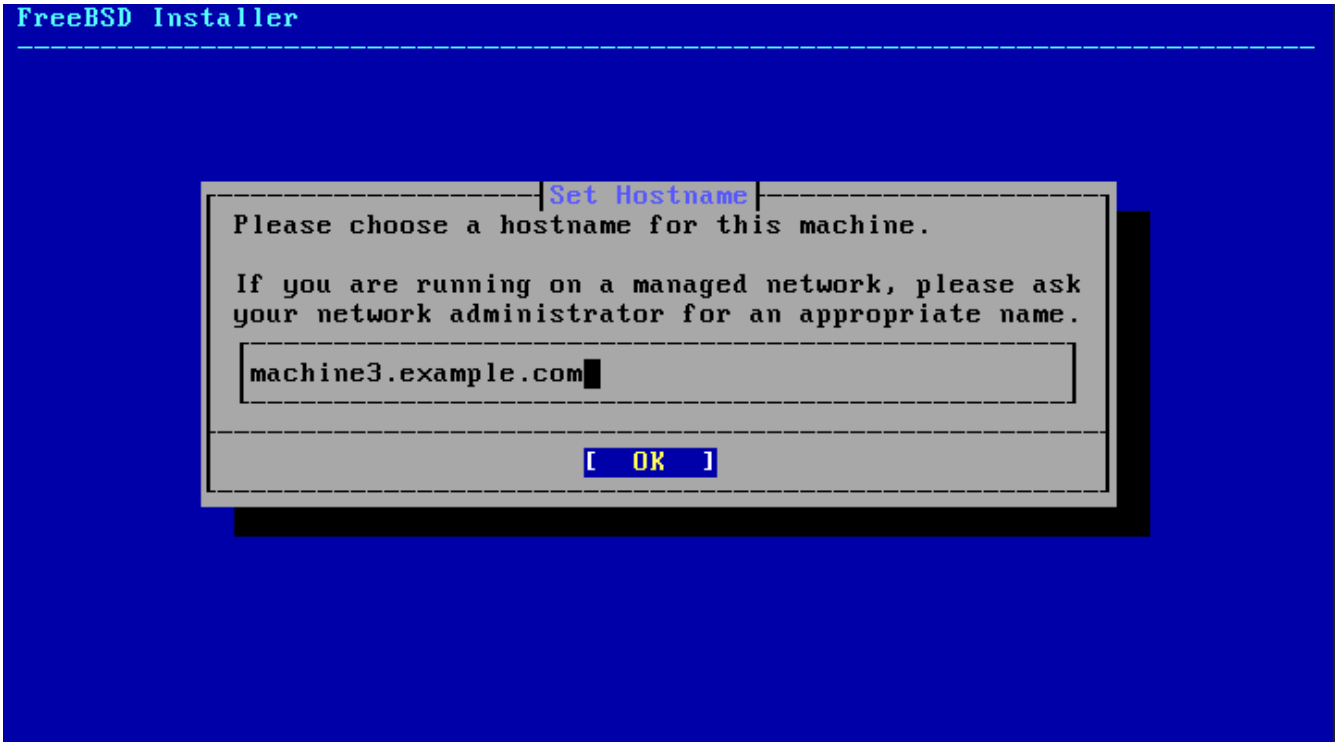


図 7. ホスト名の設定

ネットワーク上でユニークなホスト名を入力してください。 入力するホスト名は、`machine3.example.com` のように完全修飾のホスト名で入力してください。

2.5.3. インストールするコンポーネントの設定

次に、 bsdinstall は、インストールするオプションのコンポーネントの選択に移ります。

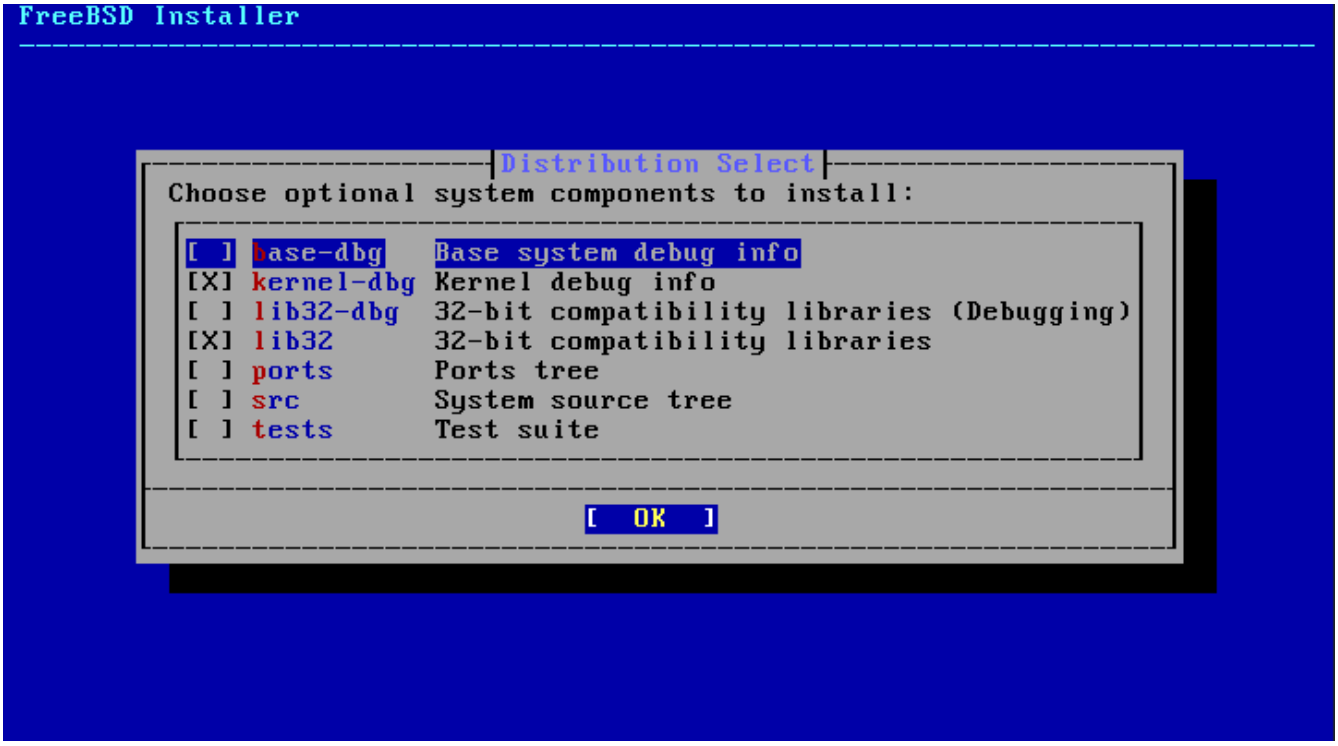


図 *8.* インストールするコンポーネントの設定

どのコンポーネントをインストールするかは、
システムの用途と用意されているディスク容量に依存します。 *base system* として知られている FreeBSD カーネルとユーザランドは、 常にインストールされます。
アーキテクチャによっては、表示されないコンポーネントもあります。

- `base-dbg` - デバッグシンボルを有効にしたベースツール (cat, ls や他の多くのツール)。
- `kernel-dbg` - デバッグシンボルを有効にしたカーネルおよびモジュール。
- `lib32-dbg` - 32-bit のアプリケーションを 64-bit 版の FreeBSD で実行する際に必要となる互換ライブラリ (デバッグシンボルは有効)。
- `lib32` - 32-bit のアプリケーションを 64-bit 版の FreeBSD で実行する際に必要となる互換ライブラリ。
- `ports` - FreeBSD Ports Collection は、 サードパーティ製ソフトウェアパッケージのダウンロード、コンパイル、 インストールを自動化するように設計されたファイルの集まりです。 Ports Collection の使い方については、 アプリケーションのインストール - packages と ports で説明します。

インストールプログラムは、システムのディスクに十分な空き容量があるかどうかを確認しないので、ハードディスクに十分な容量があるときだけ、このオプションを選択するしてください。 FreeBSD Ports Collection が必要とする容量は、約 3 GB です。

- `src` - FreeBSD のカーネルおよびユーザランド両方の完全なソースコードです。 ほとんどのアプリケーションは必要としませんが、 デバイスドライバやカーネルモジュール、 Ports Collection のアプリケーションによってはコンパイル時に必要となります。 このソースは、FreeBSD そのものの開発に使うこともできます。 すべてのソースツリーをインストールするには 1 GB のディスク容量を必要とします。 また、FreeBSD システム全体のコンパイルには、 さらに 5 GB の容量が必要です。
- `tests` - FreeBSD テストスイート。

2.5.4. ネットワークからのインストール

ネットワークからのインストール で示されているメニューは、-bootonly.iso または -mini-memstick.img からインストールする時のみ表示されます。
これらのインストールメディアはインストールファイルを含んでいません。　　このメニューは、ネットワーク経由でインストールファイルをダウンロードする必要があるため、ネットワークインタフェースを最初に設定する必要があることを示しています。
このメニューがインストールのプロセスで表示された場合には、　　ネットワークインタフェースの設定 に書かれている手順に従ってください。

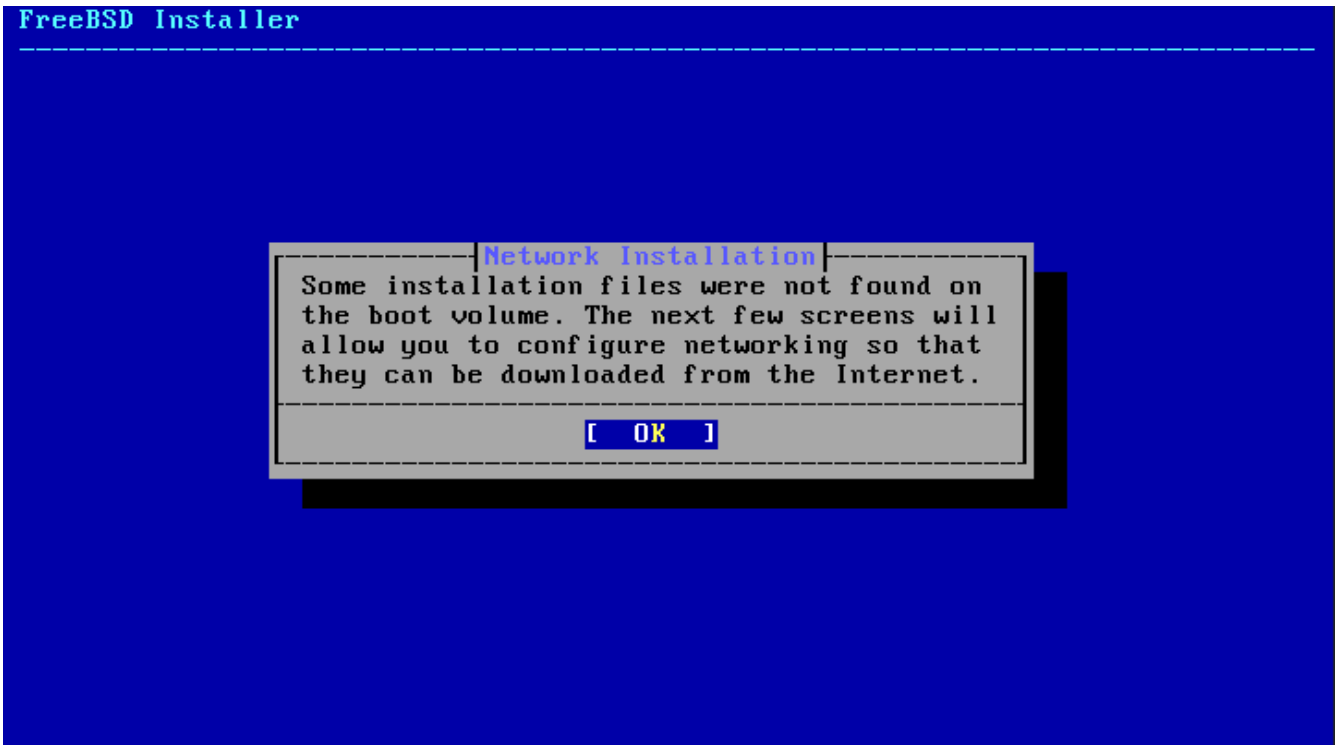


図 9. ネットワークからのインストール

2.6. ディスク領域の割り当て

次のメニューでは、ディスク領域を割り当てる方法を選択します。

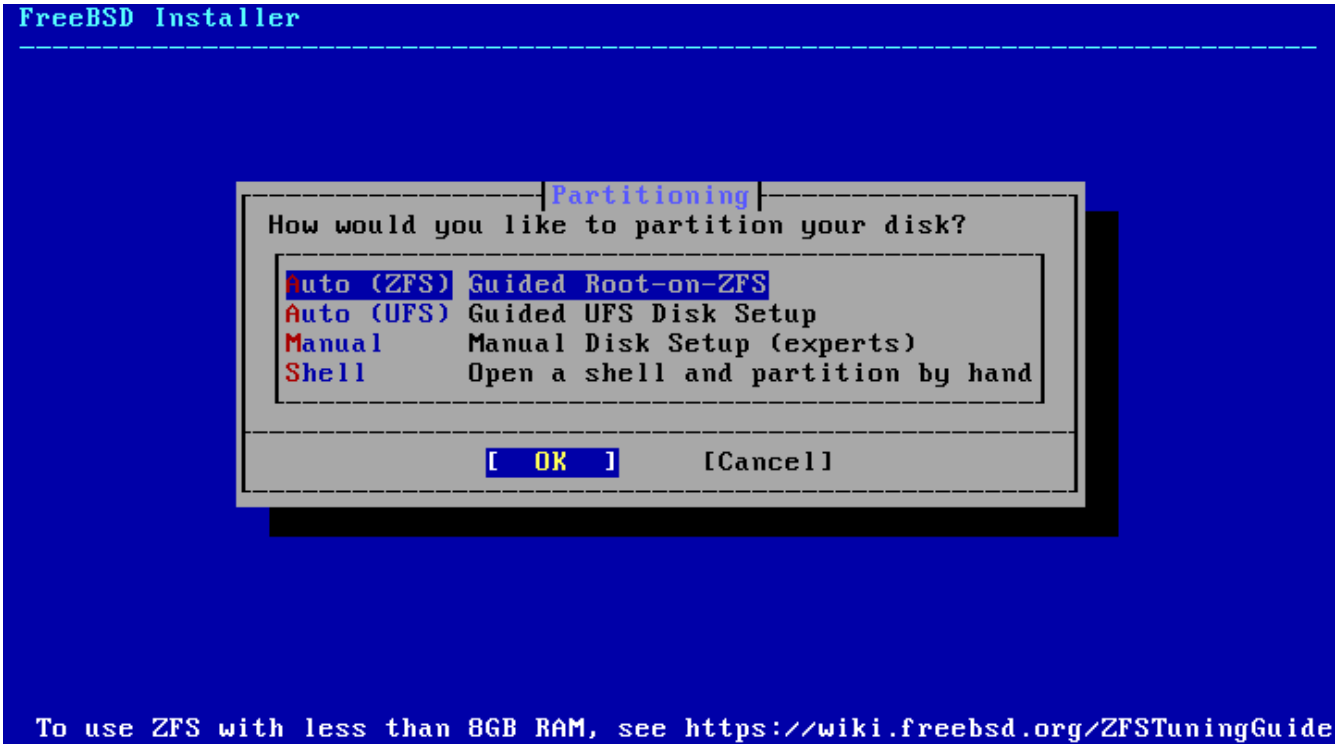


図 *10.* パーティション分割の選択

bsdinstall では、ディスク領域の割り当てのために 4 つの方法が用意されています。

- Auto (ZFS) によるパーティションの分割では、root-on-ZFS システムを構築します。GELI 暗号に対応した ブート環境 を構築することもできます。
- Auto (UFS) によるパーティションの分割では、UFS ファイルシステムを使ってディスクパーティションを自動的に分割します。
- Manual によるパーティションの分割は、高度な知識を持つユーザ向けで、メニューオプションからカスタマイズしたパーティションを作成できます。
- Shell では、シェルプロンプトを起動し、高度な知識を持つユーザが、gpart(8), fdisk(8), bsdlabel(8) のようなコマンドラインのプログラムを実行して、カスタマイズしたパーティションを作成できます。

この章では、ディスクパーティションをレイアウトする際の検討事項を説明します。
その後、各パーティションの作成方法について説明します。

2.6.1. パーティションレイアウトのデザイン

デフォルトのファイルシステムのパーティションレイアウトは、システム全体をひとつのファイルシステムで構成します。十分なディスク容量または複数のディスクを用いる環境において UFS を用いる場合は、複数のファイルシステムを検討する価値があります。
ファイルシステムのレイアウトを行う際には、
ハードディスクの外周部は内周部よりもデータ転送が速いということを思い出してください。
これに従えば、小さくて激しくアクセスされるファイルシステムを外周付近に、/usr のようなより大きなパーティションはディスクの内側に配置すべきでしょう。
そのため、パーティションを作成する際には、/、スワップ、/var, /usr のような順で作ってゆくのがよいでしょう。

/var パーティションのサイズは、あなたが計算機をどのように使おうとしているかを反映します。

このパーティションには主としてメールボックスやログファイル、　　プリンタスプールが置かれます。
メールボックスとログファイルは、
システムのユーザ数やログの保持期間に依存して予期し得ぬサイズにまで成長する可能性があります。
概して、ほとんどのユーザは、/var にギガバイト以上の空き容量を必要とはしないでしょう。

時には、たくさんのディスク容量が　　/var/tmp　　に必要になるときがあります。
新しいソフトウェアをインストールする際、package　　のツールは、package
の一時的なコピーを　　/var/tmp　　以下に展開します。　　/var/tmp
以下に十分なディスク容量が用意されていないと、Firefox　　や　　LibreOffice
のような、大きなソフトウェア　　package
のインストールが、困難になることがあります。

/usr　　パーティションには、FreeBSD　　Ports　　Collection
およびシステムのソースコードを含む、システムをサポートするのに必要な多くのファイル群が置かれます。　このパーティションには、少なくとも　2　ギガバイトの容量を用意することをおすすめします。
また、デフォルトではユーザのホームディレクトリは　　/usr/home
に置かれますが、他のパーティションに置くこともできます。　デフォルトでは、/home　は　/usr/home
へのシンボリックリンクです。

パーティションのサイズを考える時、　　必要量を念頭に置いてください。
別のパーティションには潤沢にスペースが余っているのに、
あるパーティションでスペースが足らないままというのは、 フラストレーションがたまるものです。

経験からスワップパーティションのサイズは物理メモリ (RAM) の 2 倍というのが一般的です。 RAM
の少ないシステムでは、もっとスワップを増した方が性能がよくなります
(大きなメモリの設定では少なくて済みます)。
スワップが少なすぎる設定は、あなたが後にメモリを増設したときに問題を起すばかりではなく、VM
ページスキャニングコードの能率を落します。

複数の　　SCSI　　ディスクや異なるコントローラで操作される複数の　　IDE
ディスクを持つ大規模なシステムでは、それぞれのドライブ　　(4　　台まで)
にスワップを設定することを推奨します。
各ドライブのスワップパーティションはほぼ同一サイズであるべきです。
カーネルは任意のサイズを扱うことができますが、内部のデータ構造は最大のスワップパーティションの
4　　倍に調節されます。
スワップパーティションをほぼ同一のサイズにしておくことで、カーネルはスワップスペースを最適なかたちでディスクをまたいでストライプさせることができます。
多くのスワップサイズを用意すると、全体のスワップに関するカーネルの警告メッセージがが表示されることがあります。
警告メッセージに示される手順に従って、スワップの割り当てのために許容されるメモリ量を増やすことで、この制限容量を増やすことができます。

あなたが通常スワップをたくさん使わないとしても、
プログラムが暴走しても再起動させられる前に回復することが容易になります。

システムを適切にパーティション化することで、小さいが書き込みの激しいパーティションによって引き起こされるフラグメント化を、読み出し専門のパーティションにまで波及させずにすみます。
また、書き込みの激しいパーティションをディスクの周辺部に配置することで、I/O
パフォーマンスを増大させることができます。　　大きなパーティション内の　　I/O
パフォーマンスもまた必要とされているでしょうが、ディスク周辺部へ移動させたとしても、/var
を周辺部に移動させることによって大きな効果が得られたのとは対照的に、意味のあるパフォーマンスの

増加は見込めないでしょう。

2.6.2. UFS を用いた Guided によるパーティションの分割

この方法を選択すると、 メニューには利用可能なディスクが表示されます。
複数のディスクが接続されている場合には、 FreeBSD
をインストールするディスクを選択してください。

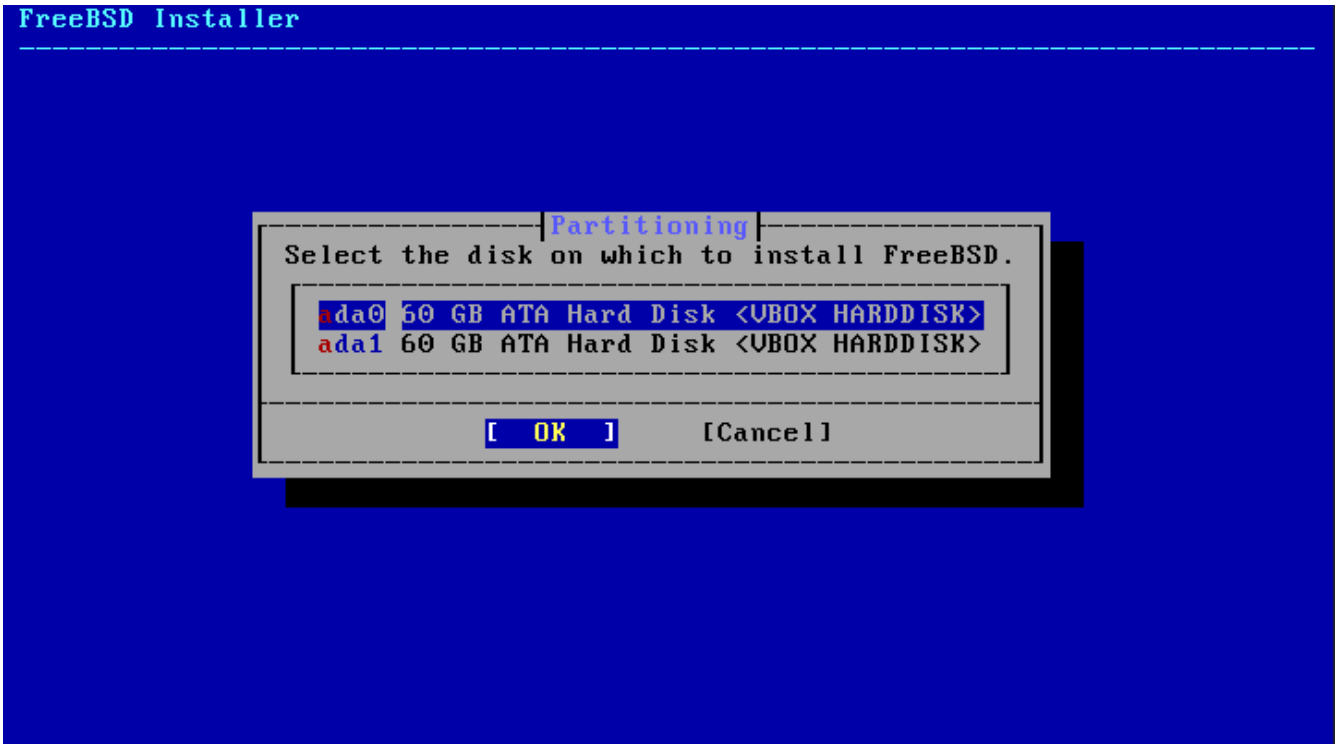


図 *11.* 複数のディスクから選択する

ディスクを選択したら、次のメニューでは、 ディスクのすべてにインストールを行うか、
または空き容量にパーティションを作成してインストールを行うかを設定します。 **[Entire Disk]**
を選択すると、 一般的なパーティションレイアウトが自動的に作成されます。 **[Partition]**
を選択すると、 ディスクの使用していない領域にパーティションレイアウトを作成します。

図 *12. Entire Disk* または *Partition* の選択

[Entire Disk] オプションを選択すると、bsdinstall はディスクの内容が消去されることを確認するダイアログを表示します。

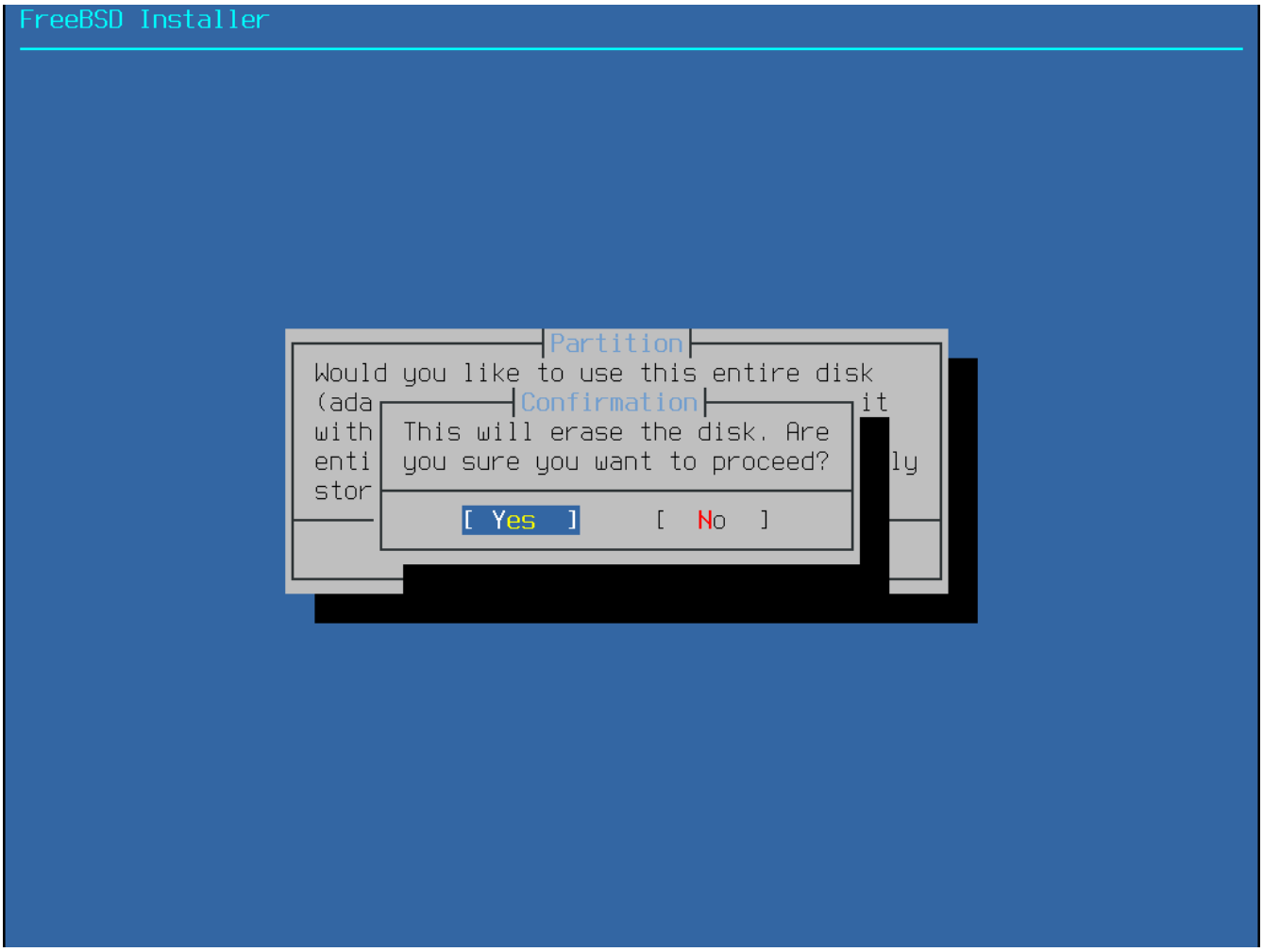


図 *13.* 確認

次のメニューでは、 利用可能なパーティションスキームタイプの一覧が表示されます。 amd64 コンピュータでは、通常 GPT が最も適切な選択となります。 GPT に対応していないような古いコンピュータでは、MBR を使う必要があります。 他のパーティションスキームは、使うことがまれであったり、古いコンピュータで用いられるものです。 パーティションスキーム に詳細があります。

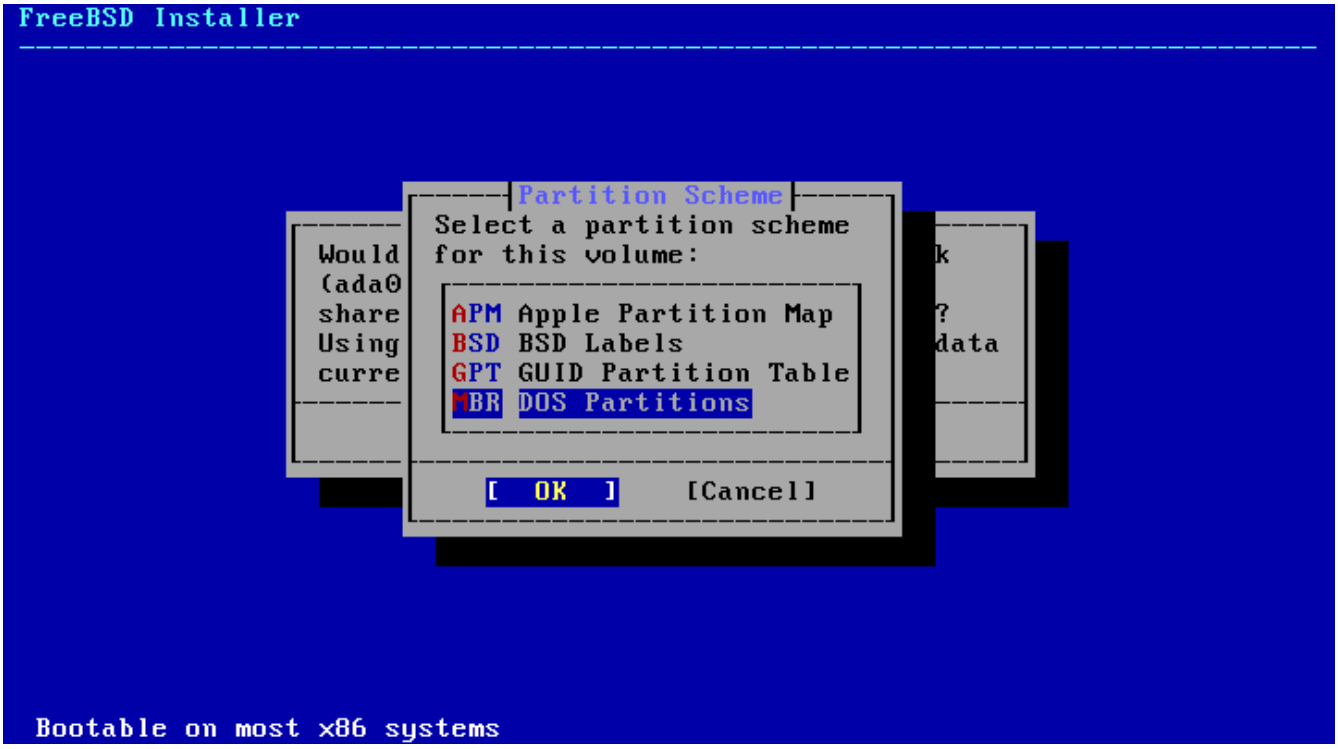


図 *14.* パーティションスキームの選択

パーティションのレイアウトを作成したら、インストールの条件を満たしているかどうかを深く確認してください。 **[Revert]** を選択すると、パーティションをオリジナルの値にリセットします。 また、**[Auto]** を選択すると、FreeBSD パーティションを自動的に作成します。 パーティションを手動で作成、変更、削除することもできます。 正しくパーティションを作成出来たら、 **[Finish]** を選択し、インストールを進めてください。

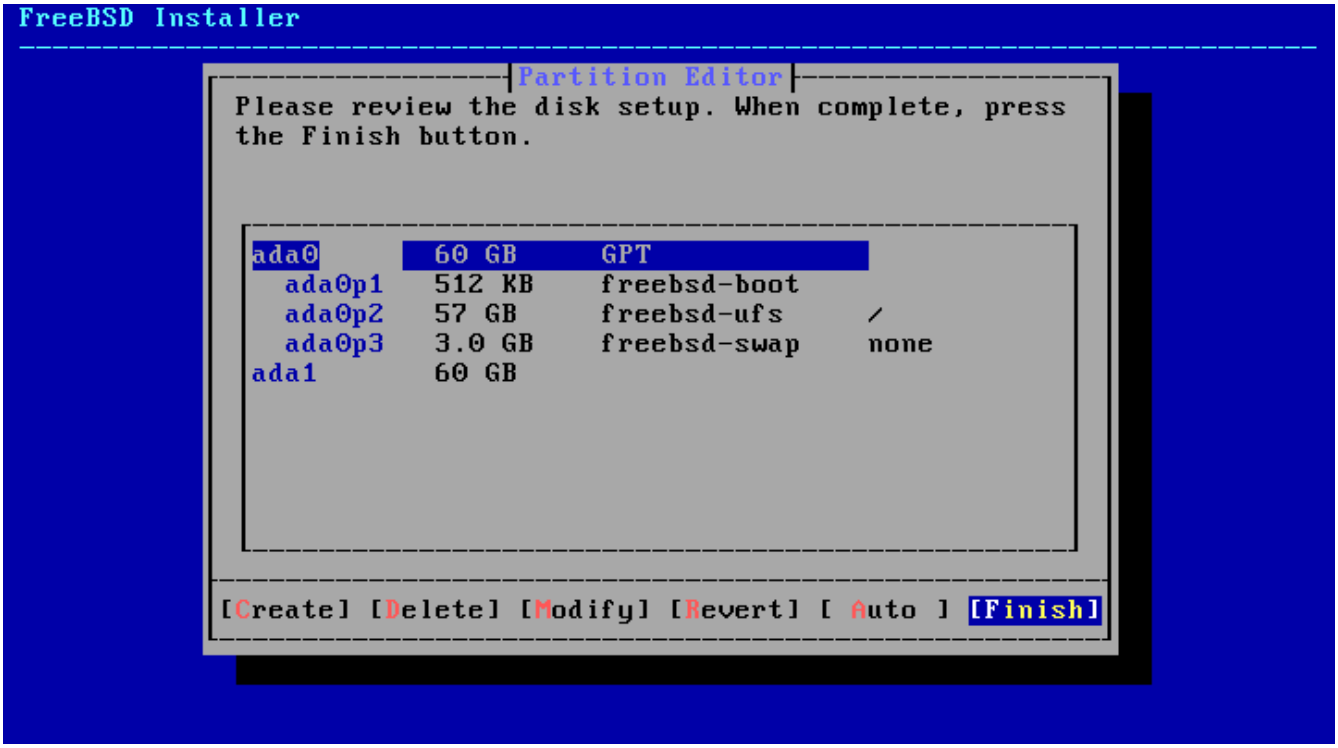


図 *15.* 作成されたパーティションの確認

ディスクを一度設定すると、次のメニューは、選択したハードドライブをフォーマットする前に、設定を変更できる最後のチャンスです。 もし変更が必要であれば、 **[Back]** を選択してメインのパーティションエディタまで戻ってください。 **[Revert & Exit]** を選択すると、ハードドライブへの変更なしにインストールを終了します。
インストールプロセスを開始するには、 **[Commit]** を選択してしてください。

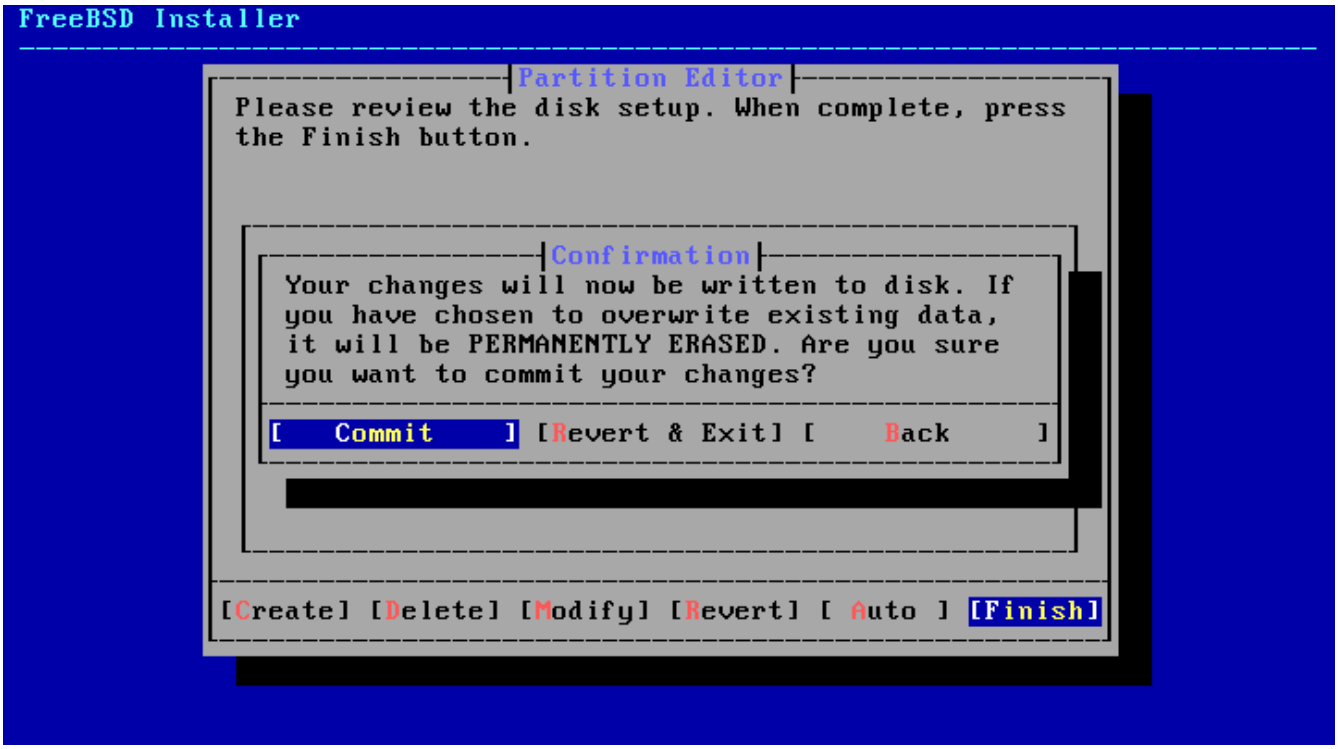


図 *16.* 最後の確認

配布ファイルのダウンロード に進んで、インストールプロセスを続けてください。

2.6.3. Manual によるパーティションの分割

この方法を選択すると、 パーティションエディタが起動します。

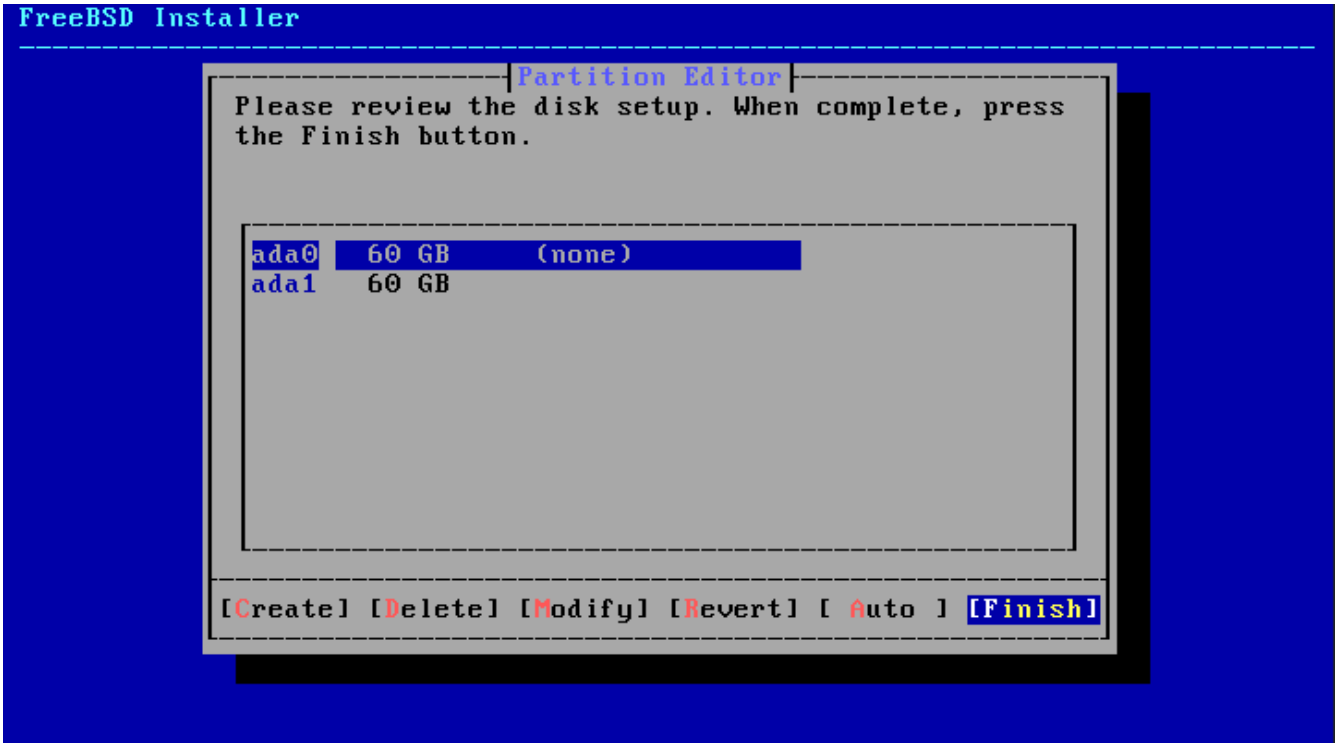


図 *17. Manual* によるパーティションの分割

インストール先のドライブ (この例では ada0) を選び、 **[Create]** を選択すると、利用可能なパーティションスキームの一覧が表示されます。

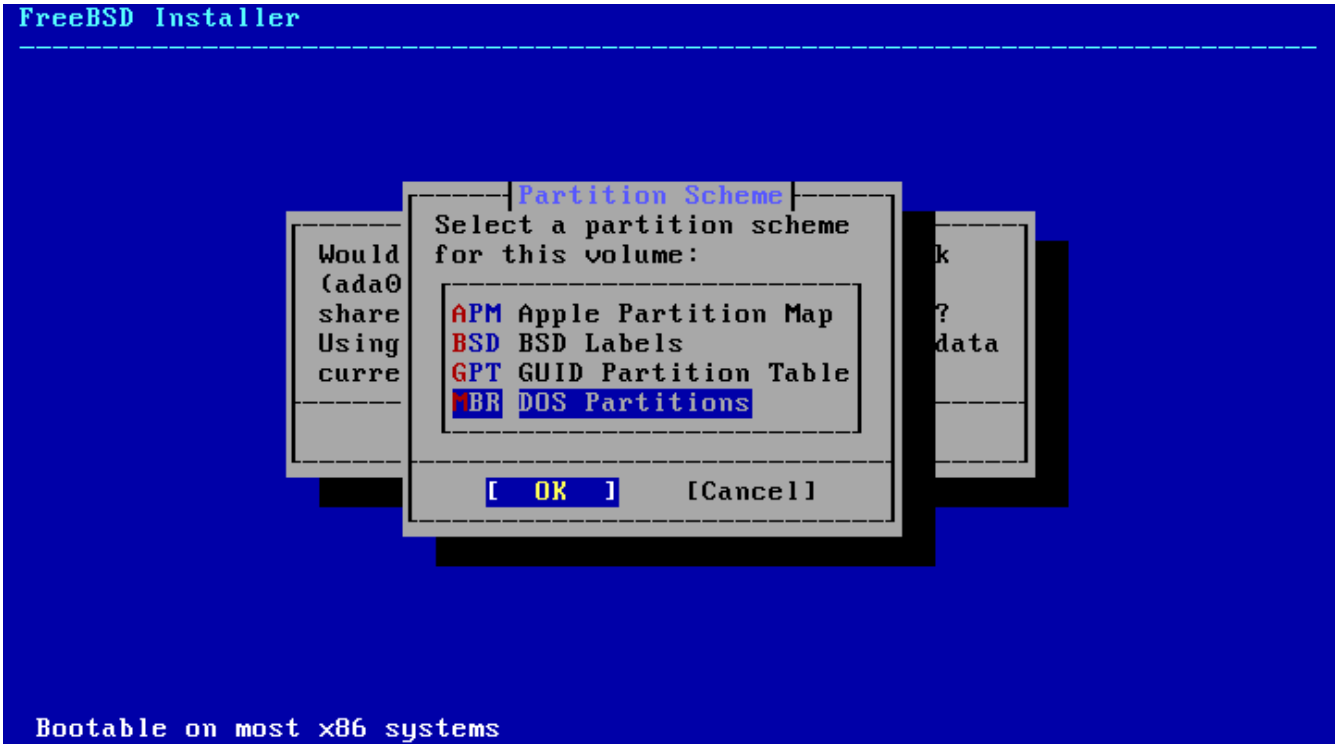


図 *18.* 手動でパーティションを作成する

amd64 コンピュータでは、通常 GPT が最も適切な選択となります。 GPT に対応していないような古いコンピュータでは、 MBR を使う必要があります。

他のパーティションスキームは、使うことがまれであったり、 古いコンピュータで用いられるものです。

表 1. パーティションスキーム

省略形	説明
APM	PowerPC® で使われている Apple Partition Map
BSD	MBR を用いない BSD ラベル。 BSD 以外のディスクユーティリティは認識しないため、しばしば *dangerously dedicated mode* と呼ばれます。
GPT	GUID Partition Table
MBR	Master Boot Record

パーティションスキームを選択して作成した後で、 もう一度 **[Create]** を選択すると、パーティションが作成されます。`Tab` キーを使ってカーソルをフィールド間で移動できます。

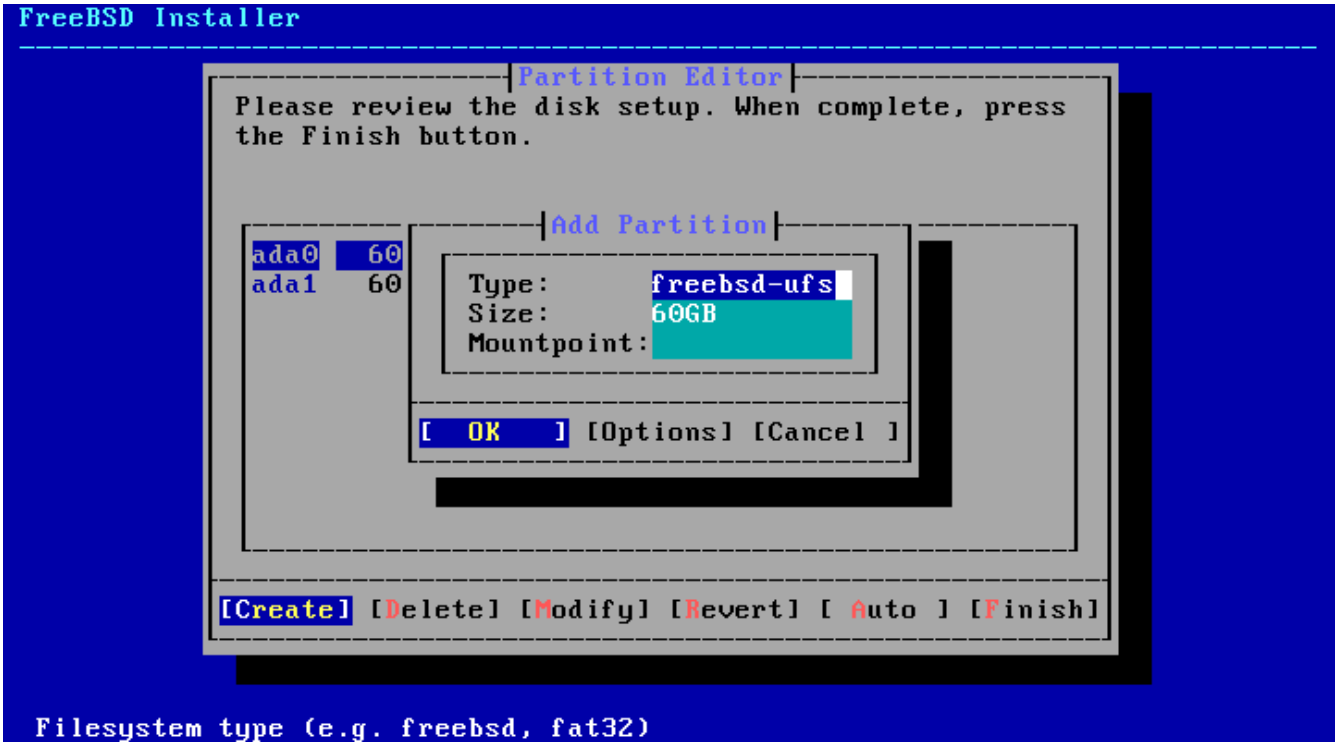


図 19. 手動でパーティションを作成する

標準の FreeBSD GPT のインストールでは、UFS または ZFS を含む少なくとも 3 つのパーティションが使われます。

- `freebsd-boot` または `efi` - FreeBSD ブートコードを含んでいます。
- `freebsd-ufs` - FreeBSD UFS ファイルシステム。
- `freebsd-zfs` - FreeBSD ZFS ファイルシステム。 詳細については、 The Z File System (ZFS) をご覧ください。
- `freebsd-swap` - FreeBSD スワップ空間。

利用可能な GPT パーティションタイプについては、gpart(8) をご覧ください。

複数のファイルシステムのパーティションを作成できます。 人によっては `/`, `/var`, `/tmp` および `/usr` にパーティションを分割する伝統的なレイアウトが好まれます。

十分なメモリを搭載したシステムでは、メモリベースのファイルシステム (tmpfs(5)) を /tmp として後で追加できます。

レイアウトの例が 伝統的なファイルシステムのパーティションの作成 にあります。

Size には、 *K* (キロバイト)、 *M* (メガバイト)、 *G* (ギガバイト) といった通常の省略形を使用出来ます。

セクタを適切に配置することで、 最良のパフォーマンスを得ることができます。 また、パーティションサイズを 4K バイトの偶数倍にすると、 512 バイトまたは 4K バイトのセクタでドライブが配置しやすくなります。 一般的に、 4K の偶数倍の場所からパーティションが開始するように設定する簡単な方法は、 1M または 1G の偶数倍のパーティションサイズを用いることです。 ただし、例外があります。 *freebsd-boot* パーティションは、ブートコードの制限により 512K 以下である必要があります。

ファイルシステムを持つパーティションでは、マウントポイント が必要となります。 1 つの UFS パーティションだけを作成したのであれば、マウントポイントは / となります。

Label は作成したパーティションを認識するための名前です。 ドライブ名や番号は、ドライブが別のコントローラやポートに接続されると変わることがありますが、パーティションラベルは変わりません。 /etc/fstab のようなファイルの中で、ドライブ名やパーティション番号ではなく、ラベルを参照することにより、システムがハードウェアの変更に対して、より寛容になります。 GPT ラベルは、ディスクが接続されると /dev/gpt/ に現れます。 他のパーティションスキームでは別のラベルとなり、/dev/ 以下の異なるディレクトリにラベルが現れます。

名前の衝突を避けるため、 各パーティションには、一意的な名前使ってください。 コンピュータ名、使用、位置情報を表す単語をラベルに追加できます。 たとえば、lab という名前のコンピュータの UFS の root パーティションには、 labroot または rootfs-lab といった名前を使ってください。

例 *1.* 伝統的なファイルシステムのパーティションの作成

伝統的なパーティションレイアウト (/, /var, /tmp および /usr ディレクトリが各パーティションの別のファイルシステム) を作成するには、GPT パーティションスキームを作成し、その後、示されているようにパーティションを作成してください。 示されているパーティションサイズは 20G のディスク用です。 ディスクにより多くの容量があれば、swap または /var パーティションを大きく取ると良いでしょう。 ここで示されているラベルには、example を意味する ex が付けられていますが、実際には上で説明したように、これとは別のユニークなラベルをつけてください。

FreeBSD の gptboot は、 デフォルトでは最初に見つかった UFS パーティションが、 / パーティションであることを前提としています。

パーティションタイプ	サイズ	マウントポイント	ラベル
freebsd-boot	512K		
freebsd-ufs	2G	/	exrootfs

パーティションタイプ	サイズ	マウントポイント	ラベル
freebsd-swap	4G		exswap
freebsd-ufs	2G	/var	exvarfs
freebsd-ufs	1G	/tmp	extmpfs
freebsd-ufs	デフォルト (ディスクの残りのすべての容量)	/usr	exusrfs

カスタムパーティションを作成したら、 **[Finish]** を選択して 配布ファイルのダウンロード に進み、インストールを先に進めてください。

2.6.4. Root-on-ZFS を用いた Guided によるパーティションの作成

このパーティションの分割モードは、 ディスクのすべての領域に対して機能するので、 ディスク上にあるすべての内容が消去されます。 メインの ZFS 設定メニューには、 プールの作成をコントロールする数多くのオプションが用意されています。

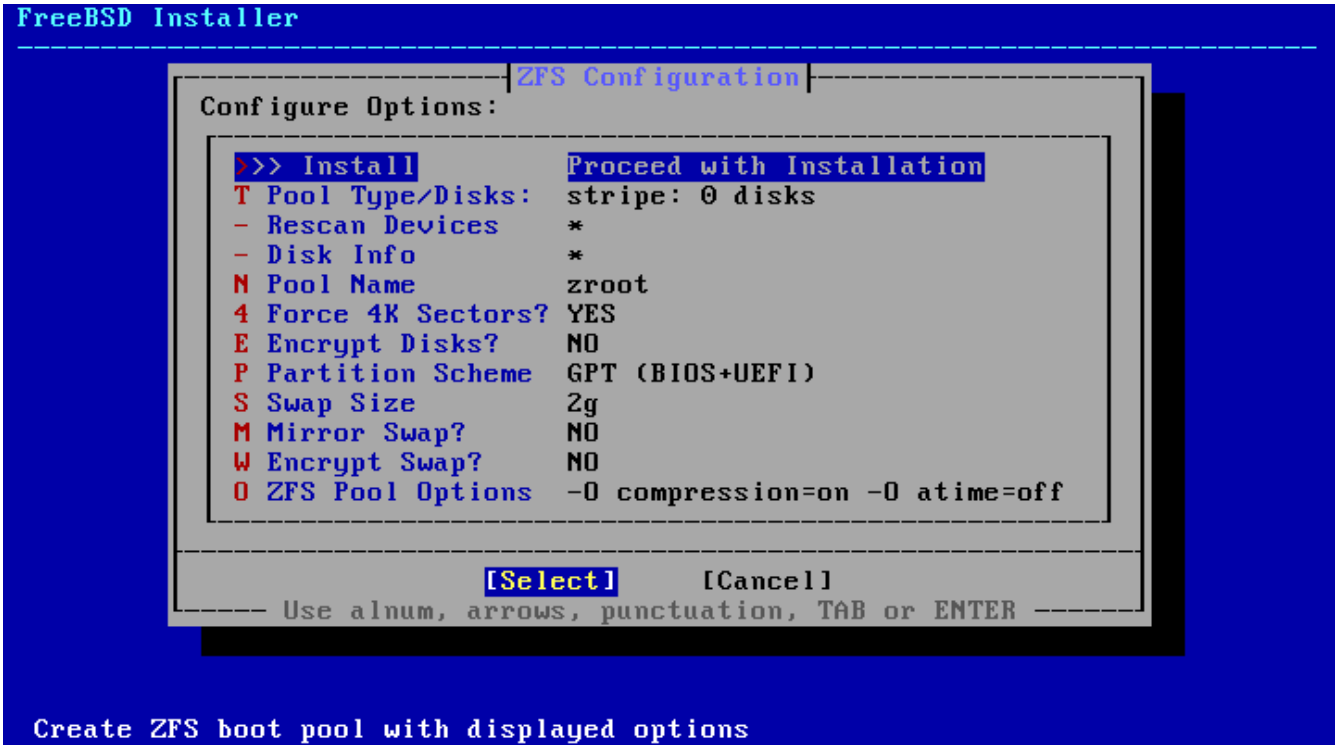


図 *20. ZFS* パーティションメニュー

このメニューのオプションは以下の通りです。

- `Install` - 選択したオプションでインストールを進めます。
- `Pool Type/Disks` - プールを構成する `Pool Type` およびディスクについて設定します。 現時点で ZFS の自動インストーラは、ストライプモードを除き、単一のトップレベルの仮想デバイスの作成のみに対応しています。 より複雑なプールを作成するには、 シェルモードによるパーティションの作成 で説明されている方法で作成してください。
- `Rescan Devices` - 利用可能なディスクの一覧を再表示します。
- `Disk Info` - このメニューを使って各ディスクを調べることができます。 パーティションテーブルやそれ以外のデバイスモデルナンバーおよびシリアルナンバーといった情報

も、可能であれば調べることができます。

- Pool Name - pool の名前を設定します。 デフォルトの名前は *zroot* です。
- Force 4K Sectors? - 4K セクタを使用するようにします。 インストーラは、デフォルトで 4K の境界に整列するようにパーティションを自動的に作成し、 ZFS が 4K セクタを使用するようにします。 これは 512 バイトセクタのディスクでも安全で、 512 バイトのディスク上に作成されたプールが将来的に 4K セクタのディスクを追加できるようにしておくことには、 ストレージ容量の追加や壊れたディスクの交換時に恩恵があります。 有効にするか無効にするかを選択して Enter キーを押してください。
- Encrypt Disks? - GELI を使ってディスクを暗号化できます。ディスクの暗号化の詳細については、 geli によるディスクの暗号化 をご覧ください。 Enter キーを押して、暗号化を有効にするか無効にするかを選択してください。
- Partition Scheme - パーティションスキームを選択します。 ほとんどの場合において、GPT が推奨されます。 別のスキームを選択する場合には、 Enter キーを押してください。
- Swap Size - スワップ容量を設定します。
- Mirror Swap? - スワップ領域をディスク間でミラー化するかどうかを設定します。 スワップ領域をミラー化すると、クラッシュダンプを取得できないので注意してください。 Enter キーを押して有効/無効を設定してください。
- Encrypt Swap? - スワップ領域の暗号化について設定します。 これはシステムの起動時に一時キーとともにスワップ領域を暗号化し、再起動時にキーは破棄されます。 Enter キーを押して有効/無効を設定してください。 詳細については、 swap 領域の暗号化 を参照してください。

T を選択して、Pool Type およびプールに対応するディスクを選択してください。

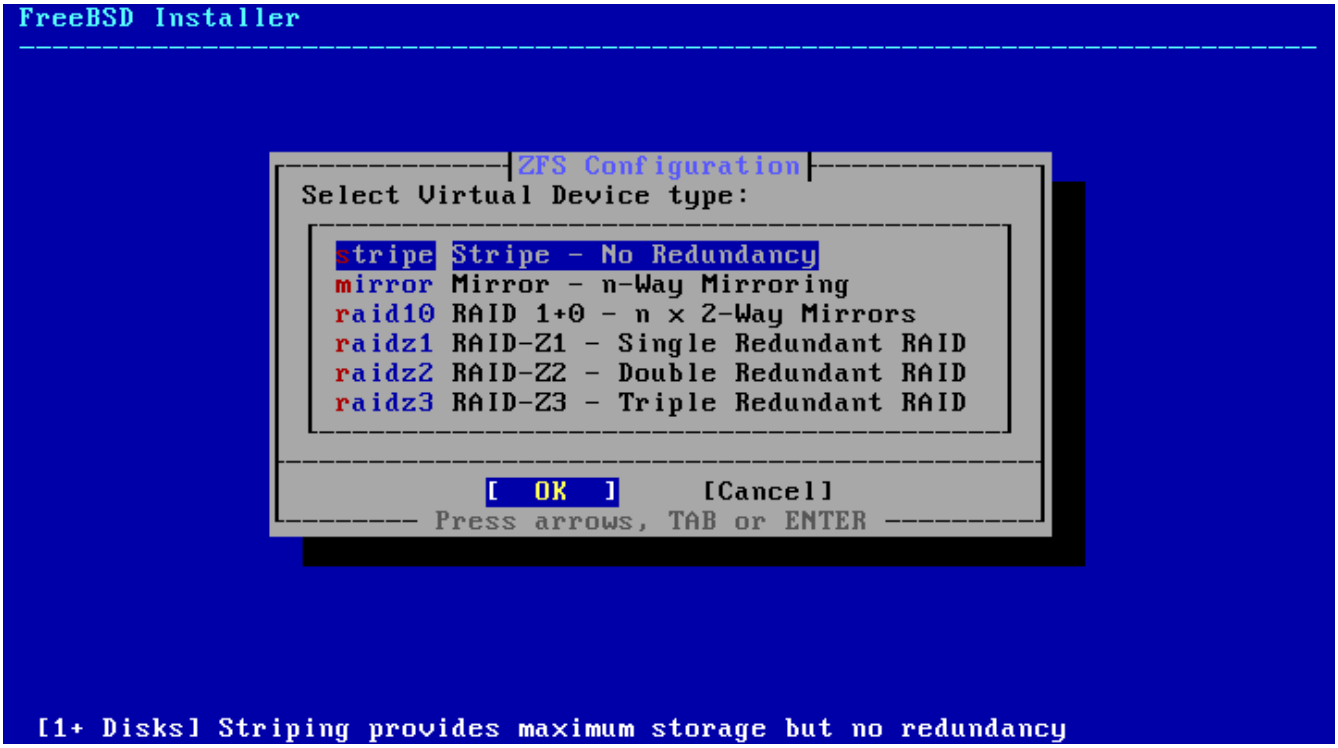


図 *21. ZFS* プールタイプ

このメニューで選択可能な Pool Type は以下の通りです。

- stripe - ストライピングでは、 接続されているすべてのデバイスの最大容量を使用できます。ただし、冗長性はありません。
一つのディスクが壊れるだけでプールにあるデータは失われてしまい、 取り返しがつきません。
- mirror - ミラーリングは各ディスク上にあるすべてのデータの完全なコピーを保存します。ミラーリングでは、並列にすべてのディスクからデータを読むため、
読み込みのパフォーマンスが向上します。 書き込みのパフォーマンスは、データが並列にすべてのディスクに書き込まれるため、遅くなります。 1 つを除くすべてのディスクが壊れることを許容します。 このオプションを選択するには、 少なくとも 2 つのディスクを必要とします。
- raid10 - ストライピングミラー。 最も効率は良いですが、ストレージ容量は少なくなります。偶数のディスクが必要で、 少なくとも 4 つのディスクが必要です。
- raidz1 - シングルパリティの RAID。 1 台のディスクの故障に耐えられます。 少なくとも 3 つのディスクが必要です。
- raidz2 - ダブルパリティの RAID。 同時に 2 台のディスクの故障に耐えられます。 少なくとも 4 つのディスクが必要です。
- raidz3 - トリプルパリティの RAID。 同時に 3 台のディスクの故障に耐えられます。 少なくとも 5 つのディスクが必要です。

Pool Type を選択したら、 利用可能なディスクの一覧が表示されます。その後、プールを構成するディスクを、1 つまたは複数選択してください。十分なディスクが選択されているかどうかについて検証が行われます。
検証に失敗するようであれば、**[<Change Selection>]** を選択して、ディスクの一覧に戻ってください。もしくは、 **[<Back>]** を選択して Pool Type に戻ってください。

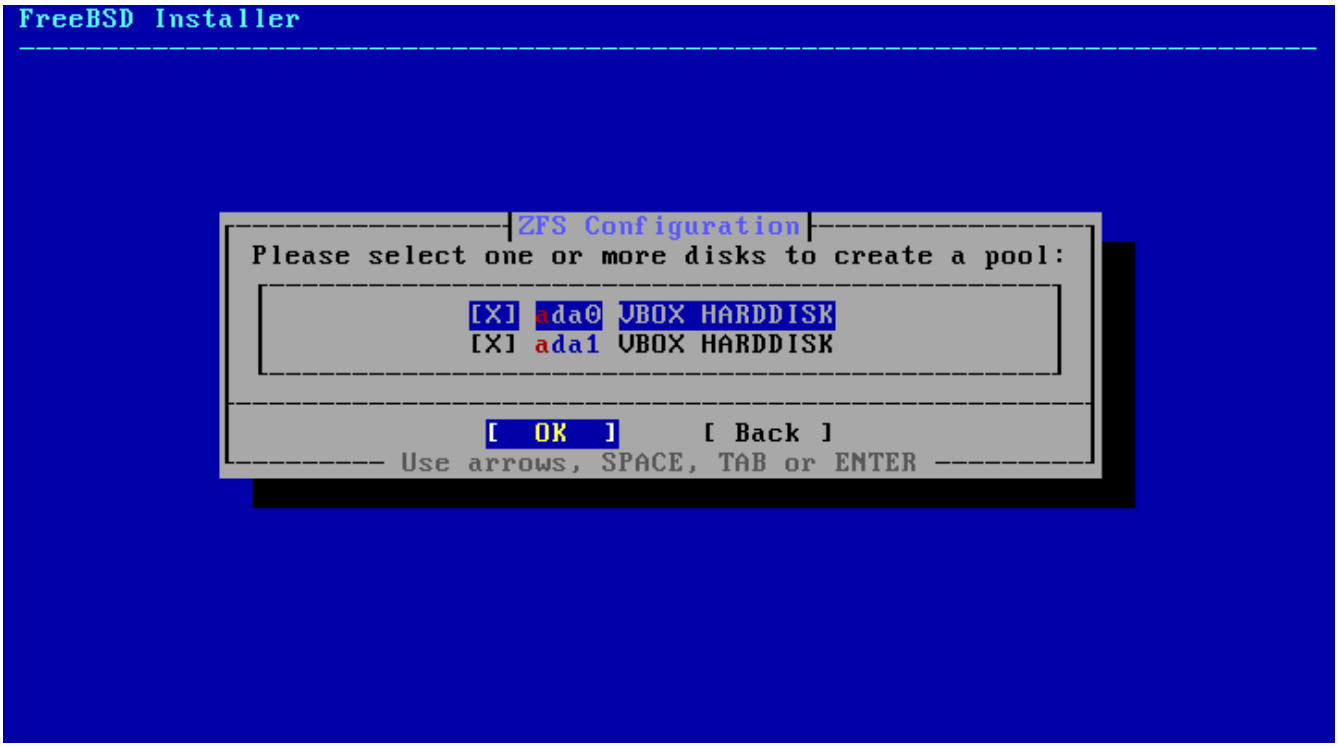


図 *22.* ディスクの選択

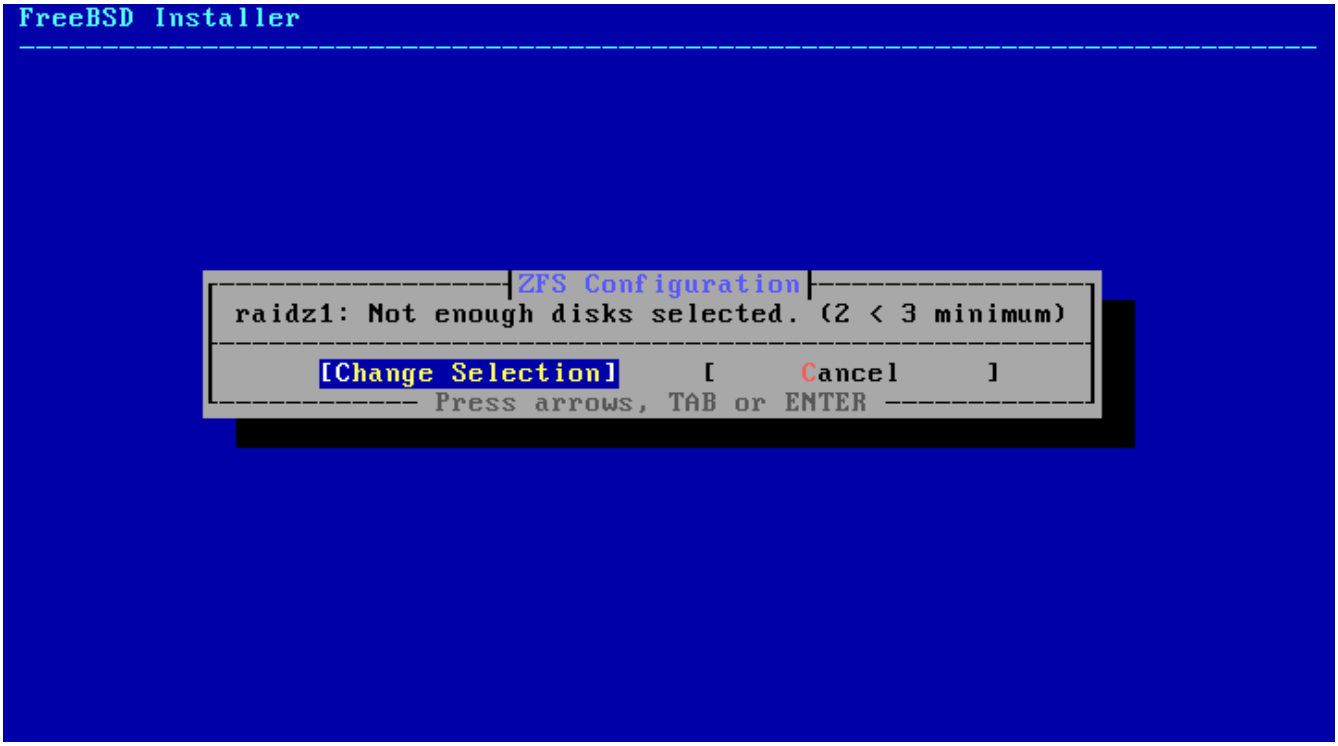


図 *23.* 無効な選択

この一覧の中に抜けているディスクがある時や、
インストーラが立ち上がった後にディスクを接続した場合に、
最新の利用可能なディスクの一覧を見るには、 **[- Rescan Devices]** を選択してください。

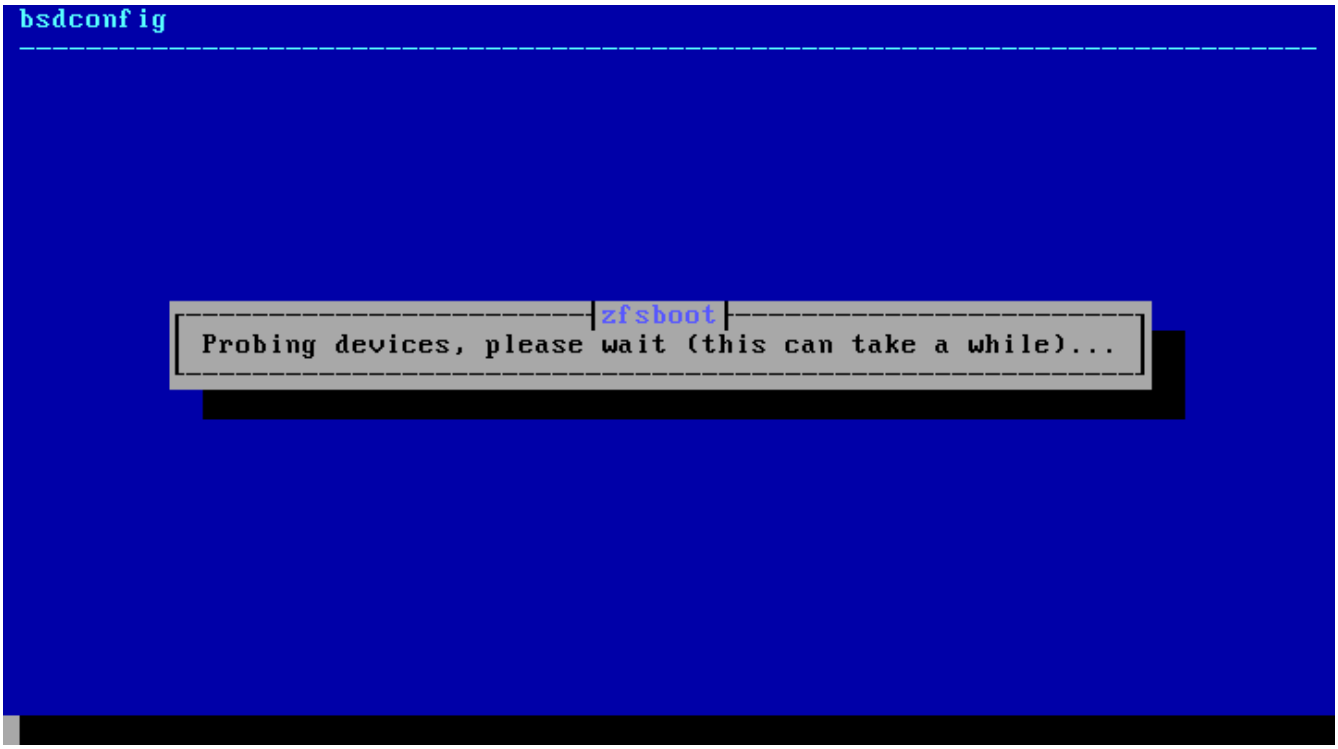


図 *24.* デバイスのリスキャン

アクシデントで間違ったディスクを削除してしまわないように、 **[- Disk Info]** メニュー選択して、
各ディスクのパーティションテーブル、および、
デバイスモデル番号およびシリアル番号などのさまざまな情報を確認してください。

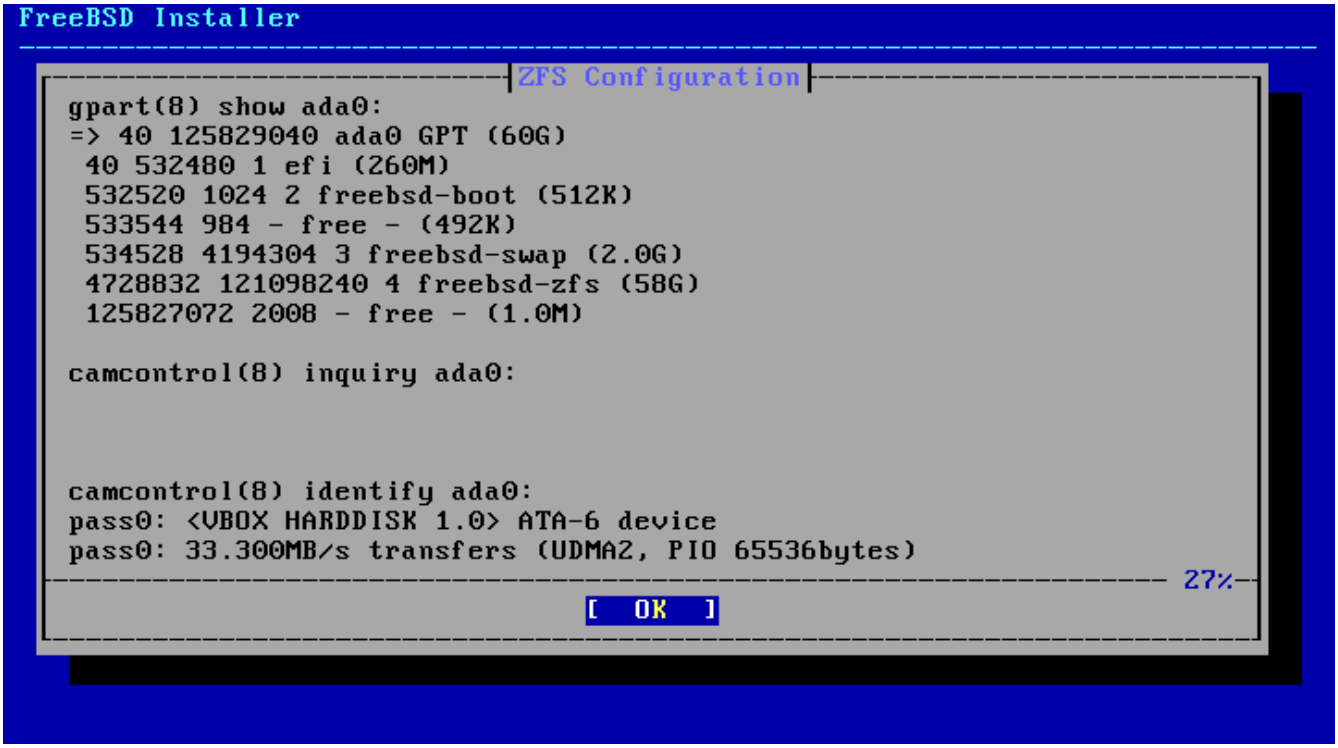


図 *25.* ディスクの解析

N を選択して、 Pool Name を設定してください。 希望する名前を入力後、 **[<OK>]** を選択して確定するか、 **[<Cancel>]** を押して、デフォルト名のままでメインメニューに戻ってください。

図 *26. Pool Name*

S を選択してスワップの容量を設定してください。 必要なスワップ容量を入力し、 **[<OK>]** を押して確定するか、 もしくは **[<Cancel>]** を押して、デフォルトの容量のまま、メインメニューに戻ってください。

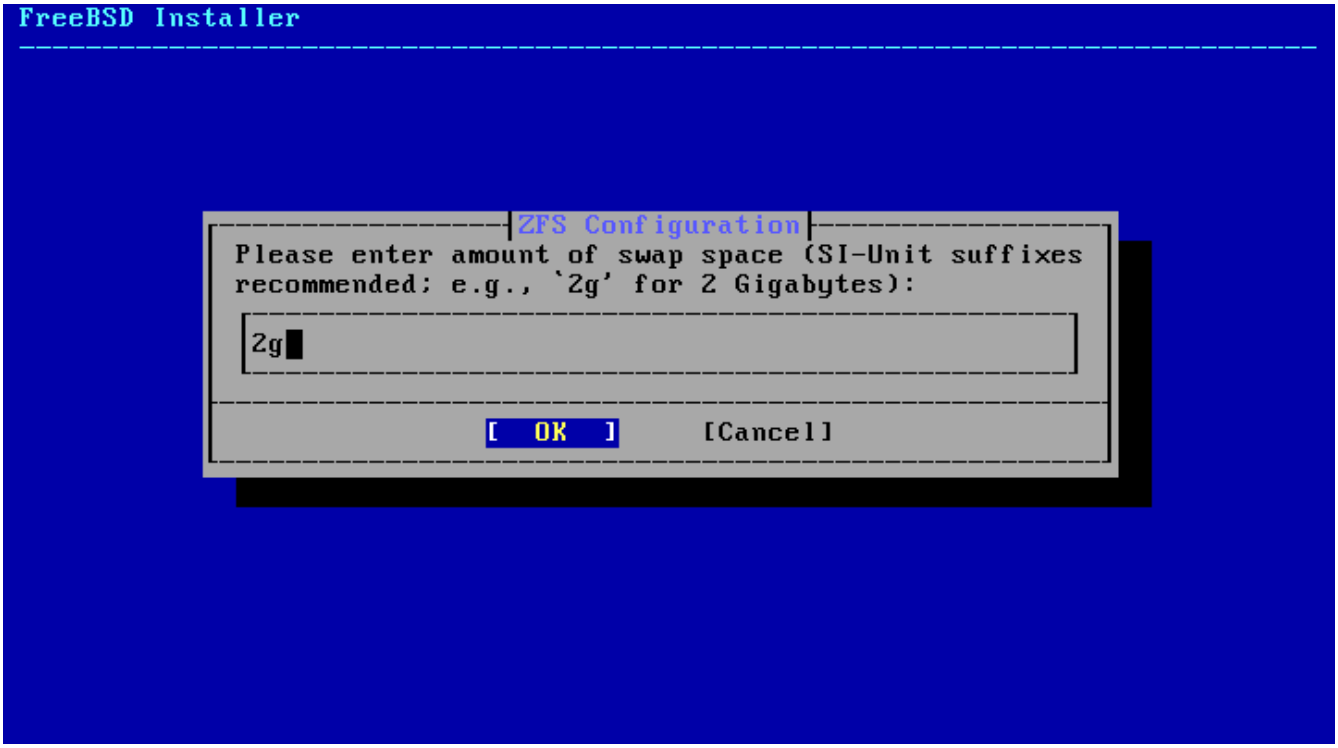


図 *27. Swap* 容量

すべてのオプションに希望する値を設定したら、メニューの上部にある **[>>> Install]** オプションを選択してください。 インストーラは、最終確認として ZFS プールを作成するために選択したドライブの内容が削除されることをキャンセルできる最後の機会を提供してくれます。

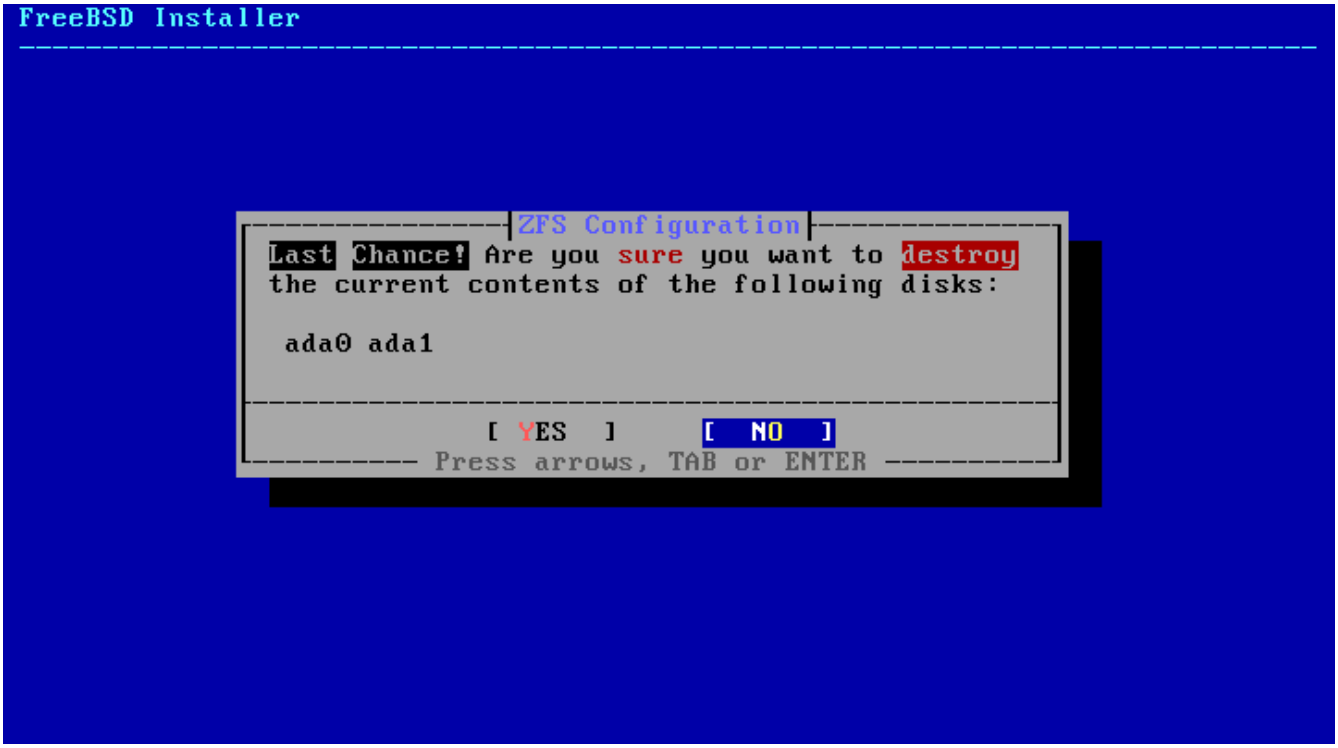


図 *28.* 最終確認

GELI ディスク暗号化を有効にしていたら、ディスクを暗号化するために用いるパスフレーズを 2 度求められます。 その後、暗号の初期化が開始します。

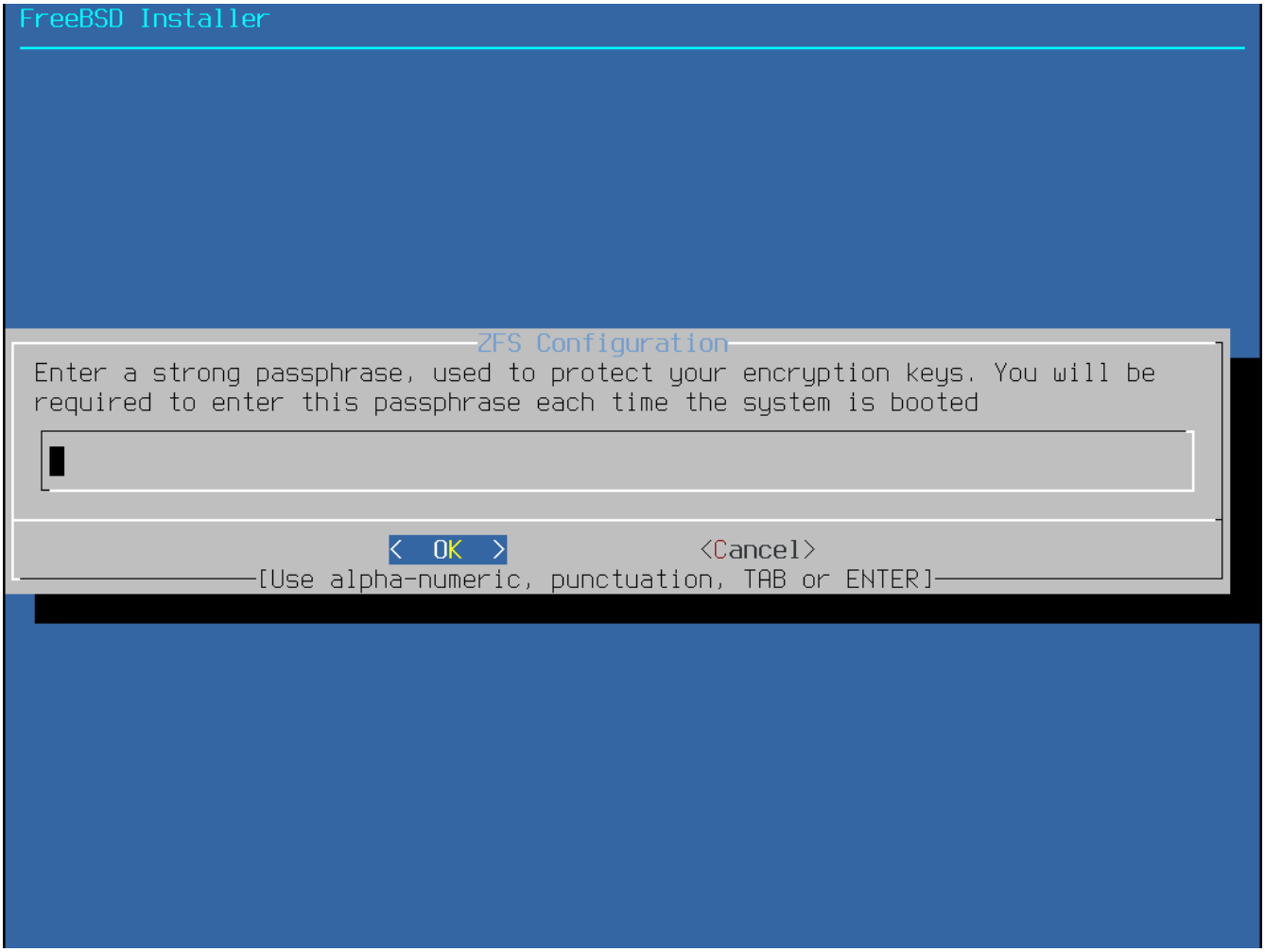


図 *29.* ディスク暗号化パスワード

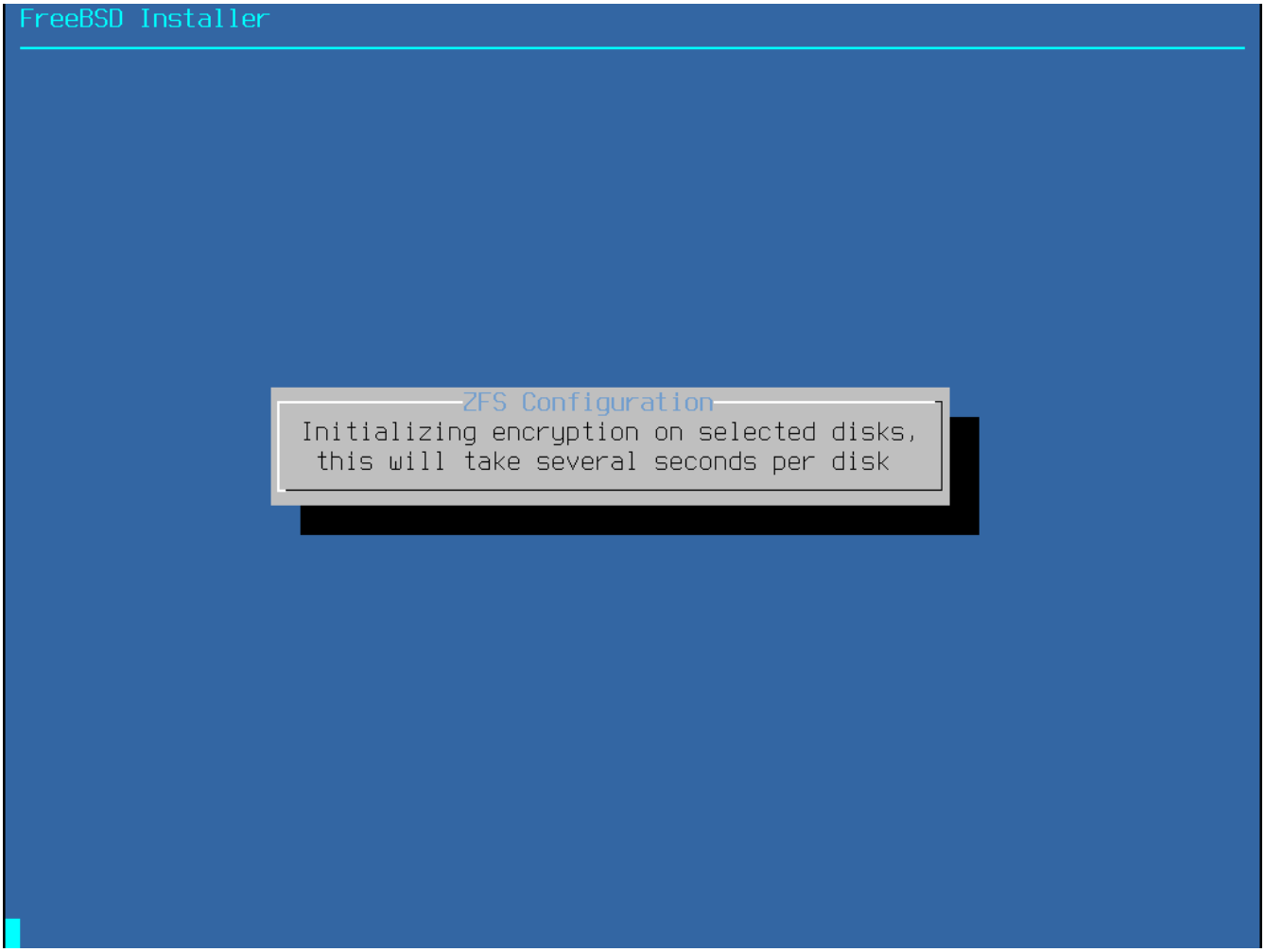


図 *30.* 暗号の初期化

その後のインストールの過程は、通常通りに進みます。　　インストールを進めるには、
配布ファイルのダウンロード に進んでください。

2.6.5. シェルモードによるパーティションの作成

高度なインストールを行う上で、bsdinstall
が提供するパーティション分割のメニューは柔軟性にかけることがあります。
手動でドライブの分割、ファイルシステムの作成、`/tmp/bsdinstall_etc/fstab`　の作成、そして　`/mnt`
以下へのファイルシステムのマウントを行うには、パーティションメニューで　　**[Shell]**
オプションを選択してください。　　このオプションは高度な技術を持つユーザ向けです。
以上を実行したら、`exit` を実行して bsdinstall に戻り、インストールを続けてください。

2.7. 配布ファイルのダウンロード

インストールにかかる時間は、どのディストリビューションを選んだか、
どのインストールメディアを使ったか、　　そしてコンピュータの速度にも依存します。
進行状況を表すメッセージが逐次表示されます。

まず最初に、インストーラは選択されているディスクをフォーマットし、
パーティションを初期化します。　`bootonly`　`media`　または　`mini`　`memstick`
メディアを用いたインストールでは、 選択されたコンポーネントがダウンロードされます。

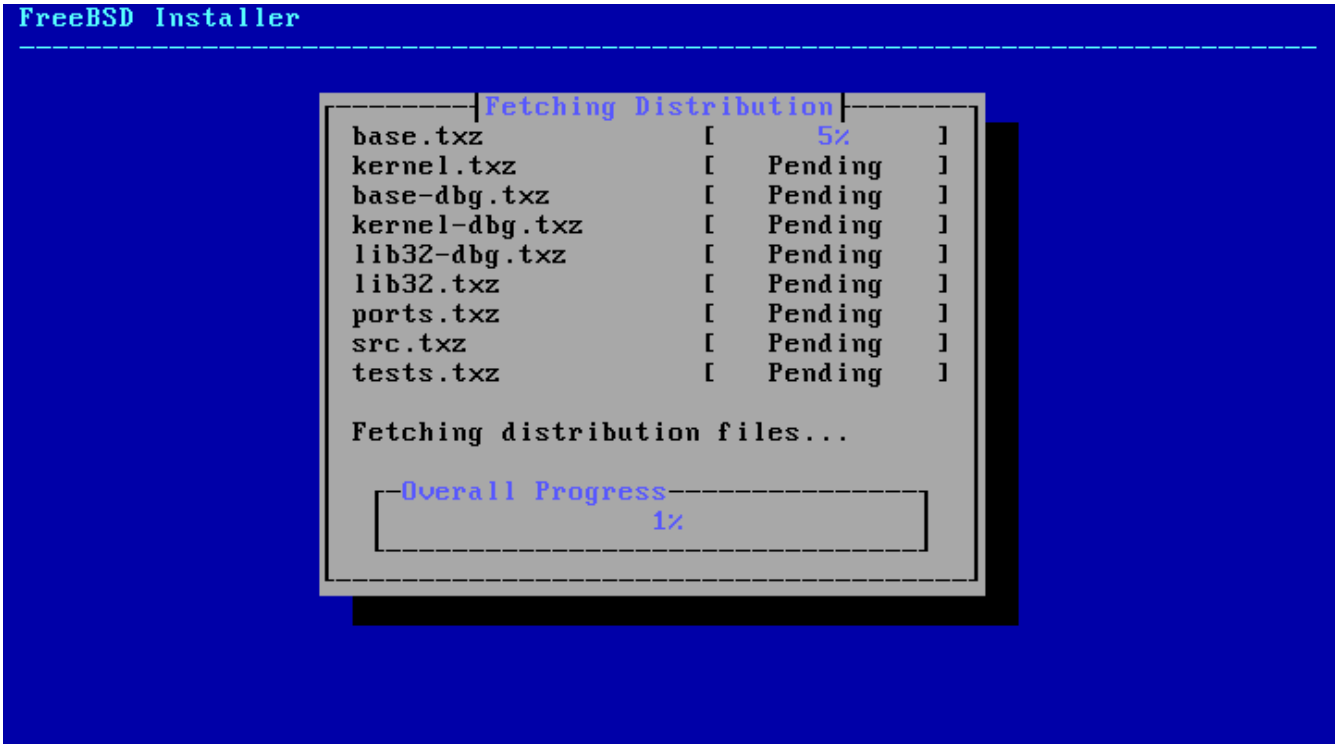


図 *31.* 配布ファイルのダウンロード

次に、ダウンロードの際にエラーが含まれなかったか、
インストールメディアからの読み取り中に読み間違いが起きなかったかどうか等、
配布ファイルの完全性の検証が行われます。

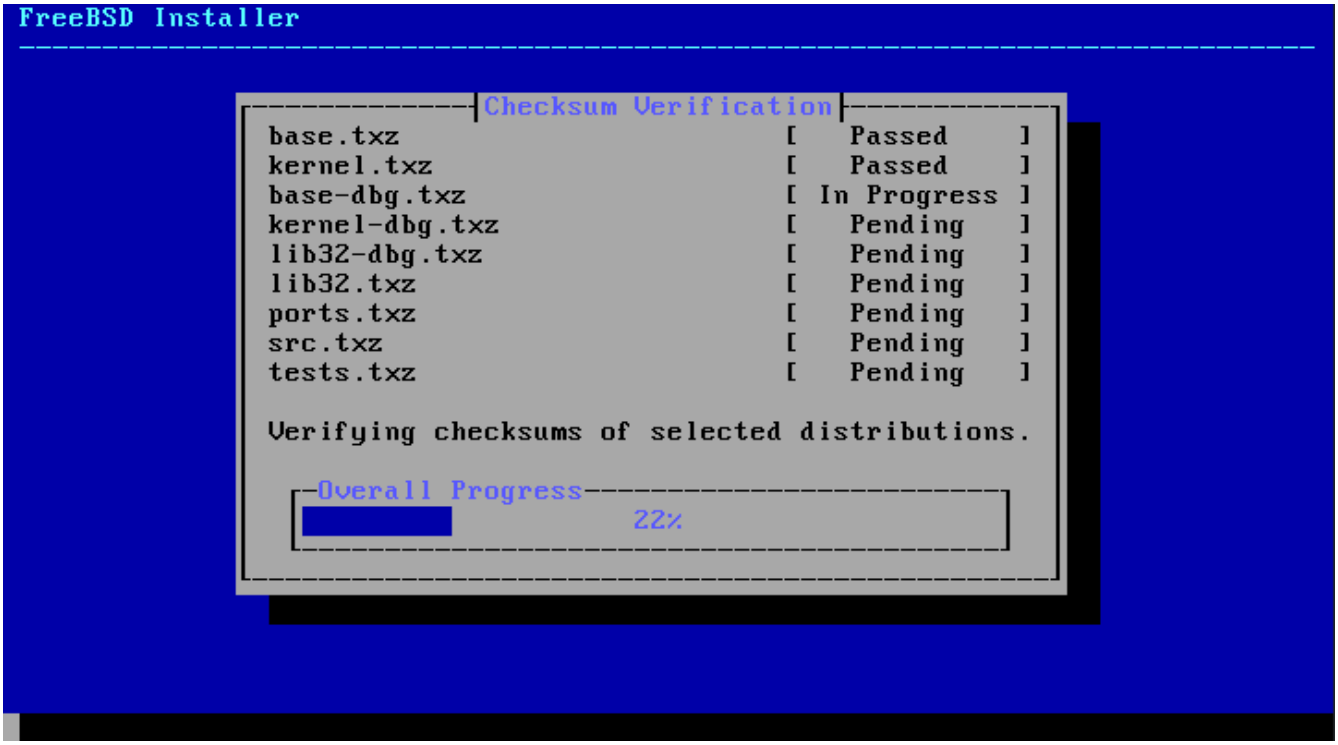


図 *32.* 配布ファイルの検証

最後に、検証された配布ファイルがディスクへ展開されます。

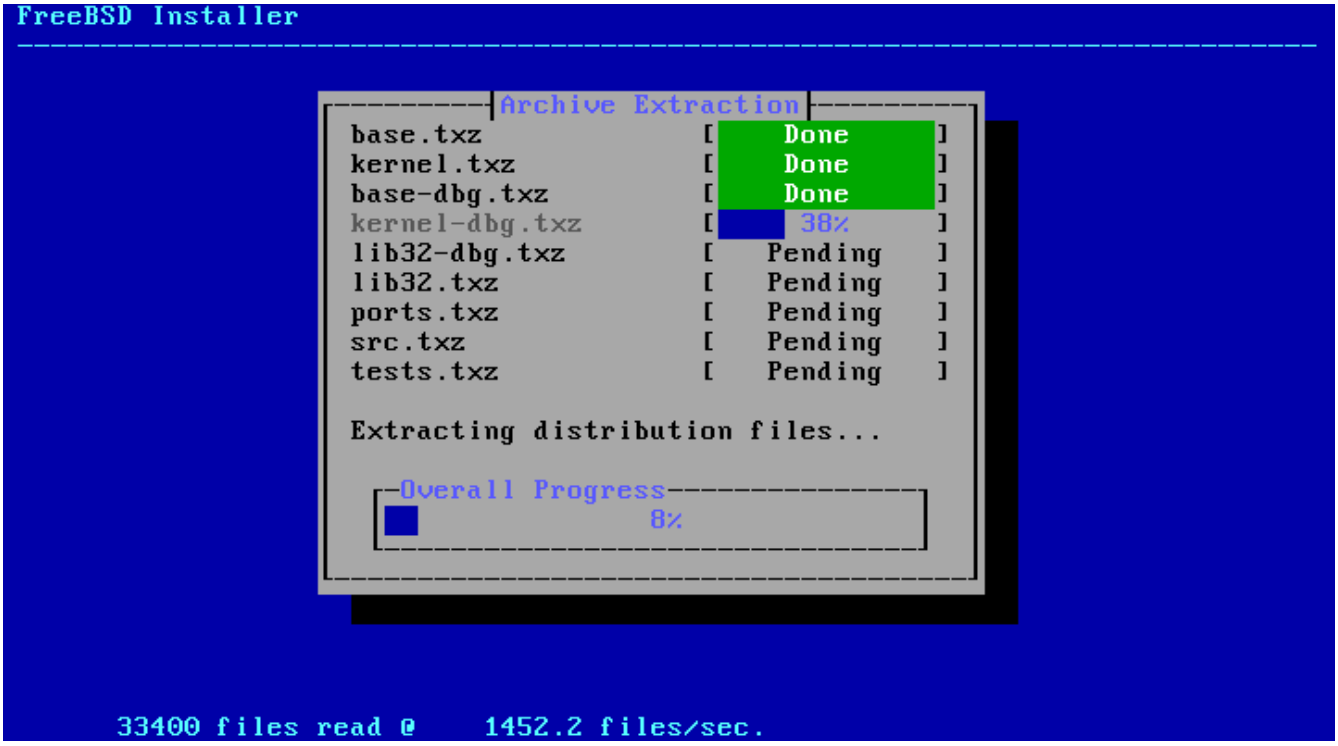


図 *33.* 配布ファイルの展開

必要な配布ファイルがすべて展開されると、 bsdinstall は、 インストール後の設定画面を表示します。
利用可能なインストール後のオプションについては次の章で説明します。

2.8. ネットワークインターフェース、アカウント、タイムゾーン、 サービスおよびセキュリティオプションの設定

2.8.1. `root` パスワードの設定

最初に `root` のパスワードを設定する必要があります。
パスワードを入力している際には、入力している文字は画面に表示されません。
入力ミスを防ぐため、パスワードは 2 回入力する必要があります。

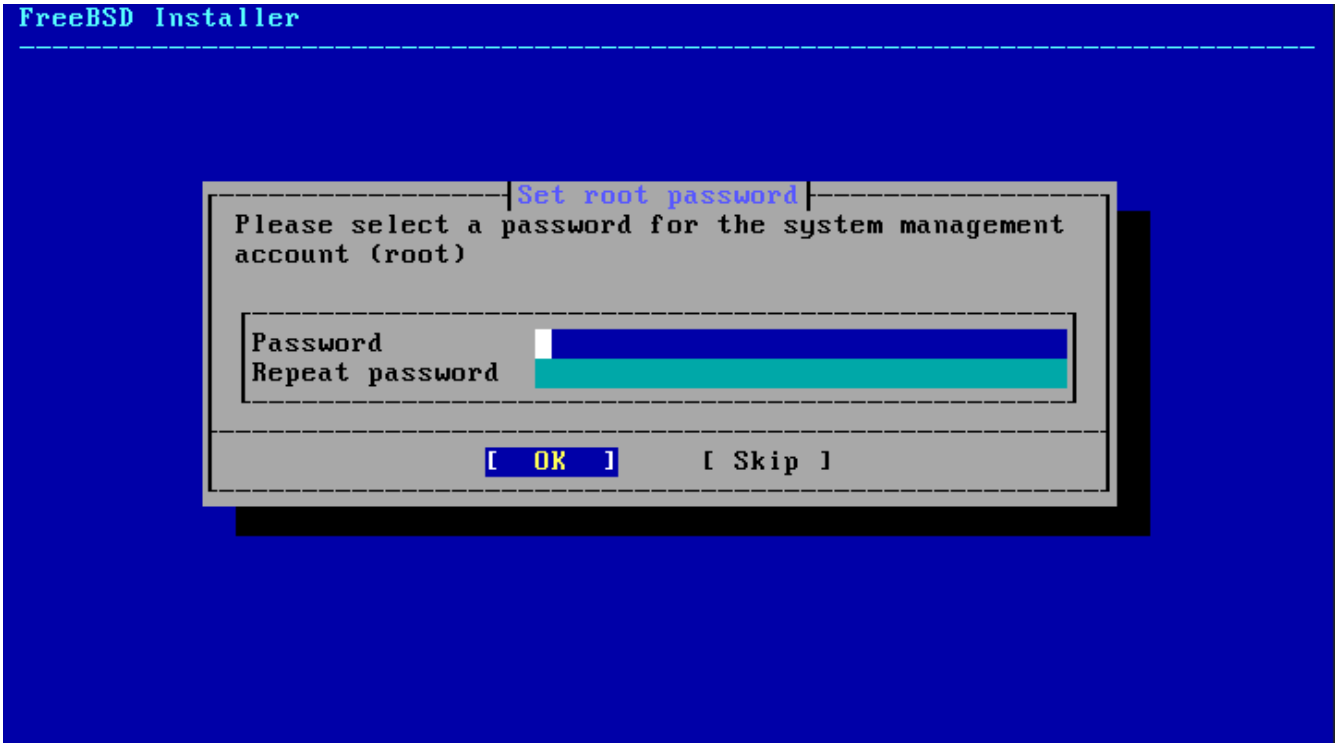


図 *34.* root パスワードの設定

2.8.2. ネットワークインタフェースの設定

次に、コンピュータが認識したすべてのネットワークインタフェースが表示されます。
設定するネットワークインタフェースを選んでください。

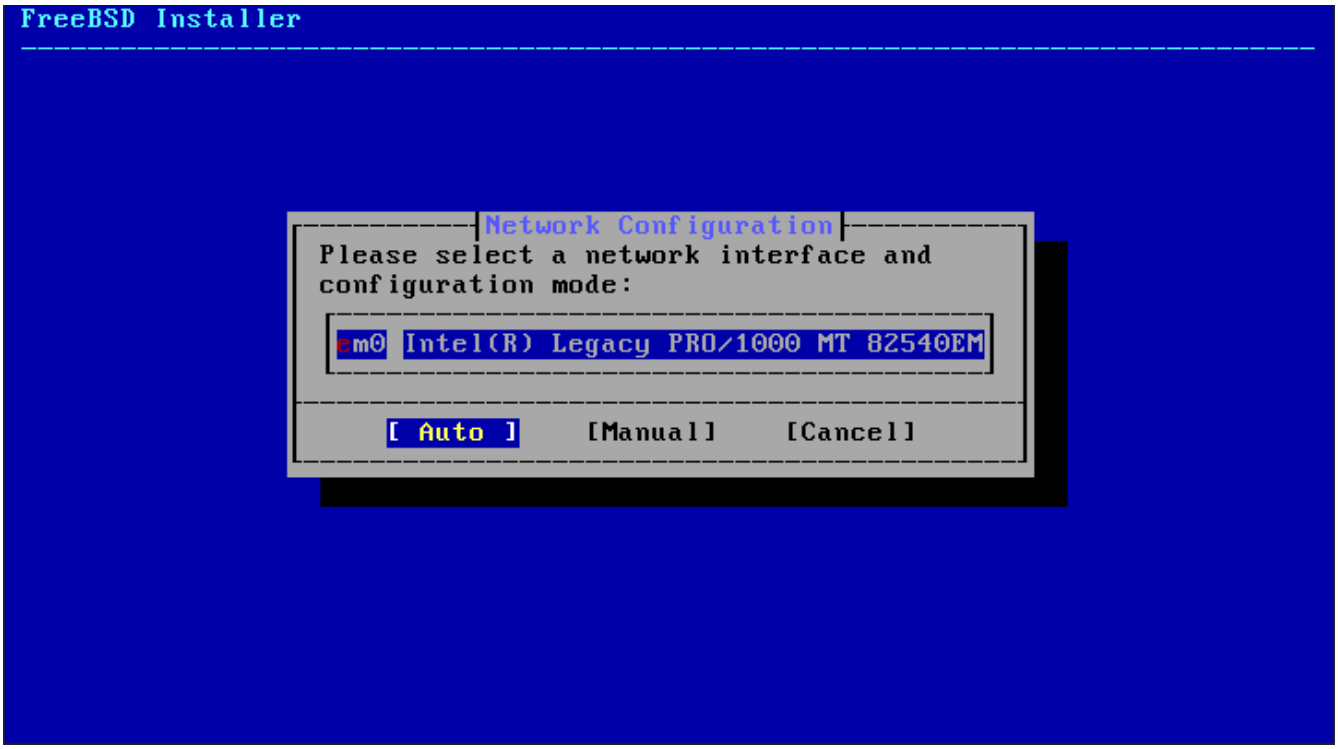


図 *35.* イーサネットインタフェースの選択

イーサネットインタフェースを選択すると、IPv4 ネットワークの選択
で表示されるメニューが表示されます。
ワイヤレスネットワークを選択すると、システムはワイヤレスアクセスポイントをスキャンします。

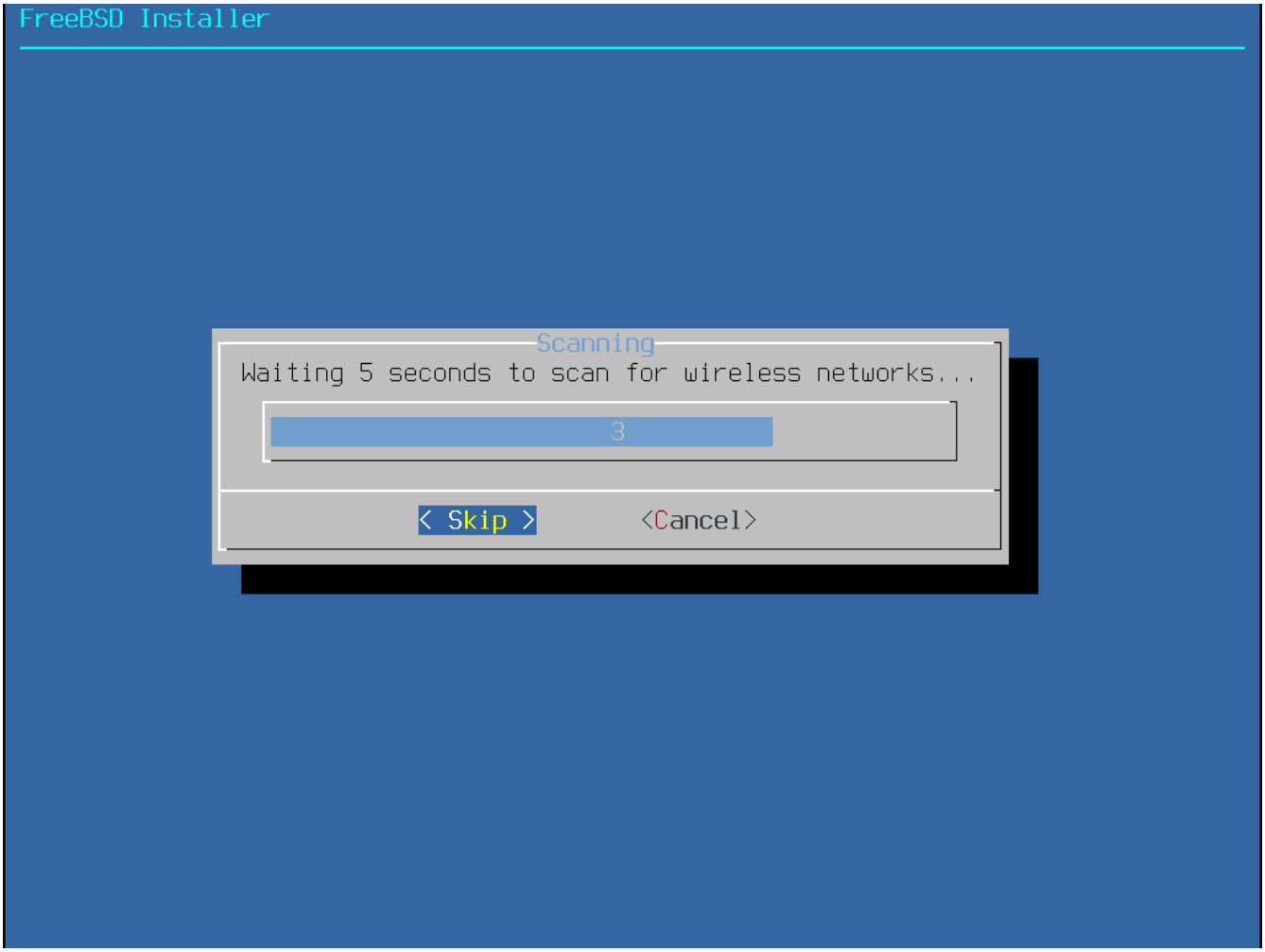


図 *36.* ワイヤレスアクセスポイントのスキャン

ワイヤレスネットワークは Service Set Identifier (SSID) によって識別されます。 SSID は、それぞれのネットワークに与えられる、短く、一意的な名前です。 スキャンで見つかった SSID は、そのネットワークで利用できる暗号化のタイプの説明とともに一覧で表示されます。 もし、期待した SSID が一覧に表示されていなければ、**[Rescan]** を選択してもう一度スキャンしてください。
それでもなお期待したネットワークが表示されなければ、接続のためのアンテナに問題がないかを確認したり、コンピュータをアクセスポイントの近くに移動してみてください。
その後もう一度スキャンしてください。

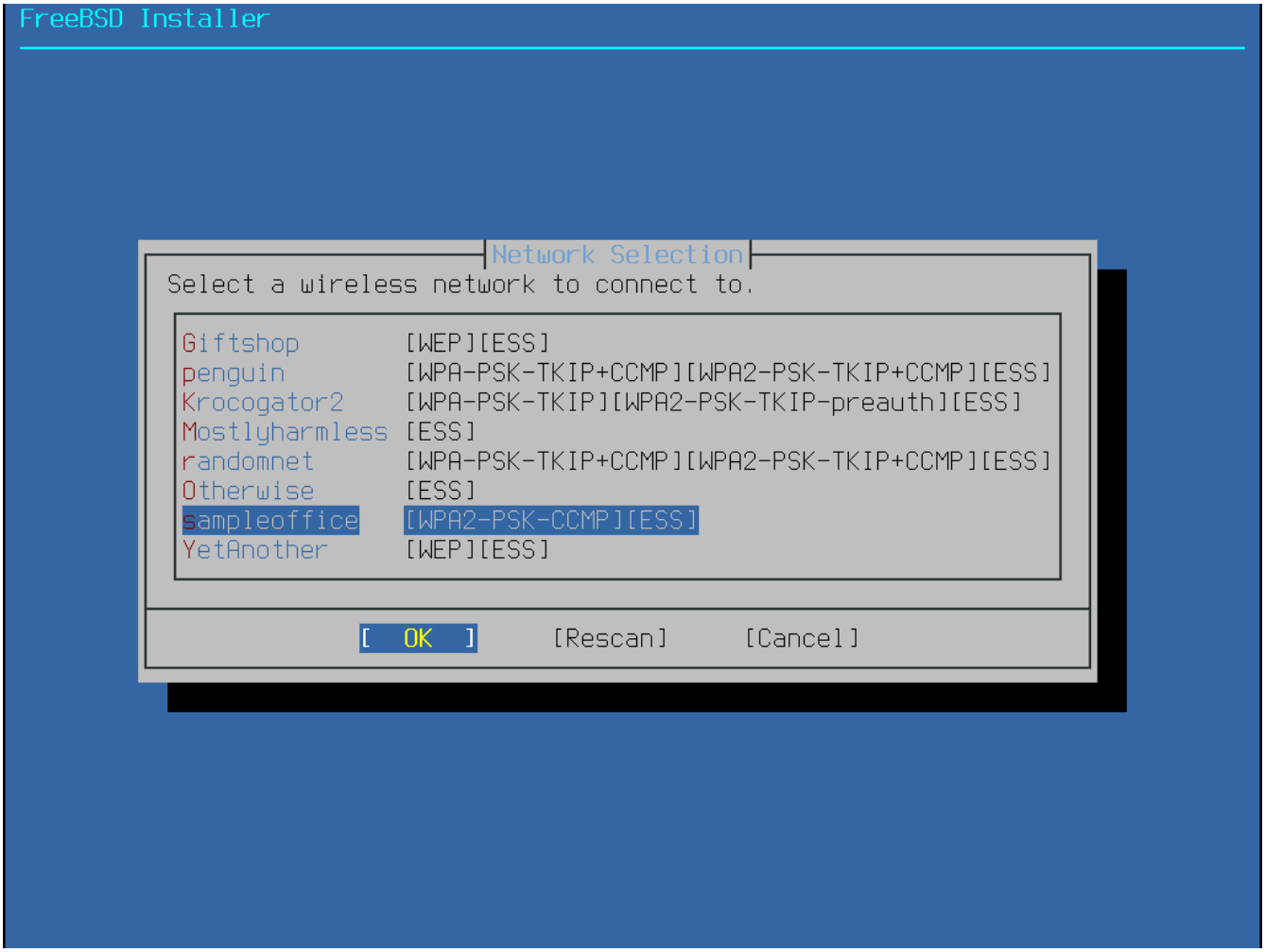


図 *37.* ワイヤレスネットワークの選択

次に、ワイヤレスネットワークに接続するための暗号情報を入力してください。 WEP のような古い暗号の安全性は低いので、WPA2 暗号が強く推奨されます。 WPA2 を使用してるネットワークでは、Pre-Shared Key (PSK) と呼ばれるパスワードを入力してください。セキュリティ上の観点から、入力ボックスに入力した文字はアスタリスクで表示されます。

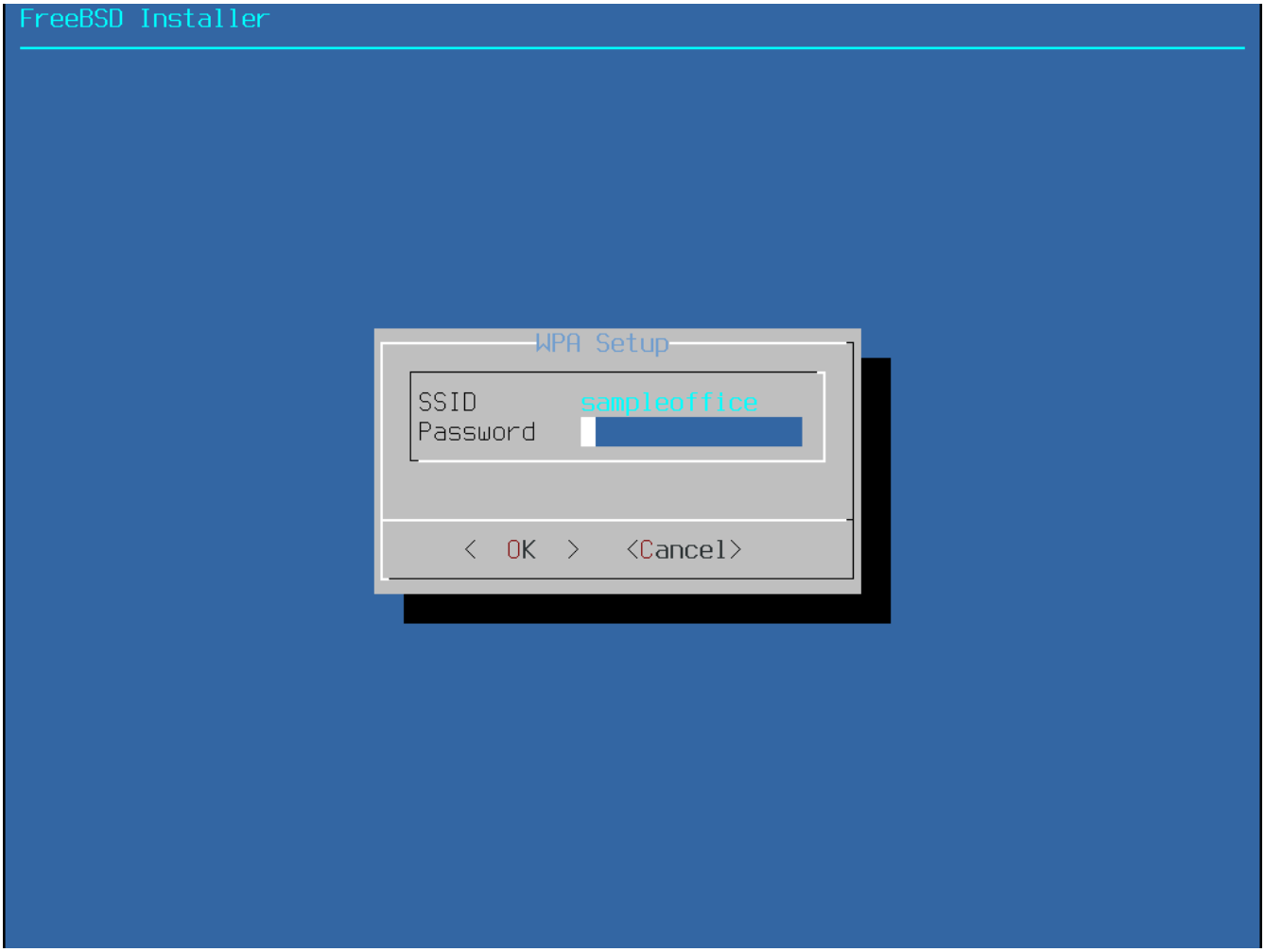


図 *38. WPA2* のセットアップ

次に、イーサネットもしくはワイヤレスインタフェースに対して、IPv4
を設定するかどうかを選択します。

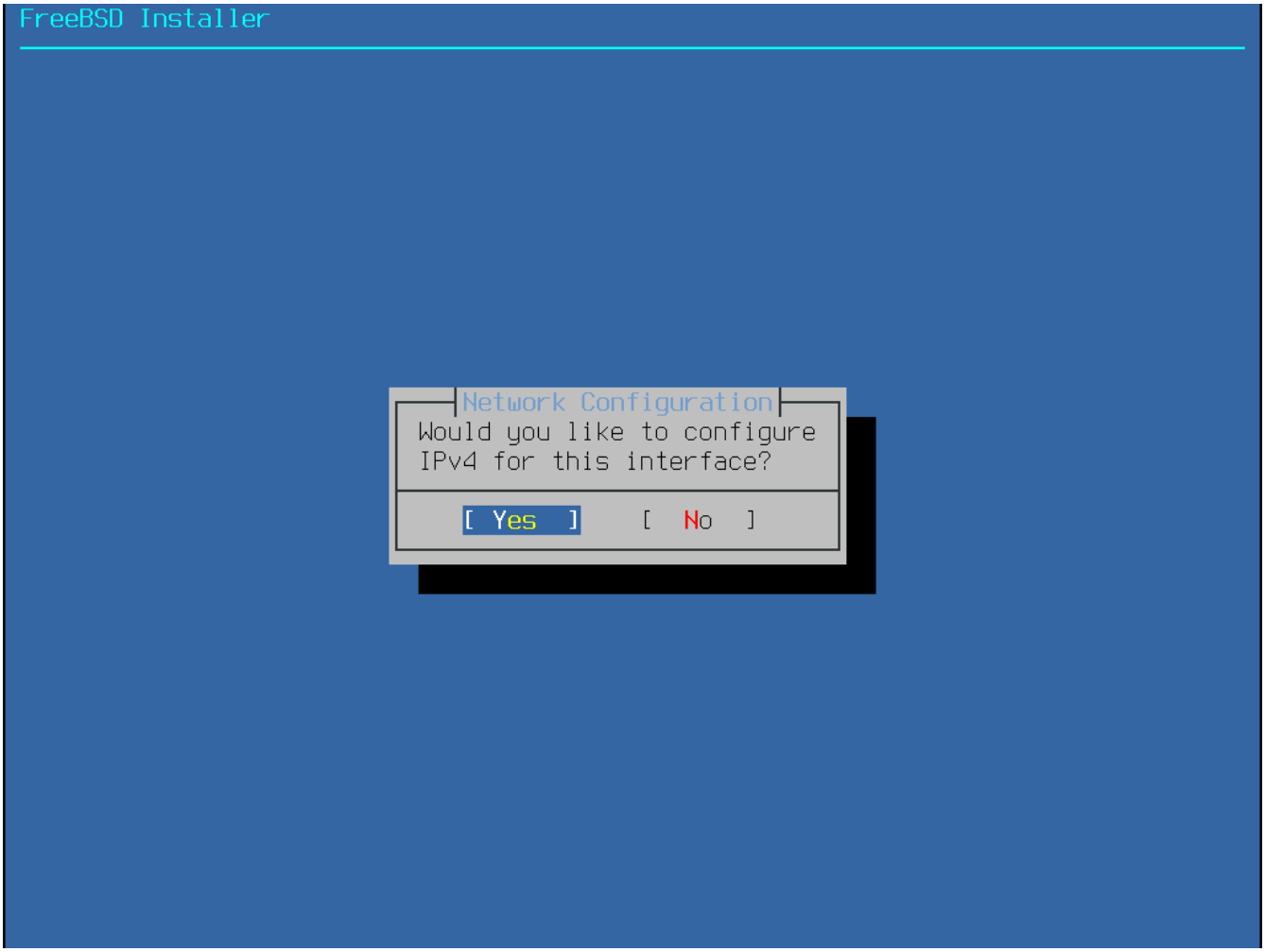


図 *39. IPv4* ネットワークの選択

IPv4 の設定方法は 2 通りあります。*DHCP* はネットワークインタフェースを自動的に適切に設定する方法で、DHCP サーバのあるネットワークでは使用すべきです。DHCP を利用できない環境では、静的な設定として、ネットワークのアドレス情報を手動で入力する必要があります。

適当なネットワーク情報を入力しても動かないので、DHCP サーバが利用できなのであれば、ネットワーク管理者またはサービスプロバイダから必要となるネットワーク情報 に示されている情報を入手してください。

DHCP サーバを利用できるのであれば、次のメニューで **[Yes]** を選択して、ネットワークインタフェースの設定を自動的に行ってください。DHCP サーバを検索し、システムに対するアドレス情報を入手する間、インストーラは少しの間停止しているように表示されます。

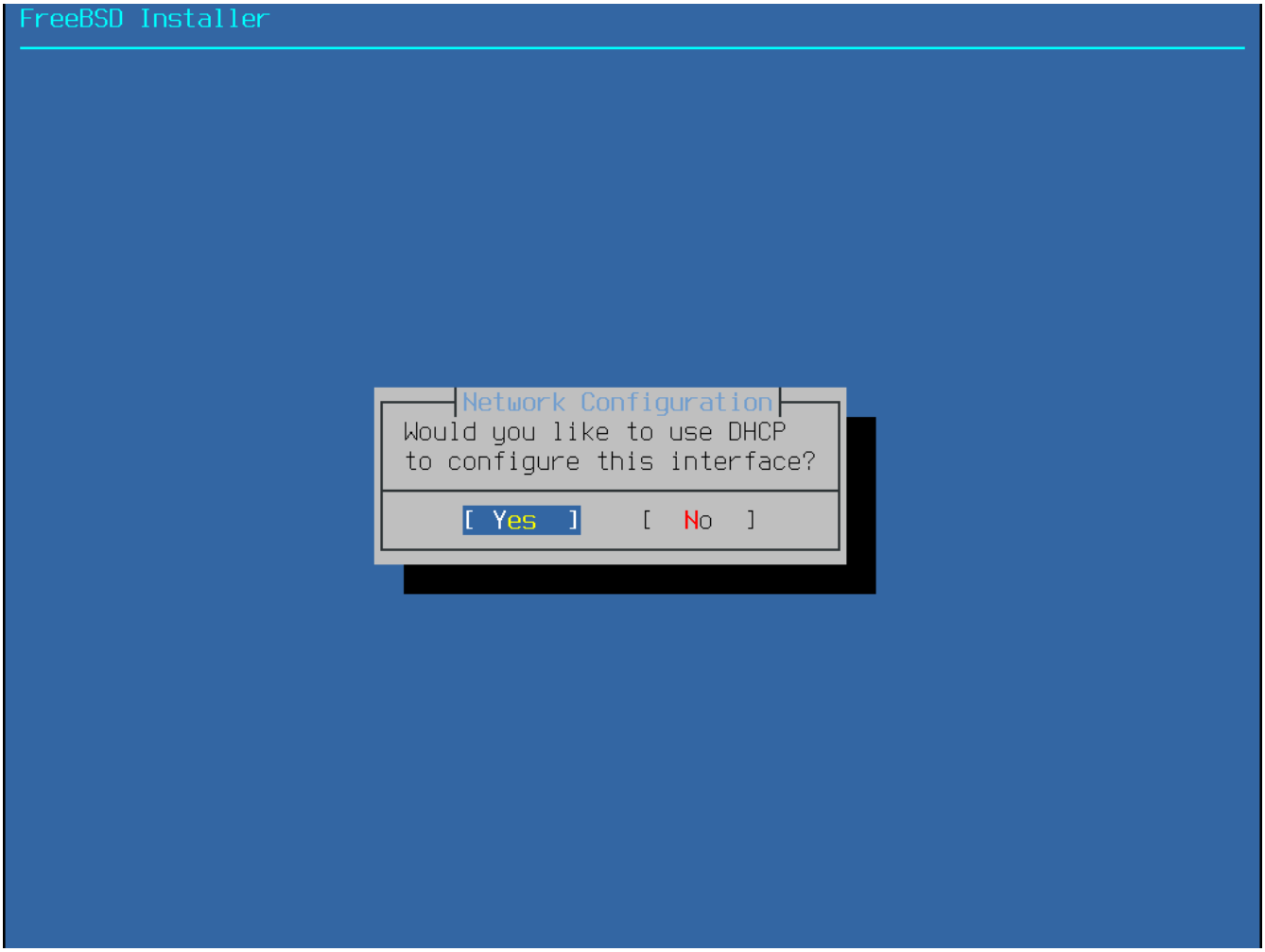


図 *40. IPv4 DHCP* 設定の選択

DHCP サーバを利用できない環境では、**[No]** を選択し、新しく表示されるメニューにおいて以下のようなアドレス情報を入力してください。

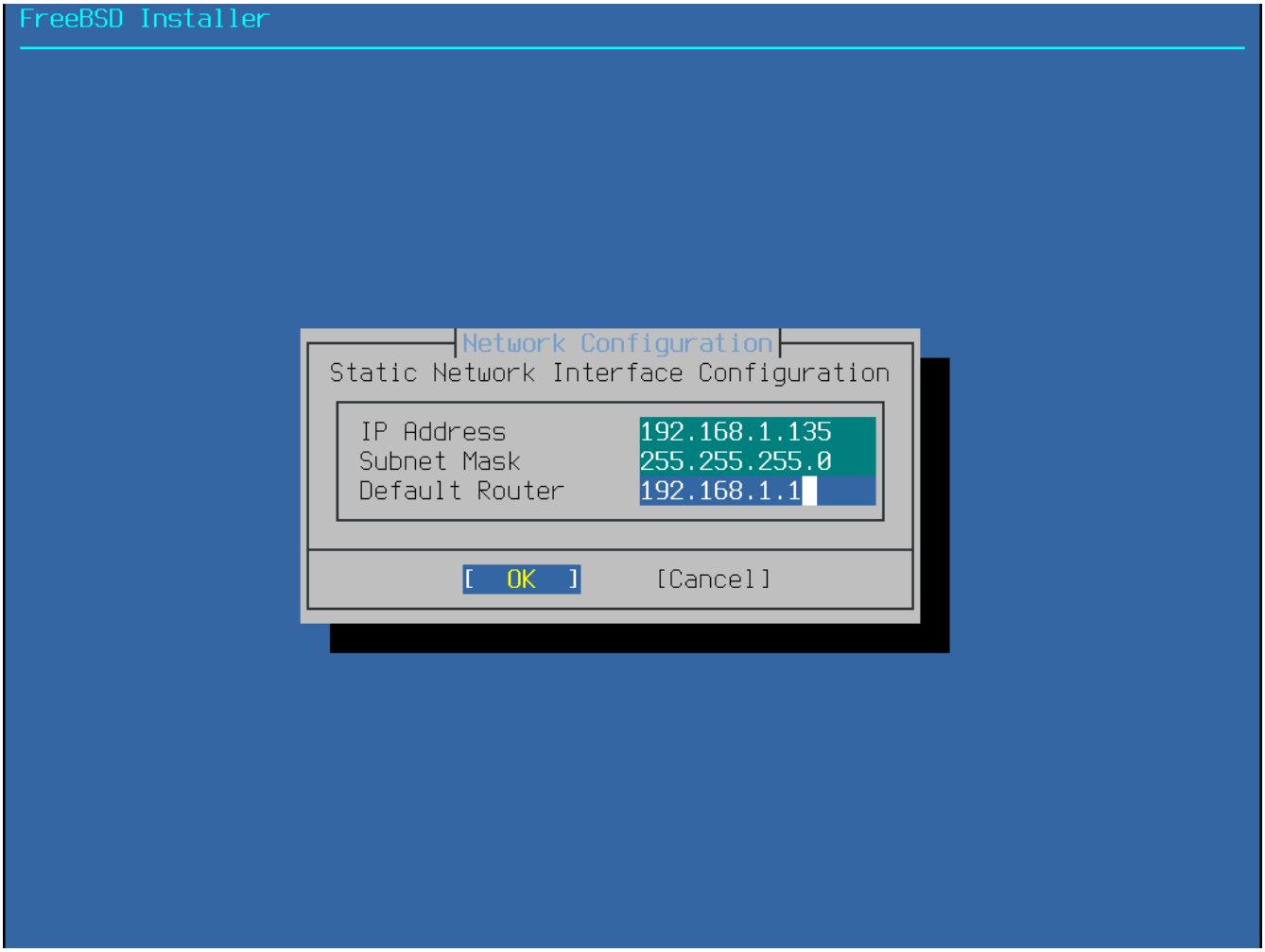


図 *41.* 静的な *IPv4* の設定

- IP Address - コンピュータに与える IPv4 アドレスです。このアドレスは一意的である必要があるため、ローカルネットワーク上の他のデバイスですでに使われているアドレスは使えません。
- Subnet Mask - ネットワークのサブネットマスクです。
- Default Router - このネットワークのデフォルトゲートウェイの IP アドレスです。

次の画面では、インタフェースを IPv6 で設定すべきかを選択します。 IPv6 が利用でき、希望するのであれば、 **[Yes]** を選択してください。

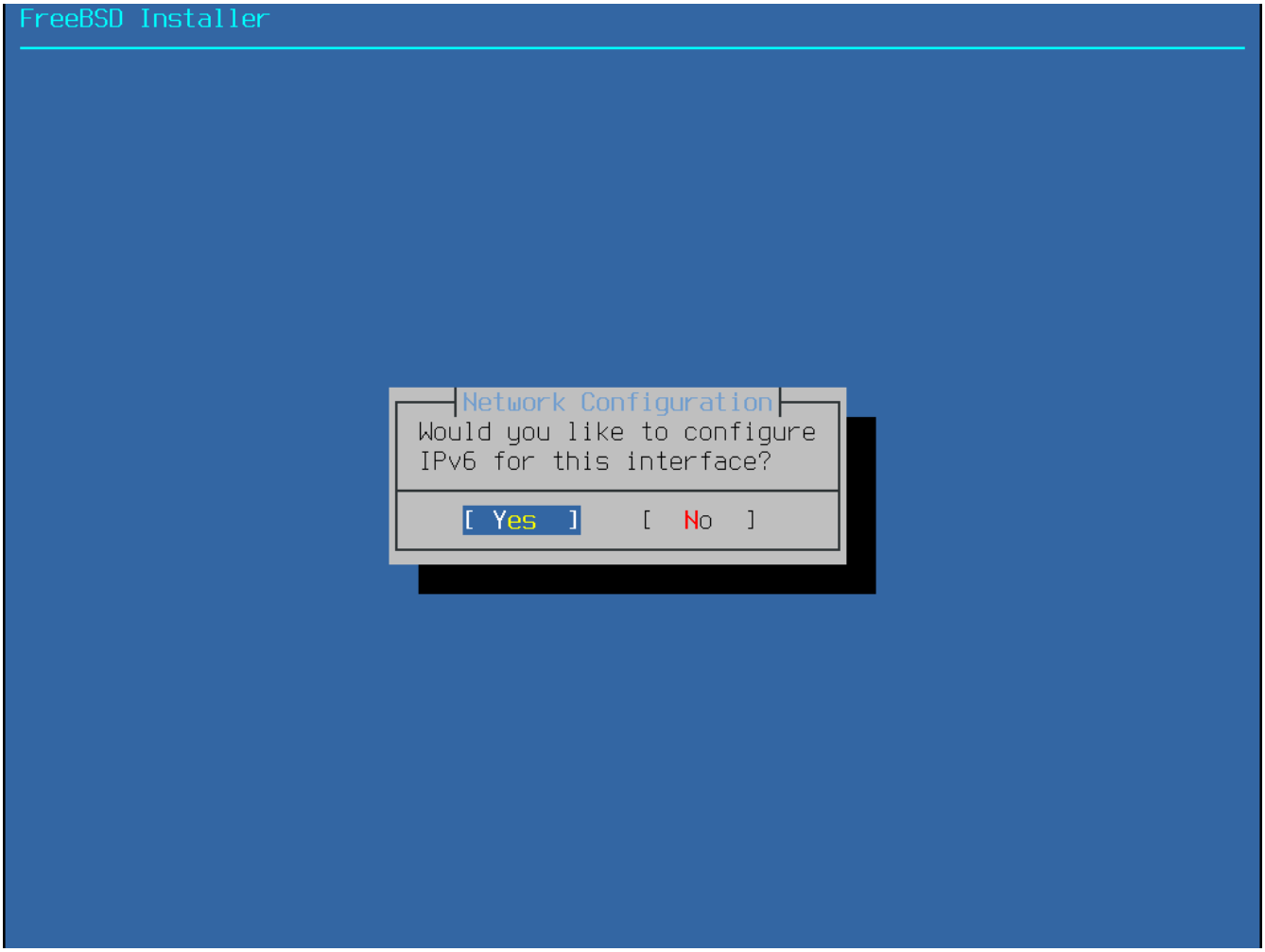


図 *42. IPv6* ネットワークの選択

IPv6 の設定に関しても 2 つの方法があります。 StateLess Address AutoConfiguration (SLAAC) は、ローカルルータから適切なネットワーク設定情報を入手するように、自動的にリクエストします。
詳細については rfc4862 を参照してください。
静的な設定では、ネットワーク情報を手動で入力する必要があります。

IPv6 ルータを利用できるのであれば、次のメニューで **[Yes]**
を選択し、ネットワークインタフェースの設定を自動的に行ってください。
インストーラはルータを見つけ出し、システムに対するアドレス情報を入手するため、
少しの間停止しているように表示されます。

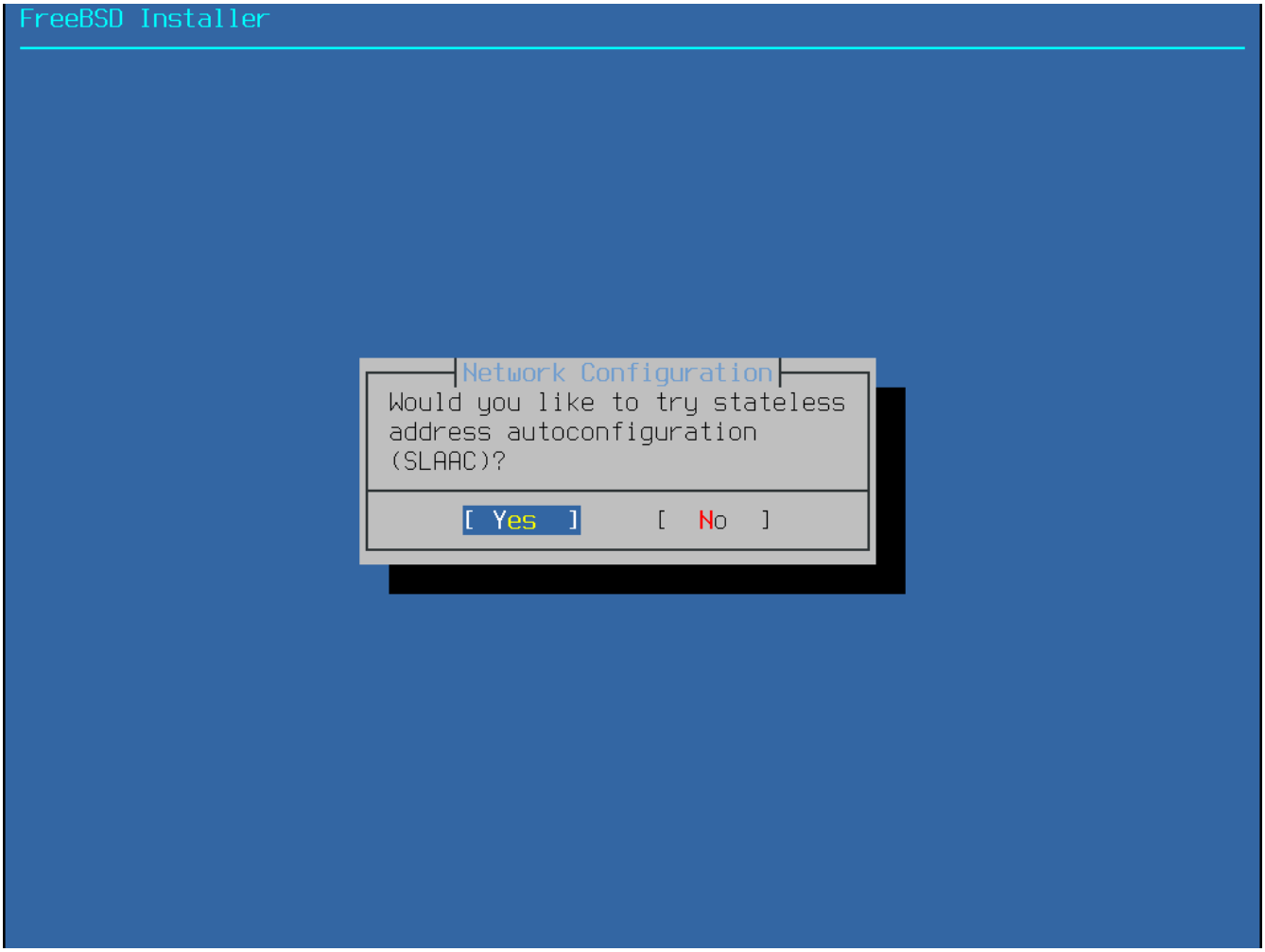


図 *43. IPv6 SLAAC* 設定の選択

IPv6 ルータが利用できない環境では、**[No]** を選択して、表示されるメニューで以下のアドレス情報を入力する必要があります。

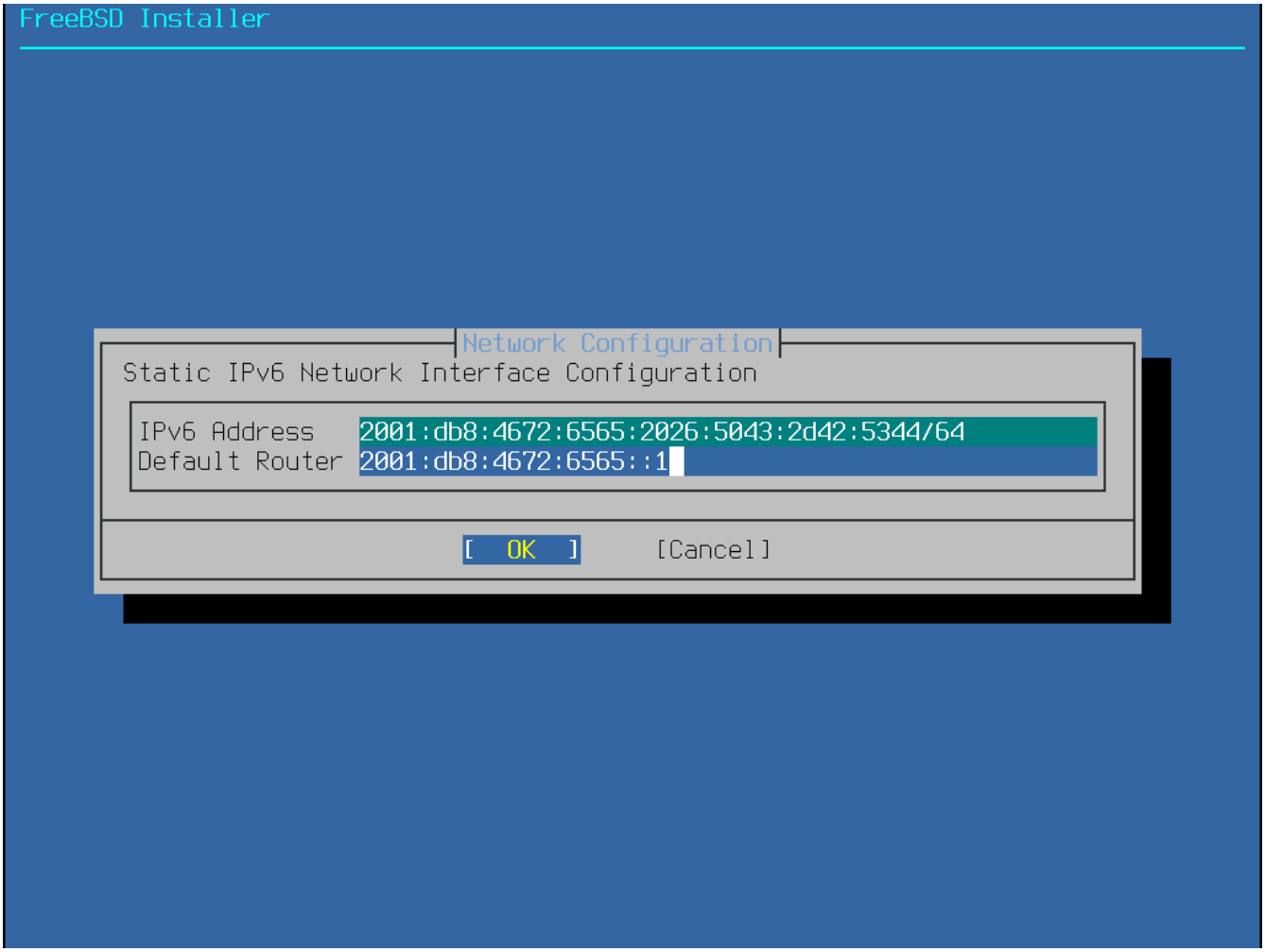


図 *44. IPv6* の静的な設定

- IPv6 Address - このコンピュータに割り当てられた IPv6 アドレスです。このアドレスは一意的である必要があるため、ローカルネットワーク上の他のデバイスですでに使われているアドレスは使えません。
- Default Router - このネットワークのデフォルトゲートウェイの IPv6 アドレスです。

最後のネットワークメニューでは、ホスト名とネットワークアドレスを変換する Domain Name System (DNS) リゾルバを設定します。 すでに DHCP または SLAAC を使って自動的にネットワークインタフェースを設定したのであれば、Resolver Configuration には値がすでに入っているでしょう。 そうでなければ、Search フィールドにローカルネットワークのドメイン名を入力してください。 DNS #1 および DNS #2 は、ローカル DNS サーバの IPv4 または IPv6 アドレスです。 少なくとも、1 つの DNS サーバは必要です。

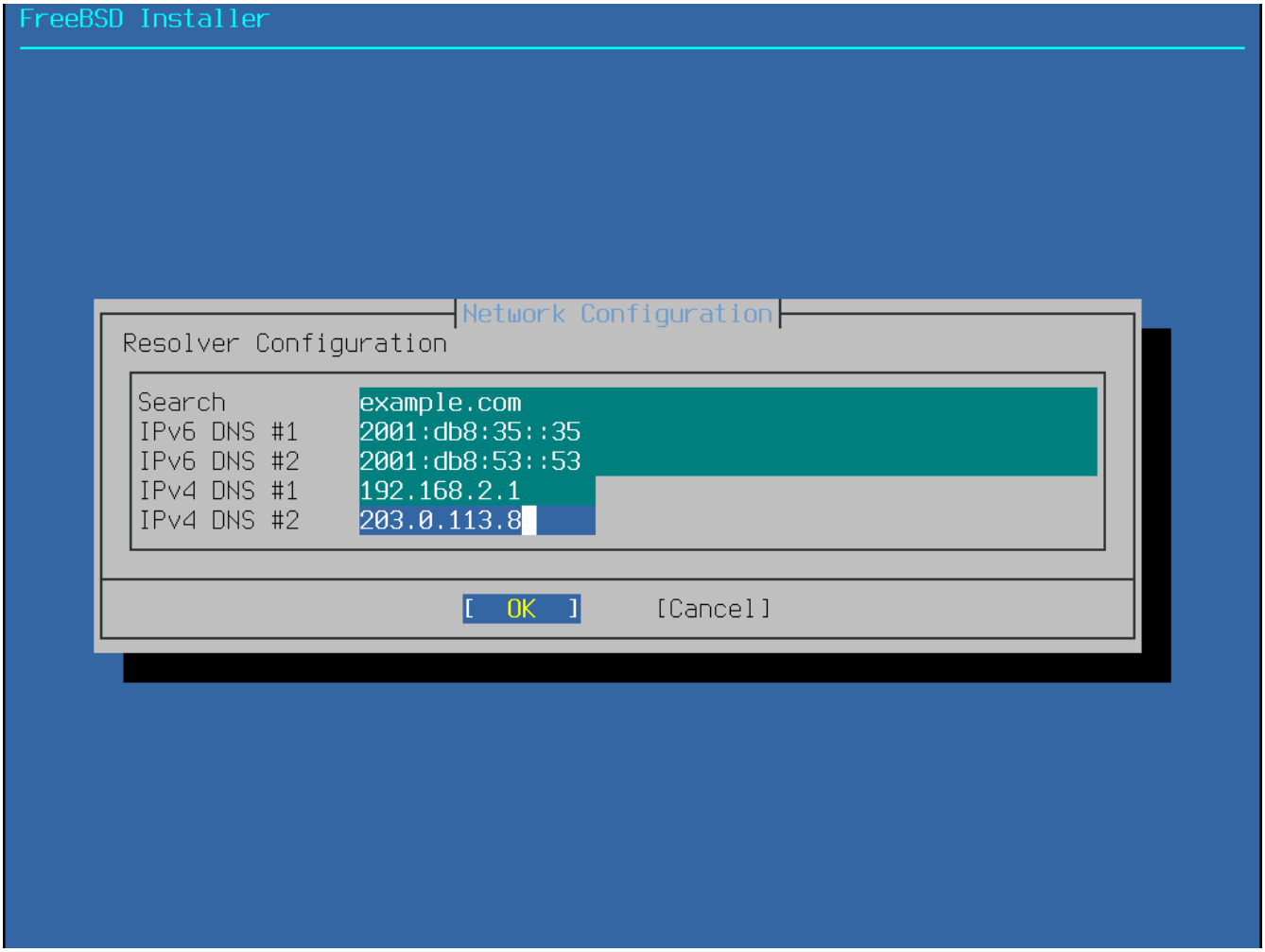


図 *45. DNS* の設定

ネットワーク接続の設定が終わったら、FreeBSD
をインストールするコンピュータと同じ地域のミラーサイトを選んでください。
ターゲットコンピュータの近くのミラーサイトを選択できれば、ファイルのダウンロードは早く終わるので、インストールの時間は短くなります。

ftp://ftp.freebsd.org (Main Site)`
を選択すると、最も近いミラーに自動的にルーティングします。

図 *46.* ミラーサイトの選択

2.8.3. タイムゾーンの設定

次のメニューでは、地域、国、タイムゾーンを指定します。
使用しているシステムのタイムゾーンを設定することで、
夏時間などの地域による時刻の違いが自動的に調整され、
タイムゾーンに関連した機能が適切に取り扱われます。

ここでの例は、　　欧州スペインのメインランドタイムゾーンにあるコンピュータに対するものです。
実際の地理的位置に対応するタイムゾーンを設定してください。

図 *47.* 地域の選択

矢印キーを使って、適切な地域を選択し、 Enter を押してください。

図 *48.* 国名の選択

矢印キーを使って、適切に国名を選び、 Enter を押してください。

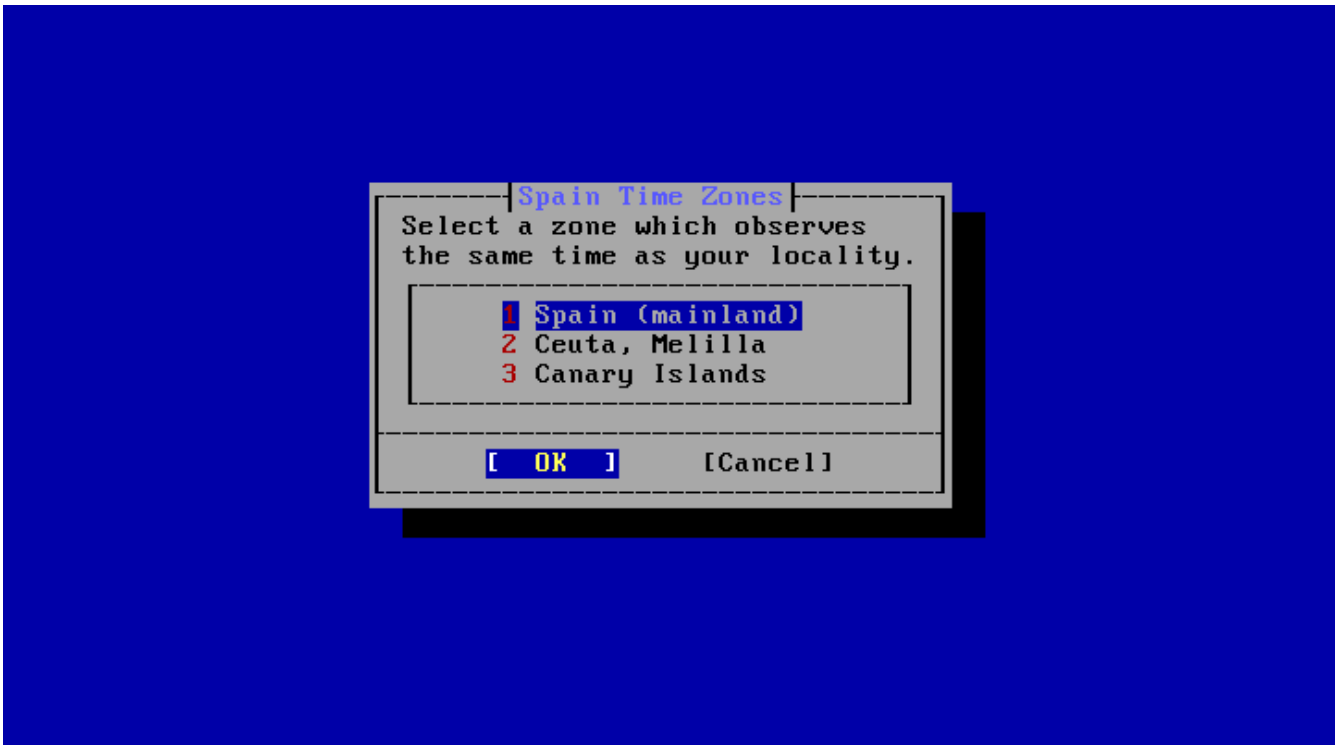


図 *49.* タイムゾーンの選択

矢印キーを使って適切なタイムゾーンを選択し、 Enter を押してください。

図 *50.* タイムゾーンの確定

タイムゾーンの省略形が正しいかどうかを確認してください。

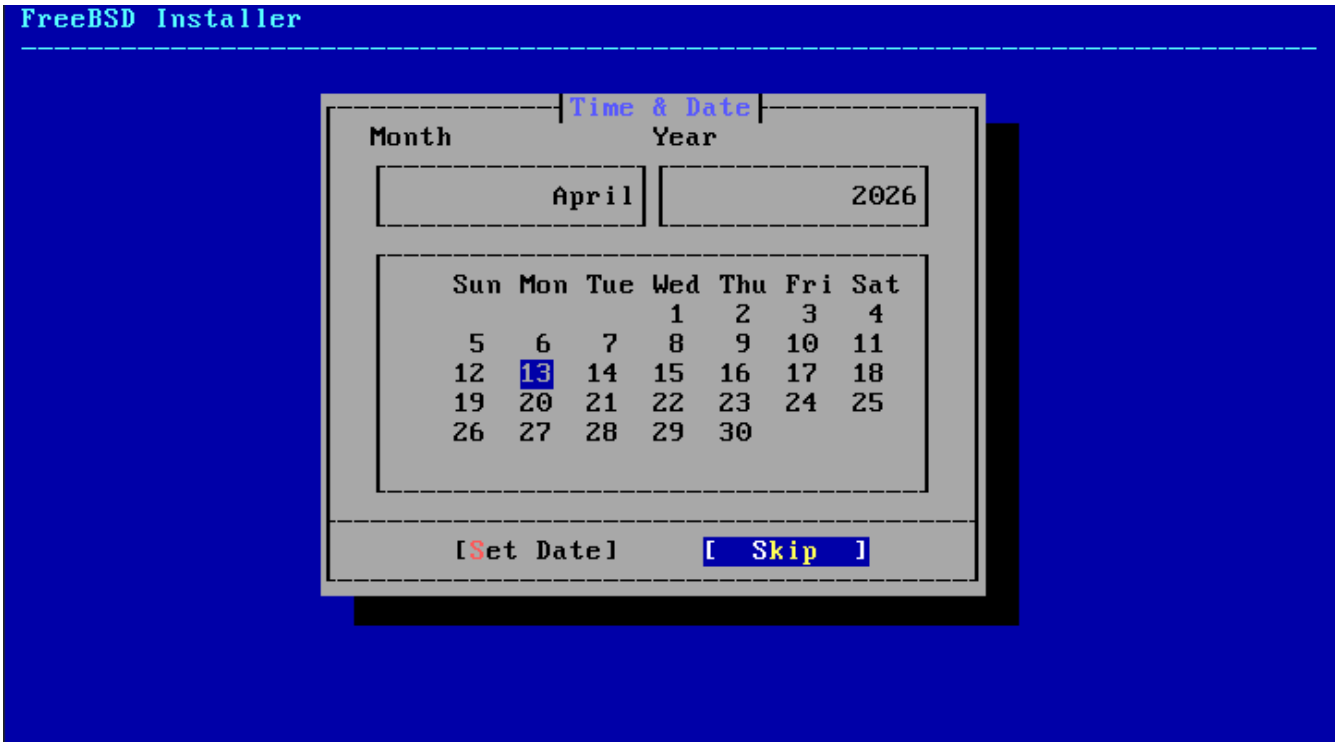


図 *51.* 日付の設定

矢印キーを使って適切な日付を選択したら、 **[Set Date]** を押してください。 **[Skip]** を押すと日付の設定をスキップできます。

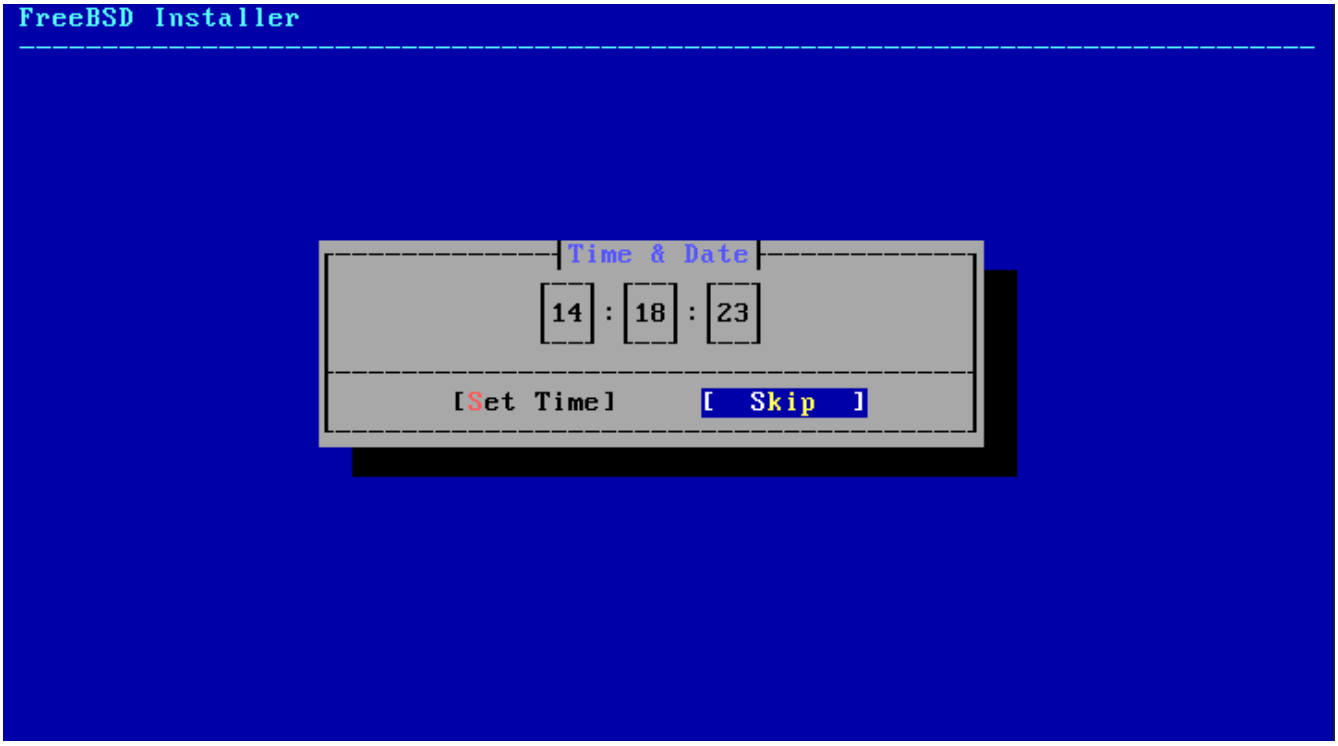


図 *52.* 時刻の設定

矢印キーを使って適切な時刻に設定したら、 **[Set Time]** を押してください。 **[Skip]** を押すと時刻の設定をスキップできます。

2.8.4. サービスを有効にする

次のメニューでは、システムが起動した時に、 起動するシステムサービスを設定します。
これらのサービスはすべてオプションです。
システムの機能として必要なサービスだけを起動するようにしてください。

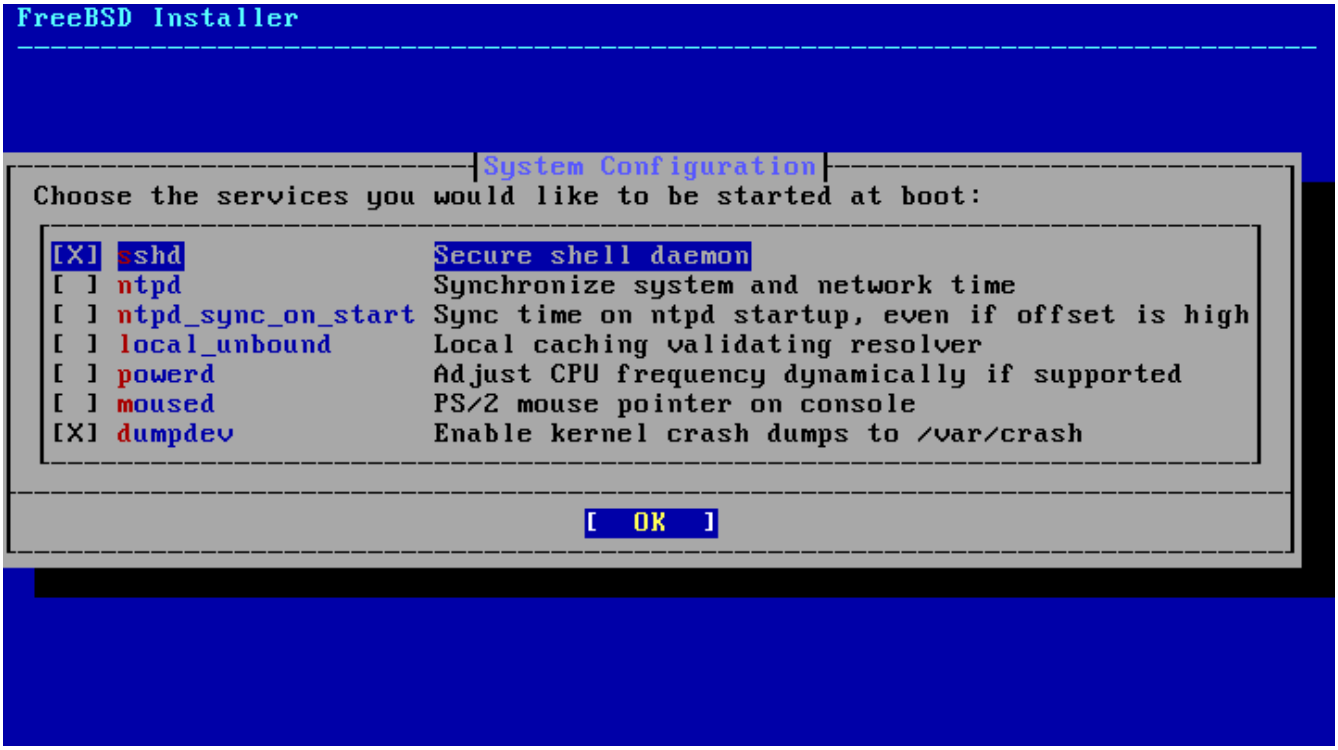


図 *53.* 追加で有効にするサービスの選択

このメニューで有効にできるサービスは以下の通りです。

- local_unbound - DNS のローカル unbound を有効にします。この設定は、ローカルキャッシュフォワードリゾルバとしての利用のみを想定しています。ネットワーク全体のリゾルバを設定したいのであれば、dns/unbound をインストールしてください。
- sshd - セキュアシェル (SSH) デーモンは、暗号化された接続上でリモートアクセスするために使われます。
システムがリモートログインを必要とする場合のみ、 このサービスを有効にしてください。
- moused - システムのコンソールで、 マウスを利用する時に、このサービスを有効にしてください。
- ntpdate - 起動時の自動時刻同期を有効にします。 この機能は、現在 ntpd(8) デーモンでも利用できます。 ntpdate(8) ユーティリティは近々その役目を終える予定です。
- ntpd - 自動時刻同期のための The Network Time Protocol (NTP) デーモンを有効にします。リモートタイムサーバまたはプールとシステムの時刻を同期する場合は、このサービスを有効にしてください。
- powerd - 電源の管理およびエネルギーを節約するための電源コントロールユーティリティ
- dumpdev - システムのデバッグを行う上で、クラッシュダンプは有用なので、可能であれば有効にすると良いでしょう。

2.8.5. セキュリティを強化するオプションを有効にする

次のメニューでは、 有効にするセキュリティオプションを設定します。
すべてはオプションですが、有効にすることが推奨されます。

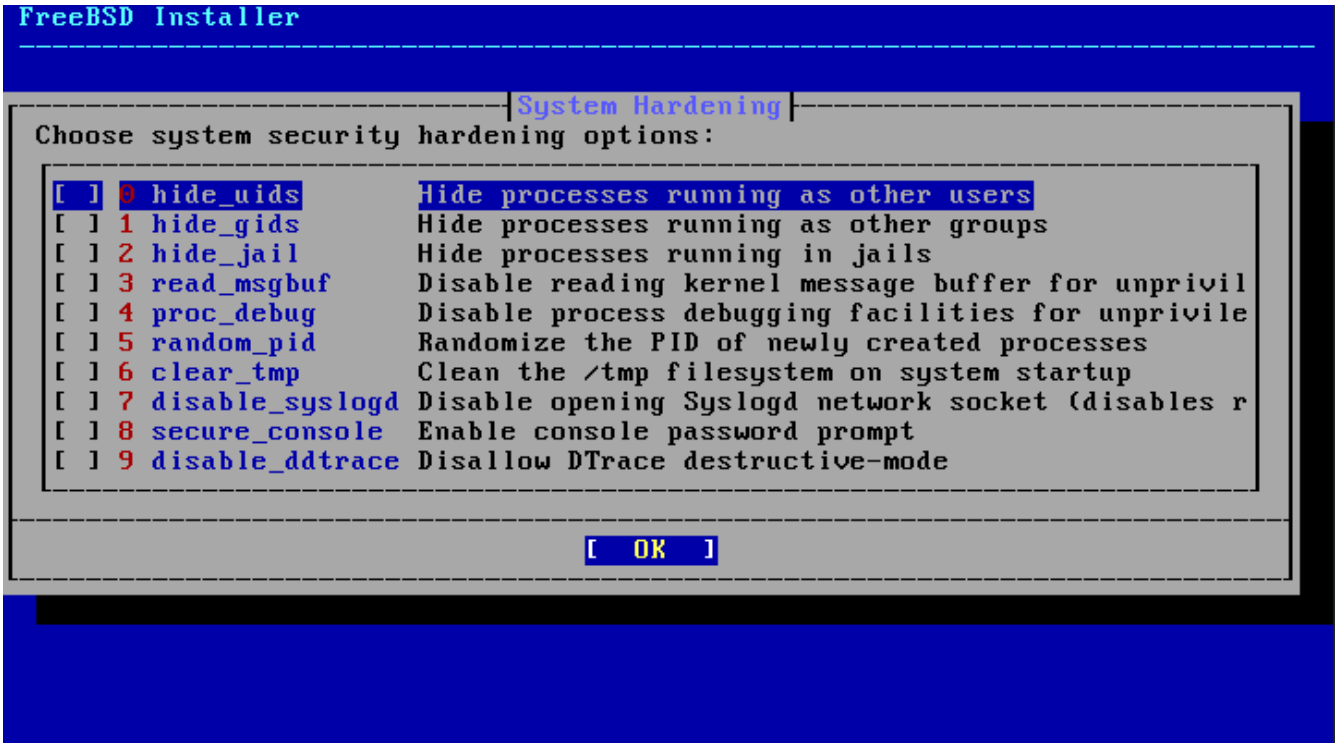


図 *54.* セキュリティを強化するオプションの設定

このメニューで有効にできるのは、以下のオプションです。

- hide_uids - 他のユーザが実行しているプロセス (UID) を隠します。
特権のないユーザが、他のユーザにより実行されているプロセスを見れないようにします。

- hide_gids - 他のグループが実行しているプロセスを隠します。 特権のないユーザが、他のグループ (GID) により実行されているプロセスを見れないようにします。
- hide_jail - jail で実行中のプロセスを隠します。 特権のないユーザが、jail の中で実行されているプロセスを見れないようにします。
- read_msgbuf - 権限のないユーザが、カーネルメッセージバッファを読めなくします。 権限のないユーザが、dmesg(8) を使ってカーネルログバッファのメッセージを見れないようにします。
- proc_debug - 権限のないユーザ対するプロセスデバッキング機能を無効にします。 ptrace() および ktrace() といった procfs 機能を含む、権限のないプロセス間のデバッキングサービスを無効にします。 このオプションによって、PHP などのスクリプト言語に対する組み込みのデバッキング機能と同様に、lldb(1), truss(1) および procstat(1) などの権限のないユーザによるデバッキングツールも無効になります。
- random_pid - プロセスの PID をランダム化します。
- clear_tmp - システムの起動時に /tmp を空にします。
- disable_syslogd - syslogd ネットワークソケットを閉じます。 デフォルトでは、FreeBSD は syslogd を -s を使った安全な方法で実行します。 これは、外からのポート 514 に対する UDP リクエストを待機しません。 このオプションを有効にすると、syslogd を -ss 付きで実行します。 これにより、syslogd は空いているどのポートからも受け付けません。 詳細は、syslogd(8) をご覧ください。
- disable_sendmail - sendmail MTA を無効にします。
- secure_console - シングルユーザモードに入る際に、コマンドプロンプトに対して root パスワードが必要となります。
- disable_ddtrace - DTrace は、 実行中のカーネルに影響を及ぼすモードで実行できます。 破壊的なアクションは、明示的に有効にしない限りは利用できません。 このオプションを有効にするには、DTrace を実行する際に -w を使ってください。 詳細については dtrace(1) をご覧ください。
- enable_aslr - アドレス空間配置のランダム化を有効にします。アドレス空間配置のランダム化の詳細については Wikipedia article をご覧ください。

2.8.6. ユーザの追加

次のメニューでは、少なくとも一人のユーザを追加してください。 システムには root ではなく、ユーザアカウントでログインすることが推奨されています。 root 権限でログインすると、実行に対して制限がなく、また、保護されません。 通常のユーザでログインすることにより、 安全でセキュリティ的に危険が少なくなります。

[Yes] を選択し、 新しいユーザを追加してください。

図 *55.* 新しいユーザのアカウントの作成

プロンプトに従い、 ユーザアカウントの作成で必要となる情報を入力してください。 ユーザ情報の入力で示されている例では、asample ユーザアカウントを作成します。

```
FreeBSD Installer
=========================
Add Users

Username: imani
Full name: imani
Uid (Leave empty for default):
Login group [imani]:
Login group is imani. Invite imani into other groups? []: wheel
Login class [default]:
Shell (sh csh tcsh nologin) [sh]:
Home directory [/home/imani]:
Home directory permissions (Leave empty for default):
Enable ZFS encryption? (yes/no) [no]:
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]:
```

図 *56.* ユーザ情報の入力

以下は、入力情報のまとめです。

- Username - ログイン時のユーザ名を入力します。一般的な慣習では、ファーストネームの最初の文字とラストネームを、
  ユーザ名がシステムで一意的になる長さで組み合わせます。 ユーザ名は、大文字と小文字を区別し、空白を含んではいけません。

- Full name - ユーザのフルネーム。 空白を含むことは可能です。 また、この情報はユーザアカウントの説明の記述に使われます。
- Uid - ユーザ ID 番号。 通常は、システムが自動的に割り当てるように、 空欄のままにします。
- Login group - 新しいユーザのログイングループ。 空欄のままにすると、デフォルトに割り当てられます
- Invite user into other groups? - ユーザを別のグループのメンバーとして追加するかどうか。 ユーザが管理者としてのアクセス必要であれば、 ここで wheel を入力してください。
- Login class - 空欄にするとデフォルトの設定になります。
- Shell - 一覧の中から、ユーザのシェルを入力してください。 シェルに関する詳細については シェル をご覧ください。
- Home directory - ユーザのホームディレクトリ。 通常は、デフォルトの場所が適切です。
- Home directory permissions - ユーザのホームディレクトリの権限。 通常は、デフォルトが適切です。
- Use password-based authentication? - 通常は、ユーザがログイン時にパスワードの入力が要求されるように yes と入力してください。
- Use an empty password? - 空のパスワードは安全ではないので、通常は no です。
- Use a random password? - 通常は、次のプロンプトでユーザ自身のパスワードを入力できるように、 no です。
- Enter password - ユーザのパスワードです。入力している文字は画面に表示されません。
- Enter password again - 確認のため、パスワードをもう一度入力します。
- Lock out the account after creation? - 通常は、ユーザがログインできるようにするため、 no です。

すべての詳細を入力したら、サマリが表示され、 正しいかどうかの確認を求められます。 入力した情報に間違いがあれば、 no を入力して修正してください。 すべてが正しく入力されていれば、 yes を入力して新しいユーザを作成してください。

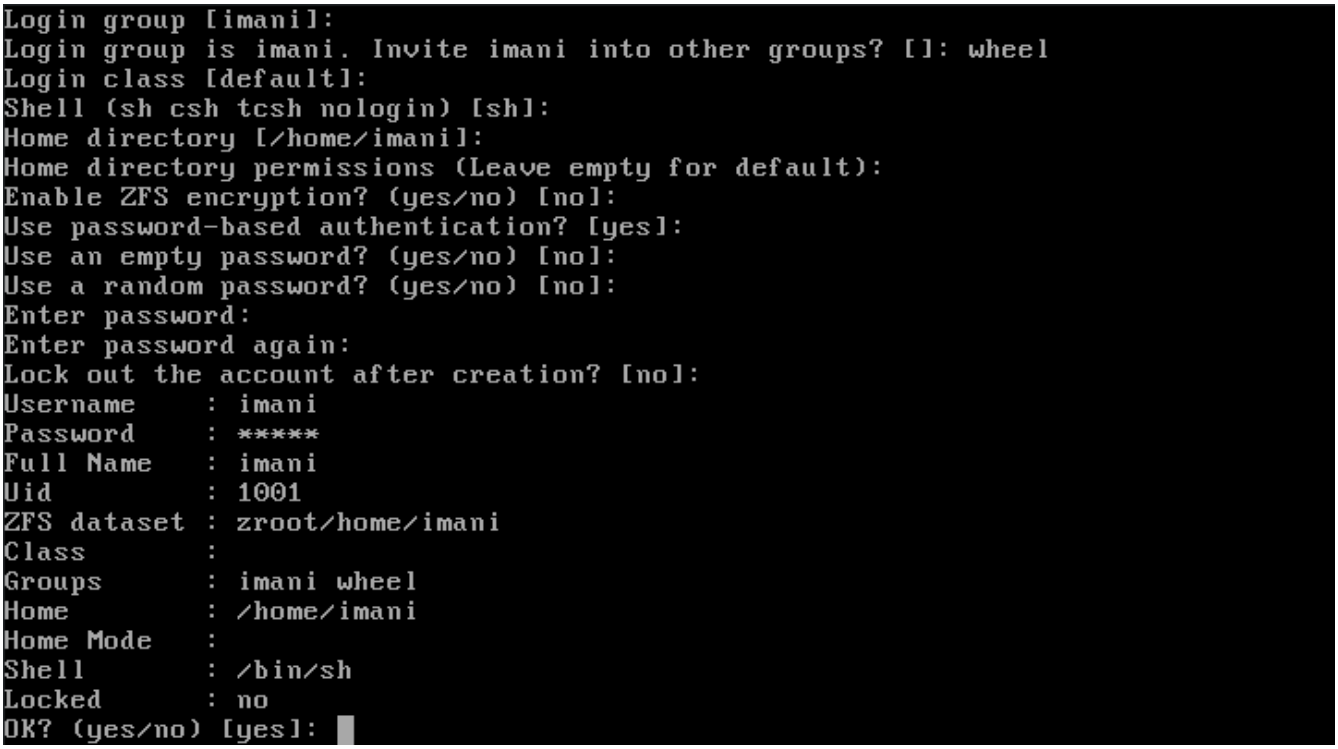


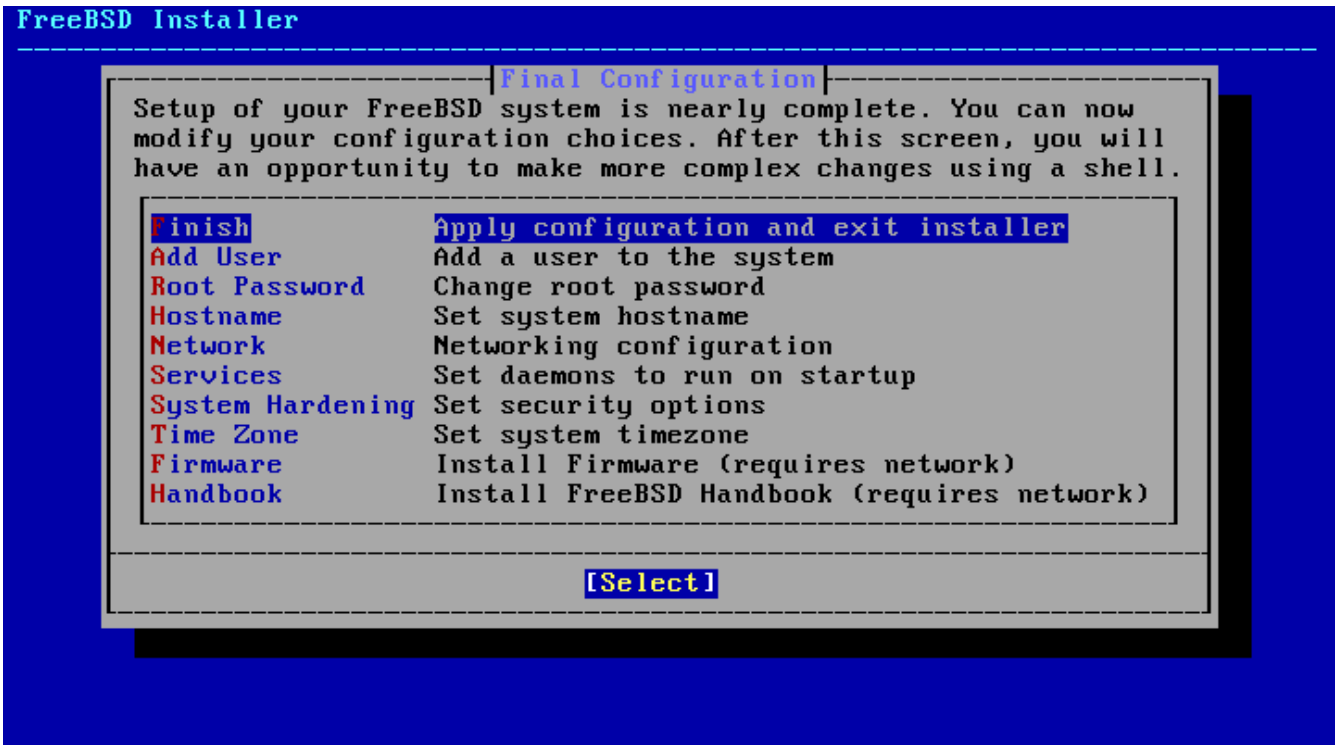
図 57. ユーザおよびグループの管理を終了する

さらにユーザを追加するのであれば、 `Add another user?` の質問に対し、 `yes` を入力してください。 `no` を入力すると、ユーザの追加が終わり、次に進みます。

ユーザの追加や、ユーザ管理の詳細については、 ユーザと基本的なアカウント管理 を参照してください。

2.8.7. 最後の設定

すべてをインストールし、設定が終わった後に、 最後に設定を修正する機会が与えられます。



インストールを完了する前に、このメニューを使って変更、または、追加の設定を行なってください。

最終の設定オプション

- Add User - ユーザの追加 で説明しています。
- Root Password - root パスワードの設定 で説明しています。
- Hostname - ホスト名の設定 で説明しています。
- Network - ネットワークインタフェースの設定 で説明しています。
- Services - サービスを有効にする で説明しています。
- System Hardening - セキュリティを強化するオプションを有効にする で説明しています。
- Time Zone - タイムゾーンの設定 で説明しています。
- Handbook - FreeBSD ハンドブックのダウンロードとインストール。

設定が完了したら、**[Exit]** を選んでください。

図 *58. Manual Configuration*

新しいシステムを再起動する前に、 bsdinstall は追加の設定が必要かどうかを尋ねてきます。 **[Yes]** を選択して新しいシステムのシェルに入るか、または **[No]** を選択して、インストールの最後のステップに進んでください。

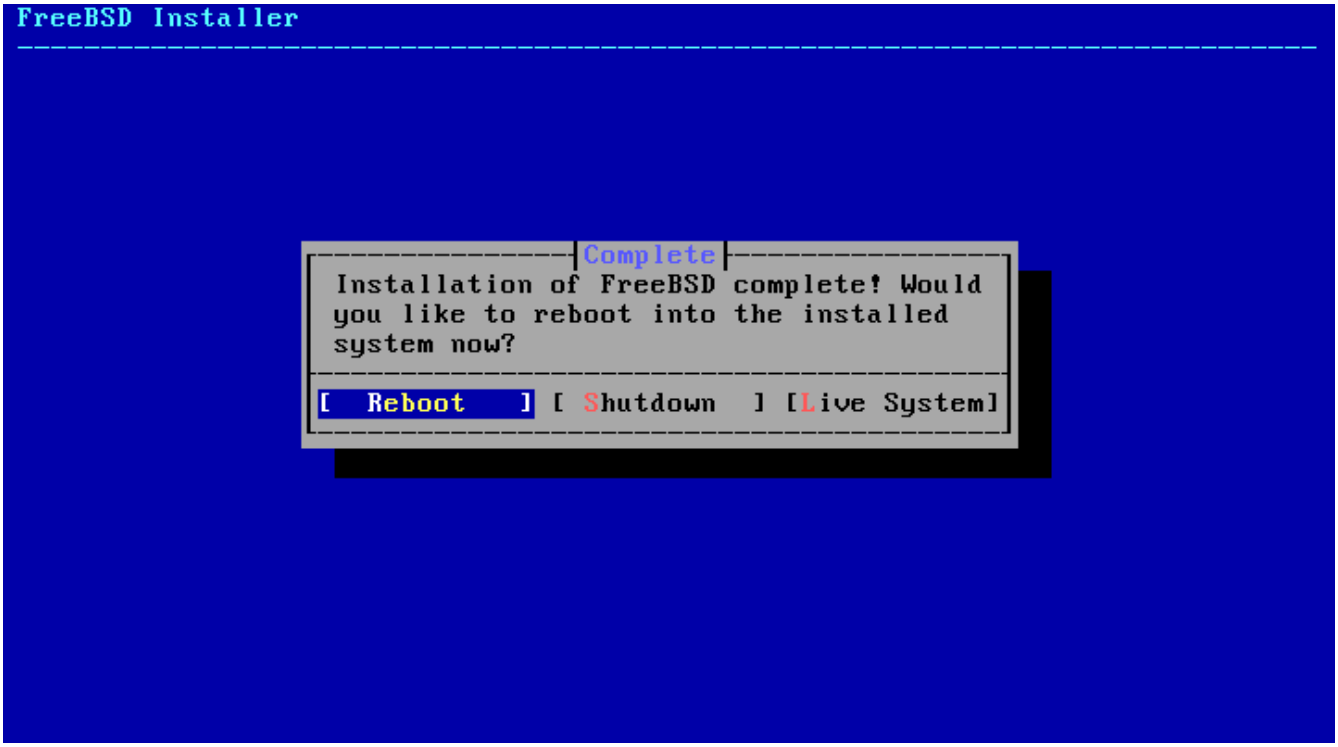


図 *59.* インストールの終了

追加の設定や、特別なセットアップが必要であれば、**[Live CD]** を選んでインストールメディアを Live CD で起動してください。

インストールが終わったら、**[Reboot]** を選んで、コンピュータを再起動し、新しい FreeBSD システムで起動してください。再起動する前には、忘れずに FreeBSD インストールメディアを外してください。
さもないと、もう一度インストールメディアから起動してしまいます。

FreeBSD の起動時には、多くのメッセージが画面に表示されます。
システムの起動後には、ログインプロンプトが表示されます。`login:`
プロンプトで、インストール時に追加したユーザ名を入力してください。`root`
でのログインは避けてください。
管理者の権限が必要となった時に、スーパユーザになる方法については、スーパーユーザアカウント を参照してください。

起動時に表示されていたメッセージは、`Scroll-Lock` を押し、scroll-back buffer で見ることができます。`PgUp`, `PgDn` そして矢印キーでメッセージをスクロールバックできます。
メッセージの確認が終わったら、`Scroll-Lock` をもう一度押すと、ディスプレイのロックを外し、コンソールに戻ることができます。
何度かシステムを起動した後で、これらのメッセージを見るには、コマンドプロンプトから `less /var/run/dmesg.boot` と入力してください。確認後に `q` を押すと、コマンドラインに戻ります。

追加で有効にするサービスの選択 で sshd を有効に設定した場合には、最初の起動時にシステムが SSH ホストキーを生成するため、少々時間がかかるかもしれません。その後の起動はより速くなるでしょう。
鍵のフィンガープリントは、以下の例のように表示されます。

```
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.
```

フィンガープリントおよび SSH についての詳細については、OpenSSH をご覧ください。

FreeBSD はデフォルトでは、グラフィカルな環境をインストールしません。
グラフィカルなウィンドウマネージャのインストール、 および設定に関するより多くの情報については、
X Window System をご覧ください。

適切に FreeBSD をシャットダウンすることは、
ハードウェアをダメージから守ったり、データの保護につながります。
システムを適切にシャットダウンする前に、 電源を落すということはしないでください!wheel
グループのメンバとなっているユーザは、 コマンドラインから su と入力し、 root
のパスワードを入力してスーパユーザとなってください。 その後、shutdown -p now
と入力すると、システムは正しくシャットダウンし、 ハードウェアが対応していれば、電源が落ちます。

2.9. トラブルシューティング

この章では、インストールの際の、
これまで報告された共通の問題に対する解決のための情報が書いてあります。

FreeBSD リリース情報 ページにあるインストールする FreeBSD のバージョンのハードウェアノートを調べて、
使用しているハードウェアに対応しているかどうかを確認してください。

いくつかのインストール上の問題は、さまざまなハードウェア装置、
特にマザーボードのファームウェアのアップデートで回避または緩和することができます。 マザーボードのファームウェアは、通常 BIOS と呼ばれます。
多くのマザーボードまたはコンピュータ製造メーカーは、
アップデートやアップグレード情報を載せているウェブサイトを用意しています。

通常、製造メーカーは、
重要な更新のようなそれなりの理由がない限り、マザーボードの BIOS のアップグレードは行わないよう推奨しています。
アップデートの過程で失敗する可能性があり、 その場合 BIOS が不完全な状態になり、コンピュータが動作しない原因となり得るからです。

システムの起動時に、ハードウェアの検出中にシステムが固まったり、インストールプロセスでおかしな振る舞いをする場合には、ACPI が原因の可能性があります。 i386 および amd64 プラットフォームにおいて、 FreeBSD はシステムの設定を手助けするシステム ACPI サービスを、起動時に検出された場合に広く使います。 残念ながら、まだいくつかの不具合が、 ACPI ドライバとシステムのマザーボードおよび BIOS ファームウェア両方に存在しています。 起動ステージ 3 において、ヒント情報 `hint.acpi.0.disabled` を以下のように設定すると ACPI を無効にできます。

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

この設定はシステムが起動するたびにリセットされるので、`/boot/loader.conf` ファイルに `hint.acpi.0.disabled="1"` を追加する必要があります。 ブートローダのより詳しい情報については FreeBSD の起動のプロセス で説明します。

2.10. Live CD を使う

ウェルカムメニュー で示されている bsdinstall のウェルカムメニューは、 **[Live CD]** オプションを提供します。 これは、 オペレーティングシステムに FreeBSD を使うべきかどうか迷っていて、 インストール前に機能を試して見たいと思っている方に有用です。

[Live CD] を使う際は、以下のことに気をつけてください。

- システムにアクセスする際には、認証を求められます。 ユーザ名は `root`、パスワードは空欄としてください。
- システムはインストールメディアから直接起動するので、
ハードディスクにインストールされたシステムに比べ、
パフォーマンスはかなり遅い可能性があります。
- このオプションのユーザインタフェースは、 コマンドプロンプトのみです。
グラフィカルなユーザインタフェースではありません。

Chapter 3. FreeBSD の基礎知識

3.1. この章では

この章では FreeBSD オペレーティングシステムの基本的なコマンドと機能について記述しています。ここに書かれてあることのほとんどは、どんな UNIX® -like なオペレーティングシステムにもあてはまります。FreeBSD の初心者であれば、この章を読んでおいた方がきっといいはずです。

この章を読んで分かることは、次のようなことです。

- 仮想コンソールの使い方と設定方法
- FreeBSD システム上でユーザやグループを作成し管理する方法
- UNIX® のファイルの許可属性の仕組みと FreeBSD のファイルフラグについて
- FreeBSD のファイルシステムの構成
- FreeBSD のディスク構成
- ファイルシステムをマウント、アンマウントする方法
- プロセス、デーモンとシグナルとはなにか
- シェルとはなにか。 また、デフォルトのログイン環境を変える方法
- テキストエディタの基本的な使い方
- デバイスおよびデバイスノードとはなにか
- さらに詳しい情報を得るためのマニュアルページの読み方

3.2. 仮想コンソールと端末

起動時に自動的にグラフィカルな環境が起動するように FreeBSD を設定していなければ、システムが起動すると、以下のようなコマンドラインのログインプロンプトが表示されます。

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

login:
```

最初の行はシステムの情報です。 `amd64` は、このシステム上で 64 ビット版の FreeBSD が動作していることを示しています。 ホスト名は `pc3.example.org`、`ttyv0` は "システムコンソール" であることを示しています。 次の行はログインプロンプトです。

FreeBSD はマルチユーザシステムなので、ユーザを区別する何がしかの手段が必要です。システム上のプログラムを実行できるようになるには、すべてのユーザに対してシステムにログインすることが義務付けられています。 すべてのユーザは、一意な "ユーザ名" と "パスワード" を持っています。

システムコンソールにログインするには、システムのインストール時に ユーザの追加 で追加したユーザ名を入力して、`Enter` を押してください。次にそのユーザのパスワードを入力して、`Enter` を押してください。

セキュリティの観点から、パスワードは 表示されません。

パスワードを正確に入力したら、日々のメッセージ (MOTD) が表示され、コマンドプロンプトが表示されます。 ユーザ作成時に選択したシェルに依存しますが、このプロンプトは `#`, `$` または `%` といった記号です。 プロンプトはユーザが FreeBSD のシステムコンソールへログインし、利用可能なコマンドを実行する準備ができていることを示しています。

3.2.1. 仮想コンソール

システムコンソールからシステムに対話的にコマンドを実行できますが、FreeBSD システム上でキーボードによりコマンドラインから利用しているユーザは、通常代わりに仮想コンソールにログインします。
デフォルトではシステムからのメッセージはシステムコンソールに出力され、これらのメッセージが、ユーザが作業しているコマンドまたはファイル上に表示されるため、ユーザが現在の作業に集中できなくなるためです。

デフォルトでは FreeBSD は、複数の仮想コンソールを表示してコマンドを入力できるように設定されています。
各仮想コンソールは、個別のログインプロンプトおよびシェルを持っており、簡単に仮想コンソール間の切り替えができます。
これにより、グラフィカルな環境において同時に複数のウィンドウを開いてコマンドラインの環境を提供できます。

FreeBSD では `Alt` + `F1` から `Alt` + `F8` までのキーの組み合わせが、仮想コンソール間の切り替えに予約されています。 システムコンソール (`ttyv0`) に切り替えるには、`Alt` + `F1` を使ってください。 最初の仮想コンソール (`ttyv1`) にアクセスするには `Alt` + `F2`、2 番目の仮想コンソール (`ttyv2`) にアクセスするには `Alt` + `F3`、といったように使ってください。 Xorg をグラフィカルなコンソールとして使用しているときには、`Ctrl` + `Alt` + `F1` の組み合わせを使用すると、テキストベースの仮想コンソールへ戻ります。

あるコンソールから他に切り替えるのに応じて、FreeBSD は画面への出力を管理します。 結果として、FreeBSD で動かすコマンドを入力するのに使える複数の画面とキーボードを仮想的に実現できるのです。
ある仮想コンソールで実行したプログラムは、ユーザが別の仮想コンソールに切り替えても実行を停止しません。

FreeBSD のコンソールおよびキーボードドライバに関するさらなる技術的な説明については、kbdcontrol(1), vidcontrol(1), atkbd(4), syscons(4) および vt(4) を参照してください。

FreeBSD では以下の `/etc/ttys` のセクションのように複数の利用可能な仮想コンソールが設定されています。

```
# 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
```

仮想コンソールを無効にするには、無効にしたい仮想コンソールの行をコメント記号 (#) から始まるように設定してください。 たとえば、利用可能な仮想コンソールを 8 つから 4 つに減らす場合には、ttyv5 から ttyv8 までの仮想コンソールを表す最後の 4 行の先頭に # を挿入してください。 システムコンソールを表す ttyv0 から始まる行はコメントアウトしないでください。 最後の仮想コンソール (ttyv8) は、 X Window System で説明されているように Xrog がインストールされて設定されている場合に、グラフィカル環境にアクセスするために使用されます。

このファイルのそれぞれのカラムと仮想コンソールに設定可能なオプションの詳しい説明は、ttys(5) のマニュアルを参照してください。

3.2.2. シングルユーザモード

FreeBSD のブートメニューでは、"シングルユーザモード" と表示されているオプションが提供されています。 このオプションを選択すると、システムは "シングルユーザモード" と呼ばれる特別なモードで起動します。 このモードは、システムが起動しない場合に修正のため、または root のパスワードが分からなくなってしまいリセットするときなど、特別な状況で利用されます。 シングルユーザモードで動かしている場合は、ネットワークや他の仮想コンソールは利用できません。 しかし、システムへの完全な root 権限を利用でき、デフォルトの設定では root のパスワードは必要ありません。 このような理由のため、このモードで起動する場合には物理的なキーボードへのアクセスが必要であり、FreeBSD システムの安全性の観点からキーボードに物理的にアクセスできる人を決めておく事が必要です。 シングルユーザモードを管理する設定は、/etc/ttys ファイルの以下のセクションにあります。

```
# 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
```

デフォルトでは、status は secure に設定されています。 これは、キーボードへアクセスできるかユーザが誰であるかが重要ではない、もしくはアクセスできるユーザについては物理的なセキュリティポリシーでコントロールされていることが前提となっています。 この設定を insecure に変更するケースとしては、システムは安全ではなく、誰でもキーボードにアクセスできる環境が想定されます。 この行を insecure に変更すると、FreeBSD がシングルユーザモードで起動した場合に root のパスワードが要求されます。

insecure に変更する場合は十分注意してください! root のパスワードを忘れてしまうと、シングルユーザモードで起動することはできますが、FreeBSD

の起動のプロセスに詳しくない人が起動できるようにするのは難しいかも知れません。

3.2.3. コンソールのビデオモードの変更

FreeBSD のデフォルトのビデオモードは 1024x768 や 1280x1024 など、グラフィックチップおよびディスプレイが対応しているサイズに調整されます。
別のビデオモードを使うには、VESA モジュールをロードしてください。

```
# kldload vesa
```

その後、ハードウェアが対応しているビデオモードを vidcontrol(1) を使って確認してください。
以下を実行すると、対応しているビデオモードを調べることができます。

```
# vidcontrol -i mode
```

このコマンドは、使用しているハードウェアが対応しているビデオモードの一覧を表示します。
その後、vidcontrol(1) を root ユーザで実行して、新しく使用するビデオモードを選択してください。

```
# vidcontrol MODE_279
```

このビデオモードで良ければ、起動時に自動的に設定されるように /etc/rc.conf
に以下のように追加してください。

```
allscreens_flags="MODE_279"
```

3.3. ユーザと基本的なアカウント管理

FreeBSD は、複数のユーザが同時にコンピュータを使えるようにします。
スクリーンとキーボードの前に一度に座れるのはその中の一人だけですが
ユーザは何人でもネットワークを通してログインできます。 システムを使うためには、
どのユーザもアカウントがなければなりません。

この章では、以下のことを説明します。

- FreeBSD システムにおけるさまざまな種類のユーザアカウントについて
- ユーザアカウントを追加、削除および変更する方法
- ユーザやグループが利用できるリソースの上限を制御する方法
- グループの作成、 およびグループにユーザをメンバとして追加する方法

3.3.1. アカウントの種類

FreeBSD システムへアクセスするには、 かならずアカウントが使われ、
また、プロセスもすべてユーザによって実行されるので、
ユーザとアカウントの管理は、重要なものです。

アカウントには大きく分けて三種類あります。 システムアカウント (system accounts)、ユーザアカウント (user accounts)、 そしてスーパーユーザ (superuser) です。

システムアカウント

システムアカウントは、DNS、メール、
ウェブサーバといった各種サービスを運用するために使われます。
この目的は、セキュリティを確保するためです。
もしすべてのサービスがスーパーユーザで実行されていると、
それらのサービスはどんな動作でも可能となり、 適切な制限を適用することができません。

システムアカウントの具体例は、 `daemon`, `operator`, `bind`, `news` および `www` といったものです。

`nobody` は通常の特権を持たないシステムアカウントです。 しかし、`nobody` を利用するサービスが増えれば増えるほど、 それに所属するファイルやプロセスも増え、その特権も大きくなるということを忘れないようにしてください。

ユーザアカウント

ユーザアカウントは、 主に現実のユーザがシステムにアクセスする手段として用いられるものです。
システムにアクセスするすべてのユーザは、 それぞれ唯一のユーザアカウントを持つべきです。
こうすることで管理者は誰が何を行なっているかがわかりますし、
他の人の設定を壊してしまうようなことを避けることができます。

それぞれのユーザは快適にシステムを利用するため、 シェル、エディタ、キー設定、言語など、各自の環境をセットアップすることができます。

FreeBSD システム上のどのアカウントにも、 以下のような情報がなにかしら結び付けられています。

ユーザ名

`login:` プロンプトに対して入力するユーザの名前です。 各ユーザは一意なユーザ名を持つ必要があります。 有効なユーザ名を作成するには passwd(5) に記載されているいくつもの規則があります。 アプリケーションの上位互換性を保つために、8 文字以下の小文字からなるユーザ名を使うことが推奨されています。

パスワード

各アカウントにはパスワードがあります。

ユーザ **ID (UID)**

ユーザ ID (UID) は、 FreeBSD システムがユーザを一意に識別するための数値です。 ユーザ名を指定できるコマンドは、 ユーザ名を UID に変換してから扱っています。 65535 より大きな UID は、ソフトウェアによっては互換性の問題を引き起こす可能性があるので、65535 以下の UID を使用することが推奨されています。

グループ **ID (GID)**

グループ ID (GID) は、 ユーザが属する第一グループを一意に識別するための数値です。 グループは、UID ではなく、 ユーザの GID に基づいて資源へのアクセスを制御する仕組みです。 これは、ある種の設定ファイルのサイズを大幅に小さくします。
ユーザは、複数のグループに所属できます。 65535 より大きな GID は、ソフトウェアに問題を引き起こす可能性があるので、 65535 以下の GID を使うことを推奨します。

ログインクラス

ログインクラスはグループの仕組みを拡張したもので、
別々のユーザに対してシステムを調整する時に、
さらなる柔軟性を提供します。ログインクラスの詳細については、 ログインクラスの設定
で説明します。

パスワードの有効期限

デフォルトでは、パスワードに有効期限は設定されていません。
しかしながら、パスワードの有効期限をユーザごとに設定し、一部またはすべてのユーザに、一定の時間がたったらパスワードを強制的に変更させることができます。

アカウント失効時間

デフォルトでは、FreeBSD はアカウントを失効させません。
たとえば学校で生徒のアカウントがある場合など、 限られた期間だけのアカウントを作成するなら、
そのアカウントがいつ失効するか pw(8) を使って指定できます。 有効期間が経過したら、
そのアカウントのディレクトリやファイルは残っていますが、
システムへのログインはできなくなります。

ユーザの氏名

FreeBSD ではユーザ名でアカウントを一意に識別しますが、
必ずしもユーザの本名を反映したものではありません。
この情報をアカウントに関連付けることもできます。
この情報は、コメントのように、空白、大文字、および 8 字以上で記載できます。

ホームディレクトリ

ホームディレクトリは、システム中のディレクトリへのフルパスです。
これはユーザがログインした時に作業を開始するディレクトリです。
一般的な慣習は、すべてのユーザのホームディレクトリを `/home/username` か `/usr/home/username` の下に置くことです。
各ユーザは、個人のファイルやサブディレクトリを、ユーザのホームディレクトリに保存します。

ユーザシェル

シェルは、 ユーザがシステムと対話するデフォルトの環境を提供します。
いろいろな種類のシェルがあり、 経験を積んだユーザはそれぞれ好みがあり、
それをアカウントの設定に反映できます。

スーパーユーザアカウント

スーパーユーザアカウントは通常 `root` と呼ばれ、
システム管理を行なうために使われ、権限に制限がありません。
そのため、このアカウントはメールのやりとり、システムの調査、
プログラミングといった日常的な作業を行なうために使われるべきものではありません。

その理由は、スーパーユーザが通常のユーザアカウントと異なり、
操作にまったく制限を受けないことによります。
そのためスーパーユーザアカウントで操作を間違えると、
システムに重大な影響を与えてしまう恐れがあるのです。 ユーザアカウントでは、
仮に操作を間違えてもシステムを壊してしまうようなことはできないようになっています。
そのため、ユーザアカウントでログインし、

高い権限が必要なコマンドを実行するときだけスーパーユーザになることが推奨されています。

スーパーユーザで実行するコマンドはいつでも、 二回、三回と確認してください。
なぜならスペースが多かったり、文字が欠けていたりするだけで、
取り返しのつかないデータの破壊につながる可能性があるからです。

スーパーユーザの権限を得るには、さまざまな方法があります。 root
ユーザとしてログインする方法もありますが、 これはまったくお勧めできません。

スーパーユーザの権限を手に入れるには、かわりに su(1) を使って下さい。 -
オプションをつけて実行すると、 実行したユーザに root ユーザの環境が設定されます。 このコマンドは
wheel グループに入ってるユーザのみが実行でき、他のユーザは実行出来ません。 また、実行時には root
ユーザのパスワードを必要とします。

以下の例では、make install を行うときにスーパーユーザの権限が必要なので、
このコマンドを実行する時だけユーザはスーパーユーザになります。 コマンドを実行したら、ユーザは
exit を実行してスーパーユーザからログアウトし、 通常のユーザアカウントの権限に戻ります。

例 2. スーパーユーザ権限でプログラムをインストールする

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

1 人の管理者が一台のマシン、 もしくは小規模なネットワークを管理する場合には、 su(1)
のフレームワークはうまく機能するでしょう。 この代わりとなるのは、 security/sudo package または
port です。これはログ機能や、 スーパーユーザの権限で実行できるユーザやコマンドを設定できます。

3.3.2. アカウント情報の管理

FreeBSD は、ユーザアカウントを操作するためにさまざまなコマンドを用意しています。
もっとも一般的なコマンドが ユーザアカウントを管理するためのユーティティ にまとめられています。
その後で、各コマンドについて詳しい使用例を示します。
各ユーティリティの詳細や使用例についてはマニュアルページを参照してください。

表 2. ユーザアカウントを管理するためのユーティティ

コマンド	要約
adduser(8)	コマンドラインからユーザを追加するための推奨アプリケーション
rmuser(8)	コマンドラインからユーザを削除するための推奨アプリケーション
chpass(1)	ユーザデータベースの情報を変更するための柔軟なツール
passwd(1)	ユーザのパスワードを変更するコマンドラインツール
pw(8)	ユーザアカウントのあらゆる箇所を変更する強力で柔軟なツール

bsdconfig(8)	システムの設定のためのユーティリティ。アカウント管理に対応しています。

ユーザの追加

新しいユーザの登録に推奨されるプログラムは adduser(8) です。
ユーザを追加すると、このプログラムは、/etc/passwd と /etc/group を自動的に更新します。
また、新規ユーザのホームディレクトリを作成し、/usr/share/skel
から、デフォルトで使用される設定ファイルをコピーします。
また、新しく作成されたユーザに対して、ウェルカムメッセージをメールで送信することも可能です。
このユーティリティは、スーパーユーザ権限で実行する必要があります。

adduser(8) は、新しいユーザアカウントを対話的に段階的に作成するユーティリティです。 FreeBSD におけるユーザの追加
で示されているように、必要な情報を入力するか、括弧内に示されているデフォルトの値を Return
を押して承認してください。 この例では、ユーザは su(1)
によってスーパユーザ権限を取得することが可能となる wheel グループに所属します。
操作が終了すると、ユーティリティは別のユーザを追加するか、終了するかを尋ねてきます。

例 *3. FreeBSD* におけるユーザの追加

```
# adduser
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!
```

入力したパスワードは画面に表示されませんので、
ユーザアカウントを作成する際には、
パスワードを間違えて入力してしまわないように注意してください。

ユーザの削除

システムから完全にユーザを削除するには、スーパーユーザ権限で rmuser(8) を実行してください。
このコマンドは、次の手順を実行します。

1. 指定されたユーザの crontab(1) エントリが存在する場合には削除。
2. 指定されたユーザの at(1) ジョブをすべて削除。
3. 指定されたユーザが所有するすべてのプロセスに対して SIGKILL シグナルを送信。
4. ローカルパスワードファイルから、 指定されたユーザのエントリを削除。
5. 指定されたユーザのホームディレクトリを削除
   (ディレクトリの所有者が指定されたユーザのものだった場合)。実際のホームディレクトリへのシンボリックリンクの削除も含まれます。
6. /var/mail から、指定されたユーザの到着メールファイルを削除。
7. /tmp, /var/tmp, および /var/tmp/vi.recover から、指定されたユーザの所有するファイルを削除。
8. /etc/group にある すべてのグループから、指定されたユーザを削除します
   (指定されたユーザと同じ名前のグループで、そのユーザが削除されると空のグループとなる場合は、そのグループ自体が削除されます。 これは adduser(8) によってユーザごとに作成される、ユニークなグループに対応するものです)。
9. 指定されたユーザが所有するすべてのメッセージキュー、共通メモリセグメントおよびセマフォを削除

スーパユーザアカウントの削除に rmuser(8) を利用することはできません。
スーパユーザアカウントの削除はほとんどすべての場合、 大規模なシステムの破壊を意味するからです。

デフォルトでは、以下の例のような対話モードが使われます。

例 4. rmuser による対話的なアカウントの削除

```
# rmuser jru
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.
```

ユーザ情報の変更

すべてのユーザは、chpass(1)
を用いてデフォルトシェルやユーザアカウントに関連した個人情報を変更できます。

スーパユーザ権限に限り、このユーティリティを用いて他のユーザのアカウント情報も変更できます。

ユーザ名の他にオプションを指定しないと、 chpass(1) はユーザ情報を編集するエディタを表示します。ユーザがエディタを終了すると、 ユーザデータベースが新しい情報に更新されます。

スーパユーザ権限以外でこのユーティリティを実行した場合は、
エディタを抜けた後にユーザのパスワードを聞かれます。

スーパユーザによる chpass の使用 では、スーパーユーザは chpass jru と入力し、このユーザに対して変更可能なフィールドが表示されています。 jru がこのコマンドを実行すると、最後の 6 フィールドのみが表示され編集が可能です。この場合については、通常のユーザによる chpass の使用 で示されています。

例 5. スーパユーザによる chpass の使用

```
#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:
```

例 6. 通常のユーザによる chpass の使用

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

chfn(1) および chsh(1) コマンドはいずれも、chpass(1) へのリンクです。また、ypchpass(1), ypchfn(1) および ypchsh(1) も同様です。 NIS のサポートは自動的に行なわれますの、 コマンドの先頭に yp をつける必要はありません。 NIS の設定については、ネットワークサーバの章で説明されています。

ユーザのパスワードの変更

いかなるユーザも passwd(1) を使って簡単に自身のパスワードを変更できます。誤って、または不正なパスワードの変更を避けるため、新しいパスワードを設定する前に、もとのパスワードの入力が求められます。

例 7. 自分のパスワードの変更

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

スーパーユーザは、passwd(1)
をユーザ名を指定して実行することにより、いかなるユーザのパスワードを変更できます。
スーパーユーザの権限でこのユーティリティを実行する際には、もとのパスワードを入力する必要はありません。 そのため、ユーザが元のパスワードを忘れてしまっても、パスワードを変更できます。

例 8. スーパーユーザ権限での他のユーザのパスワード変更

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

chpass(1) 同様、yppasswd(1) は、 passwd(1) へのリンクになっていますので、 NIS はどちらのコマンドでも動作します。

システムユーザおよびグループの作成、削除、変更および表示

pw(8)
は、ユーザやグループの作成、削除、変更および表示を行なうコマンドラインのユーティリティです。
これは、システムユーザファイルやシステムグループファイルのフロントエンドとして働きます。 pw(8) はとても強力な一連のコマンドラインオプションを有しており、シェルスクリプトで使うのに向いていますが、新しいユーザは、この章で紹介されている他のコマンドに比べて難しいと感じるかもしれません。

3.3.3. グループの管理

グループとは、ユーザを羅列したものです。 グループは、グループ名と GID で識別されます。 FreeBSD では、 あるプロセスが何かするのを許可するかどうかをカーネルが判断する際に、 プロセスの UID とそのユーザが所属するグループの一覧を利用します。 ほとんどの場合、ユーザもしくはプロセスの GID は一覧の最初のグループを指しています。

グループ名から GID への写像は /etc/group にあります。 これは、コロンで区切られた 4 項目からなるテキストファイルです。 1 番目の項目はグループ名、2 番目は暗号化されたパスワード、3 番目が GID、4 番目がカンマで区切られたメンバの一覧です。 文法についての完全な説明は、group(5) をご覧ください。

スーパーユーザは、/etc/group をテキストエディタで編集できます。 ただし、よくある間違いを見つけてくれる vigr(8) を用いてグループファイルを編集することが好ましいです。 もしくは、pw(8) を使ってグループの追加や編集をできます。 たとえば、teamtwo というグループを追加して、その存在を確認するには、次のように使います。

operator
グループを使う時には、意図しないスーパーユーザへのアクセス権を与える可能性があるため注意が必要です。 シャットダウン、リブートおよびこのグループが所有する /dev のすべてにアクセスできるといったことが可能になってしまいます。

例 9. *pw(8)* によるグループの追加

```
# pw groupadd teamtwo
# pw groupshow teamtwo
teamtwo:*:1100:
```

この例では、1100 という番号は、 teamtwo の GID です。 この時点では、teamtwo にメンバはいません。 以下のコマンドは、 jru を teamtwo のメンバに追加します。

例 10. *pw(8)* により新しいグループにメンバを追加する

```
# pw groupmod teamtwo -M jru
# pw groupshow teamtwo
teamtwo:*:1100:jru
```

-M の引数は、カンマで区切られた新しい (空の) グループに追加するもしくは存在するグループのメンバを置き換えるユーザの一覧です。 ユーザにとっては、このグループのメンバーシップはパスワードファイルに記載されているプライマリのグループとは異なります。 pw(8) の groupshow コマンドを使った時は、そのユーザはグループの一員として表示されませんが、id(1) などのツールを使って情報を問い合わせれば、その情報を引き出せます。 ユーザをグループに追加をする際に、pw(8) は /etc/group しか扱わず、/etc/passwd から追加のデータを読んだりはしません。

例 11. *pw(8)* によるグループへのユーザ追加

```
# pw groupmod teamtwo -m db
# pw groupshow teamtwo
teamtwo:*:1100:jru,db
```

この例では、-m の引数は、カンマで区切られたグループに追加するユーザの一覧です。前の例と異なり、これらのユーザはグループに追加され、既存のグループのユーザを置き換えることはありません。

例 12. グループに所属しているユーザを調べるための *id(1)* の使い方

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

この例では、jru は jru グループと teamtwo グループのメンバです。

このコマンドや /etc/group のフォーマットの詳細については、 pw(8) および group(5) をご覧ください。

3.4. 許可属性

FreeBSD では、すべてのファイルおよびディレクトリは一組の許可属性を持っています。これらの許可属性は、ユーティリティを使って確認したり変更できます。
許可属性がどのように機能するかを知ることで、ユーザが必要とするファイルにアクセスできるかどうか、オペレーティングシステムが使用しているファイルや他のユーザが所有するファイルにアクセスできないことを理解できるようになります。

この節では、FreeBSD で使用される伝統的な UNIX® の許可属性について説明します。より細かいファイルシステムのアクセス制御に関しては、アクセス制御リスト をご覧ください。

UNIX® では、基本の許可属性は 3 つのアクセスタイプ (読み・書き・実行) を使って割り当てられます。これらのアクセスタイプを使って、ファイルの所有者 (owner)、グループ (group) その他 (others) に対するファイルアクセスを設定します。 読み、書き、実行に関する許可属性は、それぞれ r, w, および x 文字で表されます。 これらの許可属性を表す際に、オンかオフ (0) による 2 進数表記も使われます。数字で表現する場合には、 r は 4、w は 2 そして x は 1 の値を持つよう、rwx の順番で表されます。

以下は、許可属性を表す際に用いられる数字およびアルファベットをまとめた表です。
"ディレクトリの表示" カラムでは、- は許可属性がオフに設定されていることを表します。

表 3. *UNIX®* 許可属性

値	許可属性	ディレクトリの表示
0	読み込み不可、書き込み不可、実行不可	---
1	読み込み不可、書き込み不可、実行可能	--x
2	読み込み不可、書き込み可能、実行不可	-w-
3	読み込み不可、書き込み可能、実行可能	-wx
4	読み込み可能、書き込み不可、実行不可	r--
5	読み込み可能、書き込み不可、実行可能	r-x

値	許可属性	ディレクトリの表示
6	読み込み可能、書き込み可能、実行不可	rw-
7	読み込み可能、書き込み可能、実行可能	rwx

コマンドライン引数 -l とともに ls(1) を使うと、詳細なディレクトリリストを見ることができ、ファイルの所有者、グループ、その他への許可属性を示す欄があるのがわかります。 例えば、ls -l を実行して、適当なディレクトリを表示させると以下のようになります。

```
% ls -l
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
```

myfile が含まれている行の一番目の列の最初の (一番左の) 文字は、そのファイルが普通のファイルなのか、ディレクトリなのか、キャラクタ型のデバイス特殊ファイルなのか、ソケットなのか、その他の特殊な疑似ファイルデバイスなのかといった種類を示す特別な文字です。 この例において、- という文字は、普通のファイルであることを示します。 その次に来る rw- と書かれた 3 文字は、そのファイルの所有者に許可を与えるものです。 その次の r-- の 3 文字は、そのファイルが所属しているグループに許可を与えます。 最後の r-- の 3 文字は、システムに存在するその他のユーザに許可を与えます。 "-" は許可が与えられていないことを示します。
この例では、ファイルの所有者はこのファイルを読み書きでき、ファイルの所属しているグループに属するユーザはファイルを読むことだけでき、そのどちらでもないユーザは、
このファイルを読むだけできるように許可属性が与えられています。
上の表によれば、このファイルに与えられた許可属性は 644 となります。
ここで各数字は、このファイルの許可属性の 3 つの部分を表しています。

デバイスの場合の許可属性はどのようにコントロールされているのでしょうか? FreeBSD は、大部分のハードウェアをファイルとして取り扱います。
そのため、プログラムからは普通のファイルとまったく同じようにオープンし、
データの読み書きができるようになっています。 これらのデバイス特殊ファイルは /dev/ に収められています。

ディレクトリもまた、ファイルと同様に扱われます。 それは読み込み/書き込み/実行の許可属性を持ちます。
ディレクトリの実行ビットはファイルのそれとは少し違った意味を持ちます。
ディレクトリが実行可能になっているとき、cd(1) を使ってそのディレクトリに移動することができます。
これは、そのディレクトリにあるファイルにアクセスできることを意味しています
(ファイル自体の許可属性によります)。

ディレクトリの中の一覧を表示するには、そのディレクトリに読み込み属性が設定されていなければなりません。 名前が分かっているファイルを削除するには、そのファイルが含まれているディレクトリに書き込み属性 と 実行属性 の両方が必要です。

この他にも許可属性ビットはありますが、いずれも setuid バイナリや sticky

ディレクトリなどといった特殊な状況で使われます。
ファイルの許可属性そのものについて、また、それらの設定方法に関する詳しい情報は、 chmod(1) マニュアルページを参照してください。

3.4.1. シンボリック表記

シンボリック表記と呼ばれる許可属性を表す方法では、ファイルやディレクトリの許可属性を、8 進数ではなく記号を用いて設定します。 シンボリック表記による許可属性を表す方法では、(who), (action), (permissions) という書式が用いられます。 利用できる値は以下の通りです。

オプション	文字	意味
(who)	u	ユーザ
(who)	g	ファイルを所持しているグループ
(who)	o	その他
(who)	a	すべて ("world")
(action)	+	許可属性を与える
(action)	-	許可属性を取り除く
(action)	=	許可属性を指定したものにする
(permissions)	r	読み込み
(permissions)	w	書き込み
(permissions)	x	実行
(permissions)	t	Sticky ビット
(permissions)	s	UID または GID を設定する

これらの値は、これまでと同様に chmod(1) で用いますが、数字ではなく文字で指定します。 たとえば、*FILE* に対して *FILE* のグループメンバーおよび自分以外のすべてのユーザからアクセスを一切受け付けたくない、というときには以下のコマンドを実行してください。

```
% chmod go= FILE
```

カンマ区切りで設定することで、ファイルの属性を一度に 2 つ以上変更できます。 以下の例では、*FILE* に対して自分以外のユーザから書き込みの権限を取り上げ、かわりにすべてのユーザが *FILE* を実行できるようにします。

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

3.4.2. FreeBSD のファイルフラグ

ファイルの許可属性に加え、FreeBSD では "ファイルフラグ" を使えます。 これはファイルにセキュリティや管理上の属性を追加するものですが、ディレクトリには追加しません。 ファイルフラグにより、`root` ユーザでさえ誤ってファイルを消去、変更してしまうことを防ぐことができます。

ファイルフラグは、chflags(1) を使って、簡単なインタフェースで設定できます。 例えば、file1 というファイルにシステムレベルで消去不可のフラグを設定するには、以下のコマンドを実行してください。

```
# chflags sunlink file1
```

消去不可のフラグを削除するには、以下のように sunlink の前に "no" をつけて実行してください。

```
# chflags nosunlink file1
```

ファイルに設定されているフラグを確認するには、-lo と一緒に ls(1) を実行してください。

```
# ls -lo file1
```

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

いくつかのファイルフラグの追加、削除は root ユーザしかできません。 他のフラグは、ファイルの所有者が変更できます。 chflags(1) と chflags(2) から、より詳細な情報を得ることをおすすめします。

3.4.3. setuid, setgid および sticky 許可属性

これまでに説明した許可属性のほかに、 すべての管理者が知っておくべき特別な設定が 3 つあります。 それは setuid, setgid および sticky 許可属性です。

これらの設定は、通常のユーザには許可されていない機能を提供するので、UNIX® の操作において重要となることがあります。 これらの許可属性を理解するためには、実ユーザ ID と実効ユーザ ID の違いに注意してください。

実ユーザ ID は、所有したりプロセスを開始する UID です。 実効 UID は、プロセスを実行するユーザ ID です。 たとえば、ユーザがパスワードを変更するときに利用する passwd(1) は、実ユーザ ID で起動します。 しかしながら、パスワードデータベースのアップデートの際は、実効 ID の root ユーザの権限で実行されます。 この仕組みにより、Permission Denied エラーが表示されることなく、ユーザはパスワードを変更できます。

setuid 許可属性は、以下の例で示されているように、ユーザに対して s の許可属性をつけることで設定できます。

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

setuid 許可属性は、以下の例で示されているように、指定する許可属性に数字の 4 をつけることでも設定できます。

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

これで suidexample.sh の許可属性は以下のように設定されます。

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

s は、許可属性のファイル所有者の実行可能ビットに置き換わって反映されます。
この設定により、passwd(1) といったユーティリティが権限を昇格することができます。

nosuid mount(8)
オプションを使うと、このようなバイナリがユーザへの警告なしに権限を昇格できないように設定できます。 ただし nosuid
ラッパにより回避できるため、このオプションを完全には信頼できません。

リアルタイムに確認するために、2 つのターミナルを開いてください。 1 つのターミナル上で、通常のユーザ権限で passwd と入力してください。
パスワードの入力を待つ間に、もう一つのターミナル上で、プロセステーブルおよび passwd(1) のユーザ情報を確認してください。

ターミナル A:

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

ターミナル 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
```

通常のユーザ権限で passwd(1) を実行したにもかかわらず、実効 UID の root が使われています。

setgid 許可属性は setuid
許可属性と同様の機能を提供しますが、この許可属性はグループの設定を変更します。
この設定を行った上でアプリケーションまたはユーティリティを実行すると、プロセスを開始するユーザではなく、ファイルを所有するグループに対してこの許可属性を与えます。

記号を用いてファイルに setgid 許可属性を設定するには、chmod(1) で設定するグループに s の許可属性をつけて実行してください。

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

または、chmod(1) で設定する許可属性の先頭に 2 をつけて実行してください。

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

以下に示されるように、s がグループの許可属性に指定されています。

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

上記の例において、対象としているシェルスクリプトが実行可能なファイルであっても、シェルスクリプトは setuid(2) システムコールにアクセスできないため、実効ユーザ ID では実行されません。

setuid および setgid 許可属性ビットは、権限の昇格を許可するので、システムのセキュリティレベルを下げます。 一方 3 番目の特殊な許可属性 sticky bit は、システムのセキュリティを強化します。

ディレクトリに sticky bit を設定すると、ファイルの所有者のみがファイルを削除できるようになります。 /tmp といった共有のディレクトリにおいて、ファイルの所有者以外のユーザがファイルを削除できなくなるので有用です。 この許可属性を有効にするには、ファイルに対して t モードを追加してください。

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

または、許可属性に 1 をつけて設定してください。

```
# chmod 1777 /tmp
```

sticky bit が設定されていると、許可属性の最後に t が表示されます。

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

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

3.5. ディレクトリ構造

FreeBSD のディレクトリ構造は、システム全体を理解するに当たって重要です。 最も重要なディレクトリは、ルートまたは "/" です。 このディレクトリは起動時に一番最初にマウントされ、オペレーティングシステムをマルチユーザで動作させるために必要なベースシステムが含まれています。 また、ルートディレクトリには、マルチユーザへの移行中に他のファイルシステムをマウントするためのマウントポイントも含まれます。

マウントポイントとは、追加するファイルシステムを接続する先の親のファイルシステム (普通はルートファイルシステム) のディレクトリのことです。 より詳細な説明は ディスク構成 の節にあります。 標準的なマウントポイントには /usr/, /var/, /tmp/, /mnt/ および /cdrom/ があります。 通常これらのディレクトリについては、/etc/fstab というファイル中のエントリが参照されます。 このファイルは、さまざまなファイルシステムとマウントポイントの表であり、システムが参照します。 /etc/fstab に書かれたファイルシステムは noauto オプションが指定されていなければ、起動時に rc(8)

スクリプトによって自動的にマウントされます。 詳細は fstab ファイル の節をご覧ください。

ファイルシステム構造を網羅した説明は hier(7) に書かれています。
以下の表は、もっともよく使われるディレクトリの簡単な概要です。

ディレクトリ	説明
/	ファイルシステムのルートディレクトリ
/bin/	シングルユーザ環境とマルチユーザ環境の両方で重要な ユーザユーティリティ
/boot/	オペレーティングシステムの起動時に使われるプログラムと設定ファイル
/boot/defaults/	デフォルトの起動設定ファイル; loader.conf(5) 参照
/dev/	devfs(4) により管理されるデバイスファイル
/etc/	システム設定ファイルとスクリプト
/etc/defaults/	デフォルトのシステム設定ファイル; 詳細については rc(8) 参照
/etc/periodic/	cron(8) 経由で毎日・毎週・毎月実行されるスクリプト; 詳細については periodic(8) 参照
/lib/	/bin および /sbin にあるバイナリで必要とされる重要なシステムライブラリ
/libexec/	重要なシステムファイル
/media/	CD, USB ドライブおよびフロッピーディスクなどのリムーバブルメディアのマウントポイントとして使用されるサブディレクトリを含むディレクトリ
/mnt/	システム管理者が一時的なマウントポイントとしてよく使う空のディレクトリ
/net/	自動マウント NFS 共有。auto_master(5) を参照
/proc/	プロセスファイルシステム; 詳細については procfs(5) と mount_procfs(8) 参照
/rescue/	rescue(8) で説明されている緊急時のために静的にリンクされているプログラム
/root/	root アカウントのホームディレクトリ
/sbin/	シングルユーザ環境とマルチユーザ環境の両方で重要なシステムプログラムと管理ユーティリティ
/tmp/	システムの再起動では通常保存 されない 一時的なファイル。 メモリファイルシステムはよく /tmp にマウントされます。 これは rc.conf(5) の tmpmfs 関係の変数を使うか、/etc/fstab に設定項目を記入することで自動化できます。 詳しくは mdmfs(8) を参照して下さい。
/usr/	大部分のユーザユーティリティとアプリケーション
/usr/bin/	よく使うユーティリティとプログラミングツールとアプリケーション
/usr/include/	C の標準ヘッダファイル
/usr/lib/	ライブラリ
/usr/libdata/	いろいろなユーティリティのデータファイル
/usr/libexec/	他のプログラムから実行されるシステムデーモンとシステムユーティリティ

/usr/local/	ローカルのプログラムやライブラリなど。 FreeBSD ports フレームワークのデフォルトインストール先としても使われます。 /usr/local 内では、 hier(7) に書かれている /usr のための一般構造が使われます。 例外は man ディレクトリで、/usr/local/share の下ではなく /usr/local の下に直接置かれ、ports 関係文書は share/doc/port に置かれます。
/usr/ports/	FreeBSD Ports Collection (オプション)。
/usr/sbin/	ユーザにより実行されるシステムデーモンおよびシステムユーティリティ
/usr/share/	アーキテクチャに依存しないファイル
/usr/src/	BSD のソースファイルまたはローカルのソースファイル、あるいは両方
/var/	さまざまな用途のログ・一時的なファイル・スプールファイル。
/var/log/	いろいろなシステムログファイル
/var/tmp/	一時的なファイル。通常の設定では、ここにあるファイルはシステムが再起動しても失われません。

3.6. ディスク構成

ファイルを見つけるために FreeBSD が使用する構成の一番小さな単位はファイル名です。 ファイル名は、大文字と小文字を区別します。 このことは readme.txt および README.TXT が異なる二つのファイルであることを意味します。 FreeBSD はそのファイルがプログラム、または文書、あるいはその他の形式かどうかを決定するために拡張子を使用しません。

ファイルはディレクトリ内に格納されます。 ディレクトリはファイルを一つも含んでいないかもしれませんし、または数百のファイルを含んでいるかもしれません。 ディレクトリはまた別のディレクトリを含むことができるので、データを体系づけるディレクトリの階層構造を構築できます。

ファイルおよびディレクトリは、必要な他のディレクトリ名とスラッシュ (/) を後に続けてファイル名またはディレクトリ名を与えることによって参照されます。 たとえば、foo ディレクトリがあって、その中に bar ディレクトリがあるとします。 そして、その中に readme.txt があるとすると、ファイルへのフルネーム、または パス は foo/bar/readme.txt となります。 ファイルとディレクトリ名を分けるために \ を使う Windows® とは違うことに注意してください。 FreeBSD は、パスの中にドライブレターまたは他のドライブ名を使いません。 たとえば、FreeBSD では c:\foo\bar\readme.txt とは書きません。

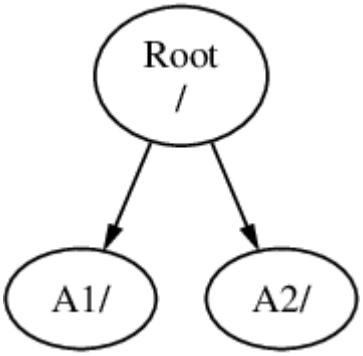
3.6.1. ファイルシステム

ディレクトリおよびファイルはファイルシステム内に格納されます。 どのファイルシステムも、そのファイルシステムのための ルートディレクトリ とよばれる、まさに頂点の位置にちょうど一つのディレクトリを含んでいます。 このルートディレクトリは他のディレクトリを含むことができます。 一つのファイルシステムは ルートファイルシステム または / として設計されています。 すべてのファイルシステムは、ルートファイルシステム以下に マウント されます。 FreeBSD システムでどんなに多くのディスクを使用しても、すべてのディレクトリは、同じディスクの一部であるように見えるので問題ありません。

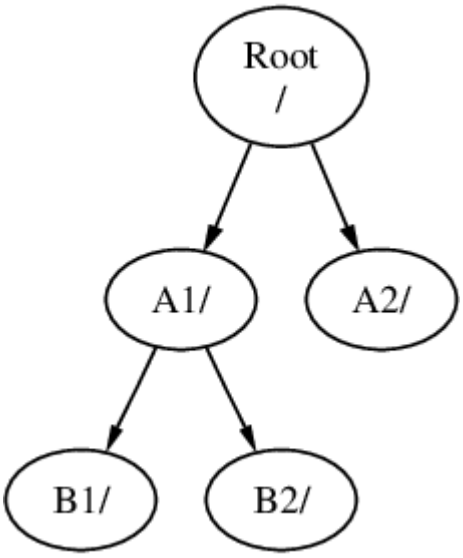
A, B および C と呼ばれる三つのファイルシステムがあるケースを考えます。

それぞれのファイルファイルシステムには一つのルートディレクトリがあり、A1, A2 と呼ばれている二つの他のディレクトリを含んでいます (同様に B1, B2 および C1, C2 があります)。

A をルートファイルシステムとします。 このディレクトリになにが含まれているか見るために ls(1) コマンドを使うと、A1 および A2 の二つのサブディレクトリが表示されるでしょう。ディレクトリツリーは以下のようになります。

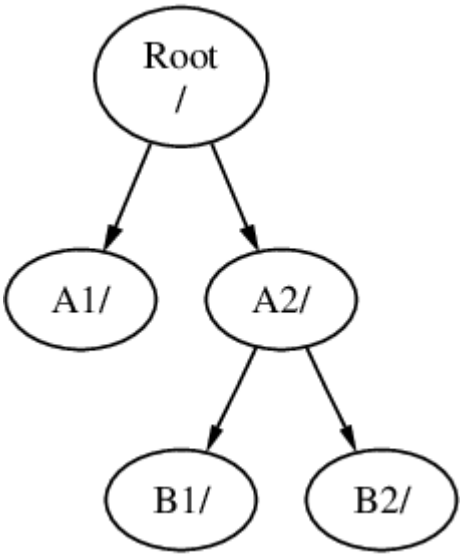


ファイルシステムはマウント先のファイルシステム内のディレクトリにマウントしなければいけません。それでは、A1 ディレクトリに B ファイルシステムをマウントすると仮定します。 B のルートディレクトリは A1 に置き換えられ、そして B 内のディレクトリがそれに応じて現れます。



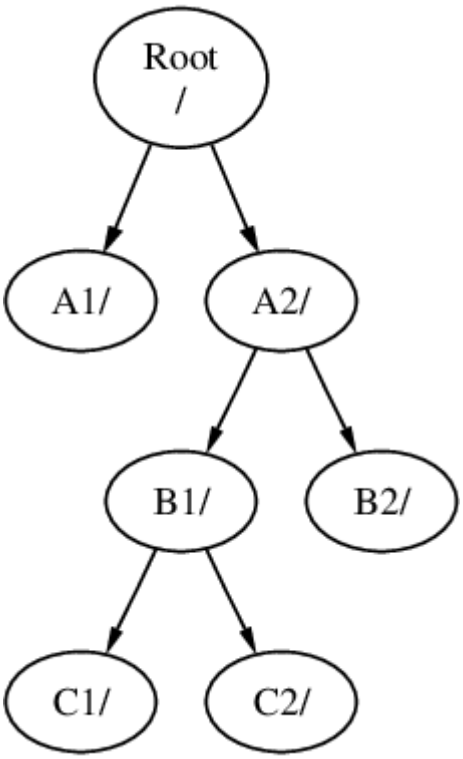
B1 または B2 内にあるどんなファイルも、必要なときに /A1/B1 または /A1/B2 で到達できます。 /A1 にあったすべてのファイルは一時的に隠されました。 それらは B が A から アンマウントされたら再び現れるでしょう。

もし B が A2 にマウントされていたら、この図のようになります。

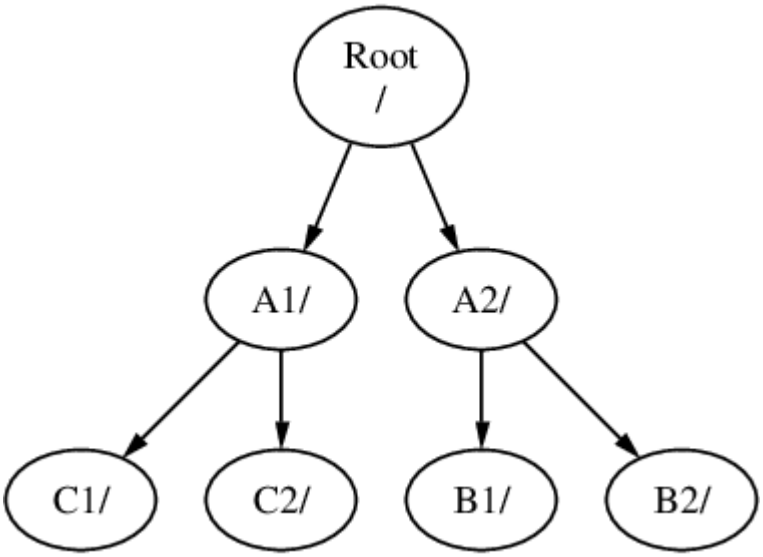


そして、パスはそれぞれ /A2/B1 および /A2/B2 となるでしょう。

ファイルシステムは互いのファイルシステム上にもマウントできます。 上記の最後の例に続けて、C ファイルシステム は B ファイルシステム内の B1 ディレクトリ上にマウントできます。次の図のようになります。



または C を A ファイルシステムの A1 ディレクトリの下に直接マウントできます。



一つの大きなルートファイルシステムを用意し、他のファイルシステムを作成する必要としないことはまったくもって可能です。 この方法にはいくつかの短所と一つの利点があります。

マルチファイルシステムの利点

- 異なったファイルシステムは異なった マウントオプション を使用できます。
  たとえば、ルートファイルシステムを読みだし専用でマウントして、不注意によってユーザが重大なファイルを削除、または編集できないようにすることができます。 また、`/home` のようなユーザが書き込み可能なファイルシステムを他のファイルシステムと分けることによって、*nosuid* でマウントすることが可能になります。
  このオプションは、ファイルシステムに記録されている *suid/guid* の実行可能ビットを有効にしないので、安全性を高めることができるでしょう。
- FreeBSD
  はファイルシステムがどのように使われているかによって、自動的にファイルシステム上のファイルの配置を最適化します。
  したがって、連続的に書き込まれた多くの小さなファイルが含まれているファイルシステムは、より大きく少ないファイルが含まれているファイルシステムと異なる最適化をするでしょう。
  一つの大きなファイルシステムを作成すると、この最適化は成り立たなくなります。
- FreeBSD のファイルシステムはトラブルが起きても強固です。
  しかしながら臨界点でのトラブルは、ファイルシステムの構造にまだ損害を与えるかもしれません。
  マルチファイルシステムへデータを分割しておくことで、
  必要なときにバックアップからレストアすることをより容易にして、まだシステムが回復するかもしれません。

シングルファイルシステムの利点

- ファイルシステムは固定サイズです。 FreeBSD をインストールするときにファイルシステムを作成して、 固定サイズを割りあてたなら、後になってそのパーティションをより大きくする必要があると気づくかもしれません。
  パーティションのサイズを変更するには、
  バックアップ、新しいサイズを指定したファイルシステムの再作成、
  バックアップしたデータをリストアする作業が必要となるでしょう。

FreeBSD には、 growfs(8) コマンドがあります。

このコマンドは、この制限を取り除いて、ファイルシステムのファイルを直ちに増加させることを可能にします。
ファイルシステムは、そのファイルシステムのあるパーティションの空いている領域に対してのみ拡張できます。
パーティションを分割した後、空いている領域があれば、gpart(8) を用いてパーティションを拡張できます。
仮想ディスクの最後のパーティションであれば、ディスクを大きくすると、パーティションを拡張できます。

3.6.2. ディスクパーティション

ファイルシステムは パーティション 内に含まれています。 ディスクは 1 つのパーティションスキーム (Manual によるパーティションの分割) を用いてパーティションに分割されます。 新しいスキームは GPT で、古い BIOS-ベースのコンピュータは MBR を使用します。 GPT は、サイズ、オフセットおよびタイプによるディスクのパーティション分割に対応しています。
多くのパーティションおよびパーティションタイプに対応しているため、GPT が利用できる場合はこのパーティションスキームを使用することが推奨されます。 GPT パーティションは、接尾語 p1 が最初のパーティション、接尾語 p2 が 2 番目のパーティションといったような接尾語を使います。 一方 MBR パーティションは少ない数のパーティションにのみ対応しています。 MBR パーティションは、FreeBSD では スライス として知られています。
スライスは他のオペレーティングシステムでも使うことができます。 FreeBSD のスライスはさらに、BSD ラベル (bsdlabel(8) 参照) を用いてパーディションに分割できます。

スライス番号は 1 から始まり s を前につけられて、デバイス名の後に続きます。 したがって、"da0*s1*" は一番目の SCSI ドライブ上の一番目のスライスです。 ディスク上に存在できる物理スライスは、4 つまでですが、適切な種類の物理スライス内に論理スライスを作成できます。
これらの拡張されたスライス番号は 5 から始まります。 したがって、 "ada0*s5*" は、一番目の SATA ディスク上の一番目の拡張スライスです。
これらのデバイスは、スライスを占有することを予期するファイルシステムによって使用されます。

GPT または BSD の各パーティションは、一つのファイルシステムだけを含むことができます。 このことは、ファイルシステムがファイルシステムの階層上の典型的なマウントポイント、または含まれているパーティション名によって記述されることを意味します。

FreeBSD は スワップ領域 にもまたディスク領域を使用します。 スワップ領域は FreeBSD に 仮想メモリを提供します。 これはあなたのコンピュータが、実際に搭載している以上のメモリがあるかのように振舞います。 FreeBSD がメモリを使い果たしたときに、現在使用されていないデータのいくつかをスワップ領域に移動し、そのデータが必要となったときに (その他のデータをスワップ領域に移動させてから) メモリ内に移動しなおします。 これは ページング と呼ばれます。

いくつかの BSD パーティションはある慣習と関係づけられています。

パーティション	慣習
a	通常、ルートパーティションを含みます。
b	通常、スワップ領域を含みます。

c	通常、スライス全体と同じサイズです。 これは、スライス全体にアクセス必要のあるユーティリティ (たとえば、ひどいブロックスキャナ) が、 c パーティションにアクセスすることを可能にします。 通常、このパーティション内にファイルシステムは作成されません。
d	d パーティションは、 それに関連づけられた特別な意味を持っていましたが、 今は無いので、普通のパーティションとして動作するでしょう。

スライスおよび "危険な専用" の物理ドライブ、 そして他のドライブは a から h までの文字として表される BSD パーティションを含んでいます。
この文字はデバイス名に追加されます。 したがって、 "da0*a*" は一番目の "危険な専用" da ドライブ上の a パーティションです。 "ada1s3*e*" は、 二番目の SATA ディスク上の三番目のスライス内にある五番目のパーティションです。

最後に、システム上のそれぞれのディスクは識別されます。
ディスク名はどの種類のディスクであるかを示す記号ではじまり、どのディスクかを示す数字が続きます。 パーティションやスライスとは異なり、ディスクの番号づけは 0 から始まります。 共通の記号は ディスクデバイス名 に示されます。

スライスにあるパーティションを参照するときには、ディスク名、s、スライス番号、そしてパーティション文字を含めてください。 ディスク名、スライス名、パーティション名のサンプル に例があります。
GPT パーディションはディスク名、p そしてパーティション番号が含まれます。

ディスクの概念的構成 は、MBR スライスを用いたディスク構成の概念のモデルを示します。

FreeBSD をインストールする際には、MBR を使用する場合にはディスクスライスを設定し、次に FreeBSD に用いるスライス内のパーティションを作成します。 GPT を使用する場合には、各ファイルシステムにパーティションを設定します。
どちらのケースでも、それぞれのパーティション内にファイルシステムまたはスワップ領域を作成し、ファイルシステムがどこにマウントされるか決定してください。 パーティションの操作についての詳細は gpart(8) をご覧ください。

表 4. ディスクデバイス名

ドライブタイプ	ドライブデバイス名
SATA および IDE ハードドライブ	ada
SCSI ハードドライブおよび USB ストレージデバイス	da
NVMe ストレージ	nvd または nda
SATA および IDE CD-ROM ドライブ	cd
SCSI CD-ROM ドライブ	cd
フロッピードライブ	fd
SCSI テープドライブ	sa
RAID ドライバ	aacd (Adaptec® AdvancedRAID), mlxd および mlyd (Mylex®), amrd (AMI MegaRAID®), idad (Compaq Smart RAID), twed (3ware® RAID) など

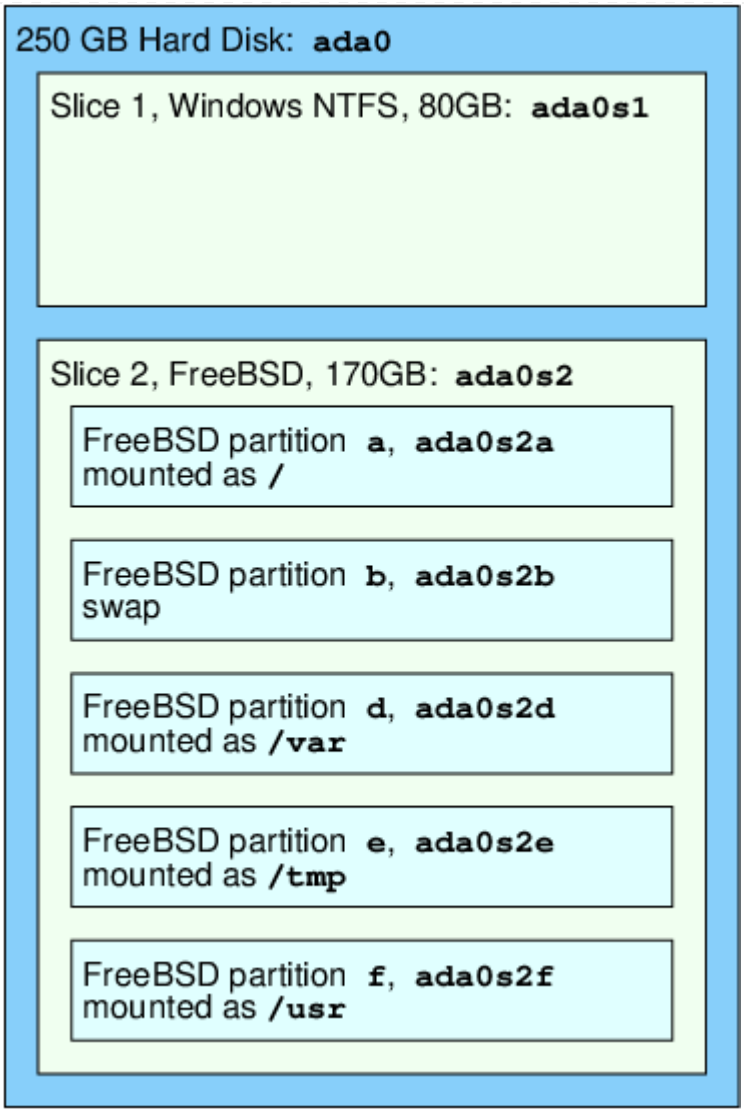
表 5. ディスク名、スライス名、パーティション名のサンプル

名前	意味
ada0s1a	一番目の SATA ディスク (ada0) 上の一番目のスライス (s1) 内の一番目のパーティション (a)。
da1s2e	二番目の SCSI ディスク (da1) 上の二番目のスライス (s2) 内の五番目のパーティション (e)。

例 13. ディスクの概念的構成

これはシステムに接続された一番目の SATA ディスクの FreeBSD から見た図を示します。ディスクサイズは 250 GB と仮定し、80 GB のスライス (MS-DOS® でいうパーティション) および 170 GB のスライスがあるとします。 一番目のスライスは Windows® NTFS ファイルシステム C: を含んでいます。 そして、二番目のスライスは FreeBSD のディスクを含んでいます。 これは FreeBSD
インストーラが四つのデータパーティションと一つのスワップパーティションを作成した例です。

四つのパーティションはそれぞれファイルシステムを含んでいます。 パティション a はルートファイルシステム、d は /var, e は /usr、そして f は /usr に使用されています。
パーティション c
はスライス全体を示しており、通常のパーティションとは異なる使われ方をします。



3.7. ファイルシステムのマウントとアンマウント

ファイルシステムは / をルート (根) とする木構造として考えると視覚的に理解しやすいでしょう。ルートディレクトリにある /dev や /usr、その他のディレクトリは枝に相当し、それらには、/usr/local などのように、さらに枝分かれすることができます。

さまざまな理由がありますが、
ディレクトリをいくつかの異なるファイルシステム上に構築するのが良いでしょう。 たとえば /var には、log/ や spool/ など、さまざまな種類の一時ファイルを置くディレクトリがあるため、あふれてしまう可能性があります。 ルートファイルシステムをあふれさせるのは得策ではありませんので、普通は /var を / から分離します。

また、次のような場合も、ディレクトリツリーを別のファイルシステムに置く理由として良くあげられます。 それは、たとえば物理的に別のディスクにディレクトリツリーを置く場合、「ネットワークファイルシステム (NFS)」
で説明されているようにネットワークファイルシステムをマウントしたり、CDROM
ドライブのような別の仮想ディスクに置くという場合です。

3.7.1. fstab ファイル

/etc/fstab に書かれているファイルシステムは、noauto オプション指定されているエントリを除いて起動プロセス の途中で自動的にマウントされます。 このファイルは、次のような書式で書かれたエントリを含んでいます。

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

device
: デバイス名。「デバイス名」 に説明があります。

mount-point
: ファイルシステムがマウントするディレクトリ。

fstype
: mount(8) に渡されるファイルシステムタイプ。 FreeBSD ファイルシステムのデフォルトは ufs です。

options
: 読み書きするファイルシステムには rw、読み込み専用のファイルシステムには ro を、必要な他のオプションの前に指定します。 よく使われるオプションは noauto で、起動時にはマウントされないファイルシステムに使います。 その他のオプションは mount(8) マニュアルページに載っています。

dumpfreq
: これは dump(8) が使うもので、 どのファイルシステムにダンプが必要なのかを決めます。この項目がなければ、0 であるものとみなされます。

passno
: これは再起動後に fsck(8) がチェックする UFS ファイルシステムの順番を決めます。ファイルシステムチェックを飛ばしたいファイルシステムには、passno を 0 に設定してください。ルートファイルシステムはどれよりも先にチェックする必要があり、passno は 1 に設定してください。 他のファイルシステムの passno は 1 以上に設定してください。 同じ passno のファイルシステムがあった場合、 fsck(8) は可能であれば並行してファイルシステムのチェックを行なおうとします。

/etc/fstab の書式やオプションに関しての詳細は、 fstab(5) をご覧ください。

3.7.2. mount(8) の使い方

ファイルシステムは mount(8) を用いてマウントされます。 基本な構文は以下のようになります。

```
# mount device mountpoint
```

/etc/fstab
に記載されているファイルシステムについても、マウントポイントを指定することでマウントできます。

mount(8) で説明されているように、このコマンドはたくさんのオプションを提供します。最もよく使われるのは次のものです。

マウントオプション

`-a`

`/etc/fstab` にある全てのファイルシステムをマウントします。例外は "noauto" の印がついているものと、`-t` フラグで除外されたものと、すでにマウントされているファイルシステムです。

`-d`

実際にマウントシステムコールする以外のすべてのことをします。このオプションは `-v` フラグと組み合わせて使い、mount(8) が実際なにをしようとしているのか調べるのに便利です。

`-f`

クリーンでないファイルシステムを強制的にマウントします (危険です)。もしくは、ファイルシステムのマウント状態を読み書き可能から読み込みのみに変更するとき、書き込みアクセスを強制的に取り消します。

`-r`

ファイルシステムを読み込み専用でマウントします。`-o ro` を使うのと同じです。

`-t fstype`

指定のファイルシステムタイプでマウントします。または、`-a` を使った場合、指定したタイプのファイルシステムのみマウントします。デフォルトのファイルシステムタイプは "ufs" です。

`-u`

ファイルシステムのマウントオプションを更新します。

`-v`

詳細な出力にします。

`-w`

ファイルシステムを読み書き可能にマウントします。

`-o` には、次のようなオプションを複数カンマで区切って指定できます。

nosuid

そのファイルシステム上の setuid や setgid フラグを解釈しません。これもセキュリティのために有用なオプションです。

3.7.3. umount(8) の使い方

ファイルシステムをアンマウントするには、umount(8) を使ってください。このコマンドは、パラメータとしてマウントポイントの一つ、デバイス名、もしくは `-a` や `-A` といったオプションを取ります。

いずれの形式でも `-f` で強制的なアンマウントを行ない、`-v` で詳細な出力を出します。ただしほとんどの場合、`-f` は使わないほうがよいでしょう。

計算機がクラッシュしたりファイルシステム上部のデータが破壊されたりする恐れがあります。

マウントされているファイルシステムすべてをアンマウントするには、`-a` と `-A` を使ってください。 `-t` にファイルシステムタイプを指定すると、指定されたものだけがアンマウントされます。 `-A` を使うとルートファイルシステムはアンマウントしません。

3.8. プロセスおよびデーモン

FreeBSD はマルチタスクのオペレーティングシステムです。 動作中のプログラムはそれぞれ プロセス と呼ばれます。 すべてのコマンドは実行すると、最低でも 1 つの新しいプロセスを開始します。 FreeBSD により実行されているシステムプロセスもたくさんあります。

各プロセスは プロセス *ID* (PID) と呼ばれる数字でただ一つに識別されます。 ファイルのように各プロセスには所有者とグループがあり、 所有者とグループの許可属性は、そのプロセスが開けるファイルやデバイスを決定するために使われます。 多くのプロセスには親プロセスもあります。 親プロセスとは、そのプロセスをスタートさせたプロセスのことです。 例えば、シェルがプロセスで、シェルから起動されるコマンドは、シェルを親プロセスとするプロセスとなります。 例外は init(8) という特別なプロセスです。 `init` は FreeBSD がスタートするときに起動される最初のプロセスで、PID は常に 1 です。

ユーザから始終入力があるように設計されていないプログラムがあり、そういったプログラムは最初から端末と切り離されています。 例えば、ウェブサーバはユーザからの入力ではなくウェブのリクエストを処理します。 メールサーバも、こういった種類のアプリケーションの一例です。 このような種類のプログラムは、 デーモン と呼ばれます。 デーモンはギリシャ神話から来ており、目に見えないように役立つことをしてくれる善でも悪でもない実体を表します。 このため、BSD のマスコットはスニーカーをはいてフォークを携えたかわいらしい姿のデーモンなのです。

通常デーモンとして動作するプログラムには末尾に "d" を持った名前をつける慣習があります。 例えば、BIND は Berkeley Internet Name Domain ですが、 実際実行されるプログラムは `named` です。 また、Apache ウェブサーバのプログラムは `httpd`、ラインプリンタスプーリングデーモンは `lpd` です。 これは単なる命名に関する慣習です。 例えば、Sendmail アプリケーションの主なメールデーモンは `sendmail` で、`maild` ではありません。

3.8.1. プロセスを確認する

システム上で実行中のプロセスを確認するには、ps(1) または top(1) を使ってください。 現在動作中のプロセスのリスト、プロセスの PID やプロセスが使っているメモリの量、どういうコマンドラインで起動されたのかなどを表示させるには、ps(1) を使ってください。 top(1) を使用すると、動作中の全てのプロセスを表示できます。 数秒ごとに表示を更新するので、計算機が何をしているのかインタラクティブに知ることができます。

デフォルトでは、ps(1) はユーザにより動作中かつ所有のコマンドのみを表示します。 例えば:

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

ps(1) の出力はいくつかの列に整形されています。 PID の列はプロセス ID を表示します。 PID は 1 から順に 99999 まで割り当てられ、その後足りなくなると最初に戻って使い回されます。ただし、使用中の PID には割り当てられません。 TT の列はプログラムが動いている tty を示し、STAT はプログラムの状態を示します。 TIME はプログラムがその CPU 上で動いている時間の長さです。通常はプログラムをスタートさせたときからの経過時間ではありません。 多くのプログラムは、CPU 上で時間を使う必要があるまでかなりの時間を費すためです。 最後に、COMMAND はそのプログラムを起動するのに使われたコマンドとなります。

表示する情報を変更するオプションが用意されています。 いちばん便利なのは auxww でしょう。 a はすべてのユーザの動作中のプロセス全部についての情報を表示します。 u はプロセスの所有者のユーザ名とメモリ使用量を表示します。 x はデーモンプロセスについての情報を表示し、ww で、スクリーンに入りきらないほど長くなったコマンドラインでも省略せず、ps(1) に各プロセスの全コマンドラインを表示させます。

top(1) の出力も同様です。

```
% top
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
```

出力は2つのセクションに分かれています。 ヘッダ (最初の 5 または 6 行) は動作している最新のプロセスの PID、システムの平均負荷 (システムがどれくらい忙しいかの指標)、システムの稼働時間 (最後の再起動からの時間) と現在の時刻を示します。ヘッダの中の他の数字は動作中のプロセスの数、使われているメモリとスワップ領域の量、そしてシステムが異なる CPU 状態に消費した時間と関係します。 ZFS ファイルシステムのモジュールをロードしている場合には、ARC 行にはディスクではなくメモリキャッシュから読み込んだデータ量が表示されます。

ヘッダの下には、PID、ユーザ名、消費 CPU 時間とプロセスを起動したコマンドといった ps(1) の出力と同じような情報を持った行が続きます。 top(1)

を使うとデフォルトでプロセスが使っているメモリ容量を表示します。 メモリ使用量の欄は 2 項目に分かれており、 一方は合計使用量、 そしてもう一方は実使用量です。
合計使用量はアプリケーションが必要としているメモリ量で、実使用量はその時点で実際に使われているメモリ量です。

top(1) は自動的に 2 秒ごとに画面を更新します。 -s 使うと更新間隔を変更することができます。

3.8.2. プロセスの終了

動作中のプロセスもしくはデーモンと通信する一つの方法は、kill(1) を用いて シグナルを送信する方法です。 送信可能なシグナルはたくさんあります。
特別な意味があるものもあれば、アプリケーションの文章に説明されているものもあります。
ユーザは自分が所有者となっているプロセスにのみシグナルを送ることができます。
他人のプロセスにシグナルを送ると、permission denied というエラーになるでしょう。 この例外は root ユーザで、 ルートユーザは誰のプロセスに対してもシグナルを送ることができます。

オペレーティングシステムもプロセスにシグナルを送ることができます。
アプリケーションを下手に書いてしまい、予想外のメモリにアクセスしようとすると、FreeBSD はプロセスに "セグメンテーション違反" シグナル (SIGSEGV) を送ります。 ある程度の時間が経ったら alarm(3) システムコールを使って警告してもらうように書かれているアプリケーションには、"警告" シグナル (SIGALRM) が送信されます。

プロセスを止めるためには2つのシグナル、SIGTERM か SIGKILL を使います。 SIGTERM は穏かにプロセスを終了させる方法です。
プロセスはシグナルを受け取ることができ、開いているすべてのログファイルを閉じ、終了前にしていたことを終えるように試みることができます。 中断できない処理の途中だと、SIGTERM をプロセスが無視することもあるかもしれません。

プロセスは SIGKILL を無視することができません。 プロセスに SIGKILL を送ると、プロセスは通常その時点で止まります。

他に良く使われるシグナルには、SIGHUP、SIGUSR1 と SIGUSR2 があります。
これらは一般的な用途のシグナルなので、このシグナルが送信されたときの応答は、アプリケーション毎に異なります。

例として、ウェブサーバの設定ファイルを変更後、ウェブサーバに設定を再読み込みさせる必要があります。 httpd を再起動するとウェブサーバは一瞬ながら停止してしまいます。 その代わりに SIGHUP シグナルを送りましょう。 デーモンごとに行動が違うので、SIGHUP が期待する結果となるように、そのデーモンの文書を読んで確認してください。

システム上のランダムプロセスを終了させるのはよくありません。 特に、PID が 1 の init(8) は特別です。 /bin/kill -s KILL 1 は推奨されていませんが、実行するといとも簡単にシステムをシャットダウンさせることができます。 Return を押す 前 に kill(1) を実行する引数を二重にチェックする 癖をつけてください。

3.9. シェル

シェル は、オペレーティングシステムを利用するためのコマンドラインインタフェースを提供します。
シェルは入力チャンネルからコマンドを受け取り、それらを実行します。
大部分のシェルは、日々の作業、ファイル管理やファイル名の展開、コマンドライン編集、コマンドマク

口、環境変数といった組み込みの機能を持ってます。 FreeBSD には Bourne Shell (sh(1)) や 高機能 C-shell (tcsh(1)) が含まれています。 また、これ以外にも zsh や bash などのシェルが FreeBSD Ports Collection から利用可能です。

どのシェルを使うかは、まったく趣味の問題です。 あなたが C のプログラマだったとすれば、tcsh(1) のような C 風のシェルの方が落ち着くかもしれません。 Linux® ユーザであれば、bash を好まれるでしょう。 それぞれのシェルは、 ユーザの好みの作業環境で利用できる (もしくはできない) 独自の機能を持っているということ、そして、どのシェルを使うことにするかを決めるのはユーザ自身ということです。

シェルの一般的な機能の一つに、ファイル名の補完があります。
コマンドやファイル名の最初の数文字を入力して Tab
を押すと、シェルにコマンドやファイル名の残りの部分を補完させることができます。 例として、foobar および footbar という二つのファイルがあるとします。 foobar を削除するために rm foo と入力し、Tab を押してファイル名を補完しようとします。

しかしシェルは rm foo とだけ出力します。 foobar および football のファイル名は、両方とも foo から始まるため、ファイル名の補完を完全には行なえませんでした。
一つ以上のファイル名にマッチした場合、ビープ音をらすシェルもあれば、選択できるすべてのファイル名を表示するシェルもあります。
この場合、希望するファイル名を同定するために、ユーザはさらに文字を入力する必要があります。 t を入力してもう一度 Tab
を押すと、シェルはファイル名を確定でき、ファイル名の残りの部分が補完されます。

もう一つあげられるシェルの特徴として、環境変数があります。
環境変数とは、シェルの環境変数におけるキーと値とのペアです。
この環境変数は、そのシェルから起動されたプログラムから参照でき、それを利用してプログラムの設定を保存するのに利用されます。 一般的な環境変数 は、一般的な環境変数とその意味の一覧です。
環境変数の名前は常に大文字です。

表 6. 一般的な環境変数

変数名	意味
USER	現在のログインユーザのユーザ名。
PATH	コロンで区切られた実行ファイル探索のための ディレクトリのリスト。
DISPLAY	接続する Xorg ディスプレイのネットワーク名 (存在する場合のみ)。
SHELL	現在のシェル。
TERM	ユーザの端末種名。 端末のケーパビリティを決定するのに使われる。
TERMCAP	種々の端末の機能を実現する端末のエスケープコードの データベースのエントリ。
OSTYPE	オペレーティングシステムの種別。
MACHTYPE	システムの CPU アーキテクチャ。
EDITOR	ユーザの選んだテキストエディタ。
PAGER	ユーザの選んだ画面上でテキストを見るためのユーティリティ。
MANPATH	コロンで区切られたマニュアルページ探索のための ディレクトリのリスト。

環境変数を設定する方法は、シェルごとに多少異なります。 tcsh(1) や csh(1) では setenv を使います。 sh(1) や bash 等の Bourne シェルでは、export を使って現在の環境変数を設定します。

以下の例では、tcsh シェルでデフォルトの EDITOR を /usr/local/bin/emacs に設定します。

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

bash では次のようになります。

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

現在の設定を確認するために、コマンドライン中の変数名の前に $ 文字を置くことで、環境変数を展開させることができます。 たとえば、echo $TERM は $TERM がセットされている内容を表示します。

シェルは特殊文字を、特別なデータを表すものとして扱います。
その特殊文字はメタキャラクタと呼ばれます。 もっとも一般的なメタキャラクタは * で、これはファイル名に含まれる、あらゆる文字を表します。
メタキャラクタはファイル名の展開に使われます。 たとえば、echo * と入力すると ls と入力したのとほとんど同じ結果を得られます。 これはシェルが * とマッチするすべてのファイルを受け取って echo はコマンドラインでそれらを表示するからです。

特殊文字をシェルに解釈させないようにするため、特殊文字の前にバックスラッシュ文字 (\) を置いてエスケープしてください。 例えば echo $TERM は端末の設定を表示し、echo \$TERM は $TERM とそのまま表示します。

3.9.1. シェルの変更

デフォルトのシェルを変更する一番簡単な方法は chsh を使うことです。 このコマンドを実行すると、環境変数 EDITOR で示されたエディタ (デフォルトでは vi(1) が設定されている) が立ち上がります。 Shell: の行を変更するシェルの絶対パスに変更してください。

代わりに chsh -s を使うと、エディタを起動せずにシェルを変更できます。 たとえば、シェルを bash に変えたいなら、次のようにしてください。

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

プロンプトに対してパスワードを入力し、Return を押すと、シェルが変更されます。 新しいシェルを使うには、一度ログオフしてから再ログインしてください。

使おうと思っているシェルは必ず /etc/shells 中に書かれていなければなりません。 シェルを アプリケーションのインストール - packages と ports で説明されている FreeBSD の Ports Collection からインストールしたのであれば、自動的にこのファイルに追加されています。 もし書かれていなければ、以下のコマンドで、パスをシェルのパスに置き換えて使って追加してください。

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

その後 chsh(1) を実行してください。

3.9.2. 高度なシェルの機能

UNIX®
のシェルは単なるコマンドインタプリタではなく、ユーザが実行したコマンドの出力をリダイレクトしたり、入力をリダイレクトすることによりコマンドをお互いに繋げることで、最終的なコマンドの出力結果を改良できます。
この機能をビルトインコマンドとともに用いることで、ユーザは最大化された効率の環境を入手できます。

シェルのリダイレクト機能を使うことで、コマンドの出力や入力を別のコマンドに送ったり、ファイルに送ることができます。 たとえば、 ls(1) コマンドの出力をキャプチャするには、出力をファイルにリダイレクトしてください。 以下はその例です。

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

実行すると、現在の作業ディレクトリにあるファイルの一覧が directory_listing.txt に出力されます。
sort(1) のようなコマンドは、入力を読み込むことができます。
先ほど得たファイルの一覧をソートするには、入力元をファイルにリダイレクトしてください。

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

入力された内容はソートされ画面に出力されます。
この出力を他のファイルにリダイレクトするには、リダイレクトの向きを混ぜるように sort(1) の出力をリダイレクトしてください。

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

これまでの例では、ファイルディスクリプタを用いてコマンドに対しリダイレクトを行っています。
すべての UNIX® システムは標準入力 (stdin)、標準出力 (stdout) および標準エラー (stderr) といったファイルディスクリプタを持っています。 それぞれに対象があり、入力はキーボードまたはマウスなどの入力を提供するものが対象、出力はスクリーンであったりプリンタ用紙が対象です。 また、エラーは診断やエラーメッセージに用いられるものが対象です。 これらは、I/O ベースのファイルディスクリプタ、時にはストリームと考えられます。

これらのディスクリプタを使用することで、シェルは出力と入力についてさまざまなコマンドを経由させ、また、ファイルに対して出力し、もしくはファイルから読み込むようにリダイレクトできます。
リダイレクトの他の方法は、パイプの機能です。

UNIX® のパイプ記号 "|" は、コマンドの出力を他のプログラムに直接渡します。
基本的には、パイプはコマンドの標準出力を他のコマンドの標準出力に渡します。 以下はその例です。

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

この例では、directory_listing.txt の内容がソートされ、その結果が less(1) に渡されます。
このコマンドを実行すると、出力がスクロールして画面から見えなくなることをさけることができて、ユーザは出力を自分のペースでスクロールできます。

3.10. テキストエディタ

FreeBSD の設定の多くは、テキストファイルの編集で行われます。
そのため、テキストエディタの扱いに慣れると良いでしょう。 FreeBSD には、基本システムの一部として二、三提供されるものと、Ports Collection から利用できる、たくさんのテキストエディタが用意されています。

学習が簡単なエディタは、easy editor の略で ee(1) と呼ばれるものです。
このエディタを立ち上げるには、`ee filename` と入力してください。 ここで *filename* は、編集しようとしているファイルの名前です。 一旦このコマンドの中に入れば、エディタの機能を操作するコマンドはすべてディスプレイの上部に表示されています。 キャレット (^) は `Ctrl` を意味するので、`^e` は `Ctrl` + `e` を押すという意味になります。 ee(1) を終了するには `Esc` を押し、そしてメインメニューから "leave editor" オプションを選択してください。
ファイルが更新されていたときは、エディタは変更をセーブするかどうかプロンプトを出します。

FreeBSD には、ベースシステムの一部として vi(1) といったより強力なテキストエディタが用意されています。 editors/emacs および editors/vim といった他のエディタは Ports Collection の一部として用意されています。
これらのエディタはやや学習が複雑ですが、より高い機能性を提供します。
しかし、あなたが多量のテキストを編集することを考えているなら、vim や Emacs といった強力なエディタを習得することは、 より多くの時間を節約することでしょう。

ファイルを編集したり、文字入力を必要とするようなアプリケーションの多くは、自動的にテキストエディタを起動します。 シェル の節で説明したように、デフォルトのエディタを変更するには `EDITOR` 環境変数に希望するエディタを設定してください。

3.11. デバイスとデバイスノード

デバイスとはシステム上のハードウェアに関するものに対してよく使われる用語で、ディスクやプリンタ、グラフィックカードやキーボードが含まれます。 FreeBSD が起動するとき、ブートメッセージの大部分は検出されたデバイスについてのものです。
ブートメッセージは `/var/run/dmesg.boot` に保存されています。

各デバイスはデバイス名と番号を持ちます。 例えば、`ada0` は最初の SATA CD-ROM ドライブで、`kbd0` はキーボードを表します。

FreeBSD におけるほとんどのデバイス、デバイスノードと呼ばれる `/dev` にあるスペシャルファイルを通してアクセスしなければなりません。

3.12. マニュアルページ

FreeBSD についてのもっとも包括的な文書は、マニュアルページの形式になっているものです。 FreeBSD システム上のほとんどすべてのプログラムには、基本的な操作方法と利用可能な引数を説明しているリファレンスマニュアルが添付されています。 これらのマニュアルは `man` を使って見ることができます。

```
% man コマンド名
```

ここで コマンド名 のところには、知りたいコマンドの名前を入れます。 たとえば ls(1)

について知りたい場合には、次のように入力します。

```
% man ls
```

マニュアルは、トピックごとにセクション番号で分類されています。 FreeBSD では、以下のセクションがあります。

1. ユーザコマンド
2. システムコールとエラー番号
3. C のライブラリ関数
4. デバイスドライバ
5. ファイル形式
6. ゲームや娯楽
7. さまざまな情報
8. システムの管理と操作のためのコマンド
9. システムカーネルインタフェース

時折、 同じトピックがオンラインマニュアルの複数のセクションに記載されている場合があります。 たとえば、`chmod` ユーザコマンドと `chmod()` システムコールの場合がそれに該当します。 man(1) にセクション番号を与えることで、 表示したいセクションを指定できます。

```
% man 1 chmod
```

上のようにすれば、ユーザコマンド chmod(1) のマニュアルページが表示されます。 オンラインマニュアルの特定セクションへの参照は、慣習的に書かれている文書で括弧の中に示されます。 すなわち、chmod(1) はユーザコマンドを、chmod(2) はシステムコールの方を示しています。

マニュアルページの名前を知らない場合には、`man -k` を使ってマニュアルページの解説 (description) からキーワードを検索してください。

```
% man -k mail
```

このコマンドは、"mail" というキーワードをコマンド解説に含むコマンドの一覧を表示します。 これは apropos(1) と同等の機能です。

`/usr/sbin` にあるすべてのコマンドの説明を読むには、以下のように実行してください。

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

または、以下を実行してください。

```
% cd /usr/sbin
```

```
% whatis * |more
```

3.12.1. GNU の Info ファイル

FreeBSD には Free Software Foundation (FSF) によるアプリケーションやユーティリティが含まれています。
これらのプログラムには、マニュアルページに加えて `info` ファイルと呼ばれるハイパーテキスト形式の文書が付属しています。 この文書は info(1)、あるいは editors/emacs をインストールしているなら emacs の info モードで読むことができます。

info(1) を使うには、次のように入力してください。

```
% info
```

`h` と入力すると、簡単な手引きを読むことができます。 クイックコマンドリファレンスは `?` を入力してください。

Chapter 4. アプリケーションのインストール - packages と ports

4.1. この章では

FreeBSD の基本システムには数多くのシステムツールが含まれています。 FreeBSD は、サードパーティ製のソフトウェアの導入を支援するために、
ソースコードをコンパイルしてインストールする Ports Collection と、
コンパイル済みのバイナリをインストールする packages という相補的な 2 つの技術を提供しています。
どちらのシステムを用いても、
ローカルメディアやネットワーク上からソフトウェアをインストールできます。

この章を読むと、以下のことがわかります。

- packages と ports の違い
- FreeBSD に移植されたサードパーティ製のソフトウェアの探し方
- pkg を用いてバイナリ package を管理する方法
- Ports Collection を用いてサードパーティ製のソフトウェアをソースコードから構築する方法
- インストール後の設定のために、 アプリケーションとともにインストールされたファイルを探す方法
- ソフトウェアのインストールに失敗した場合に、どうしたらよいか

4.2. ソフトウェアのインストール

UNIX® システムでは、
サードパーティ製ソフトウェアの典型的なインストール手順は以下のようになります。

1. ソースコード、 またはバイナリ形式で配布されているソフトウェアを探し出し、ダウンロードする。
2. 配布時のフォーマットからソフトウェアを取り出す。 一般的には compress(1), gzip(1), bzip2(1) または、xz(1) といったプログラムで圧縮された tarball です。
3. INSTALL または README ファイル、あるいは doc/ サブディレクトのファイルからドキュメントを探しだし、
ソフトウェアのインストール方法を調べる。
4. ソース形式でソフトウェアが配布されている場合はコンパイルを行う。 ここでは、Makefile の編集、 または、`configure` スクリプトの実行を伴うことがあります。
5. ソフトウェアの動作を確認し、インストールする。

FreeBSD *port* は、
アプリケーションをソースコードからコンパイルする際の処理を自動化するように設計されたファイルの集まりです。 port を構成するファイルは、
自動的にアプリケーションをダウンロードし、展開、パッチ作業、
コンパイル、そしてインストールを行うために必要な情報を含んでいます。

ソフトウェアが、すでに FreeBSD に移植され、 FreeBSD 上で試験されていなければ、適切にインストールが行われ、動作するように、 編集する必要があるかもしれません。

しかしながら、37000 を越えるサードパーティ製アプリケーションが FreeBSD に移植されています。可能な場合は、これらのアプリケーションをコンパイル済みの *packages* としてダウンロードできます。

package は、package 管理コマンドで扱うことができます。

packages と ports は依存関係を理解します。 package または port を用いてアプリケーションをインストールすると、
依存するライブラリがまだインストールされていない場合には、
最初にライブラリが自動的にインストールされます。

FreeBSD の package は、コンパイル済みのアプリケーションの全コマンド、各種設定ファイルやドキュメントを含んでいます。 pkg コマンドでは、`pkg install` といったコマンドで、 package を扱うことができます。

2 つの技術は類似していますが、 packages と ports にはそれぞれ独自の特徴があります。それぞれのアプリケーションのインストールに対する必要要件に応じてどちらかを選択してください。

package の利点

- 一般的に、あるアプリケーションの package の tarball は、 ソースコードを含む tarball より小さなサイズとなります。
- packages はコンパイルの時間を必要としません。 このことは、遅いシステム上で Mozilla, KDE, または GNOME といった大きなアプリケーションを扱う場合に重要となります。
- packages を用いれば、 ソフトウェアのコンパイルに関する知識は必要ありません。

port の利点

- packages は、通常最も多くのシステムで実行できるように、非常に保守的な設定で構築されています。 port からコンパイルすることで、コンパイルオプションを指定できます。
- アプリケーションのなかには、
どの機能をインストールするかをコンパイル時に設定するものがあります。 たとえば、Apache は多種多様な ビルトインオプションを設定できます。

  設定を区別するために、同じアプリケーションに対して 複数の packages が存在することがあります。 たとえば、Ghostscript は Xorg がインストールされているかどうかにより、 ghostscript package と ghostscript-nox11 package が選択可能となっています。 アプリケーションのコンパイルオプションが 1 つもしくは 2 つ以上になると、 複数の packages を用意することは困難になります。

- ライセンス条項で、 バイナリでの配布を禁止しているソフトウェアがあります。このようなソフトウェアはソースコードで配布される必要があり、
エンドユーザがコンパイルしなくてはなりません。
- バイナリ配布を信用していない人や、
潜在的な問題点を見つけ出すためにソースコードを読むことを好む人がいます。
- カスタマイズしたパッチを適用するためには、 ソースコードが必要になります。

ports の更新状況を把握するために、FreeBSD ports メーリングリスト や FreeBSD ports bugs メーリングリスト を購読するとよいでしょう。

アプリケーションをインストールする前に、
そのアプリケーションに関連したセキュリティ上の問題がないことを
https://vuxml.freebsd.org/ で確認するか、`pkg audit -F` と入力して、
インストールされているアプリケーションに既知の脆弱性がないことを確認してください。

この章では、packages と ports を用いた FreeBSD 上でのサードパーティ製ソフトウェアのインストール方法や管理方法について説明します。

4.3. ソフトウェアの探し方

FreeBSD 上で利用可能なアプリケーションのリストは常に増えています。
インストールするソフトウェアを探す方法はたくさん用意されています。

- FreeBSD ウェブサイトは、利用可能なすべてのアプリケーションの最新の一覧を、検索できる形で https://www.FreeBSD.org/ja/ports/ において公開しています。ports はアプリケーションの名前や、ソフトウェアのカテゴリで検索出来ます。
- Dan Langille は、包括的な検索ユーティリティや Ports Collection にあるアプリケーションの変更点を追跡する FreshPorts.org を公開しています。登録したユーザは、監視している ports がアップデートされた時に、そのことを自動的にメールで知らせてくれるような、カスタマイズ可能な監視リストを使うことができます。
- アプリケーションを見つけることが難しい場合には、SourceForge.net または GitHub.com のようなサイトで探してみてください。その後、そのアプリケーションが ports で利用可能かどうかを FreeBSD サイト で調べて下さい。
- バイナリ package リポジトリでアプリケーションを探すには、以下のように実行してください。

```
# pkg search subversion
git-subversion-1.9.2
java-subversion-1.8.8_2
p5-subversion-1.8.8_2
py27-hgsubversion-1.6
py27-subversion-1.8.8_2
ruby-subversion-1.8.8_2
subversion-1.8.8_2
subversion-book-4515
subversion-static-1.8.8_2
subversion16-1.6.23_4
subversion17-1.7.16_2
```

package 名にはバージョン番号が含まれます。また、python ベースの ports では、共に構築された python のバージョン番号も含まれます。ports によっては、複数のバージョンを利用できるものがあります。subversion では、複数のバージョンを利用できますが、

異なるコンパイルオプションで構築されたものも利用できます。 インストールする package を指定する際には、 アプリケーションに、port ツリーのパスである、 port のオリジンを指定すると良いでしょう。 `pkg search` に `-o` オプションを付けて、実行してください。各 package のオリジンの一覧が表示されます。

```
# pkg search -o subversion
devel/git-subversion
java/java-subversion
devel/p5-subversion
devel/py-hgsubversion
devel/py-subversion
devel/ruby-subversion
devel/subversion16
devel/subversion17
devel/subversion
devel/subversion-book
devel/subversion-static
```

`pkg search` は、 リポジトリデータベースの説明やその他のフィールドにおいて、シェルグロブ、正規表現、完全一致にも対応しています。 詳細については、ports-mgmt/pkg または ports-mgmt/pkg-devel のインストール後、 pkg-search(8) をご覧ください。

- Ports Collection がすでにインストールされていれば、 ports ツリーのローカルバージョンを調べることができます。 port がどのカテゴリに分類されているのかを知りたければ、 whereis(1) コマンドで調べることができます。 `whereis ファイル` と入力してください。ファイルの部分にはインストールを考えているプログラム名を入れます。

```
# whereis lsof
lsof: /usr/ports/sysutils/lsof
```

さらに、以下の例のように echo(1) を使って調べることもできます。

```
# echo /usr/ports/*/*lsof*
/usr/ports/sysutils/lsof
```

この方法では /usr/ports/distfiles 以下にダウンロードされたファイル名にもマッチします。

- また、Ports Collection に備わっている検索機能を利用して port を検索する方法もあります。この検索機能を利用するには、 cd コマンドを用いて /usr/ports ディレクトリに移動し、`make search name=プログラム名` と入力してください。 プログラム名の部分には検索したいソフトウェアの名前を入れてください。 たとえば、`lsof` を探すには次のようにします。

```
# cd /usr/ports
# make search name=lsof
```

```
Port:   lsof-4.88.d,8
Path:   /usr/ports/sysutils/lsof
Info:   Lists information about open files (similar to fstat(1))
Maint:  ler@lerctr.org
Index:  sysutils
B-deps:
R-deps:
```

Ports Collection に用意されている検索のメカニズムでは、インデックスファイルを利用して検索を行います。もし INDEX が必要であるというメッセージが表示されたら、`make fetchindex` を実行して、最新のインデックスファイルをダウンロードしてください。INDEX が用意されれば、`make search` で検索を実行できるでしょう。

"Path:" という行は、port がどこにあるかを示しています。

より絞られた情報を得るには、`quicksearch` と呼ばれる機能を使ってください。

```
# cd /usr/ports
# make quicksearch name=lsof
Port:   lsof-4.88.d,8
Path:   /usr/ports/sysutils/lsof
Info:   Lists information about open files (similar to fstat(1))
```

もっと詳しく検索するには、`make search key=string` または `make quicksearch key=string` と入力してください。*string* の部分には検索したいテキストを入れます。プログラムの名前がわからない場合でも、ある目的に関連した ports の検索に利用できるよう、テキストの部分には、コメント、説明文および依存情報を入れることができます。

`search` および `quicksearch` を使う場合には、検索文字列中の大文字と小文字を区別せずに検索が行われるので、"LSOF" を検索した結果は、"lsof" と同じ検索結果になります。

4.4. pkg によるバイナリ package の管理

pkg は、FreeBSD における伝統的な package 管理ツールの置き換えとなる次世代の管理ツールで、バイナリ packages をより早く、より簡単に管理できるようにする数多くの機能を提供します。

FreeBSD のミラーサイトが提供する事前に構築されたバイナリ package のみを使いたいと考えているサイトでは、pkg を使って package を管理するとよいでしょう。

しかしながら、ソースまたは自分自身で用意したリポジトリから構築したサイトでは、port 管理ツールが別に必要となります。

pkg はバイナリ package のみを扱うので、そのような管理ツールの置き換えとはなりません。これらのツールは、ソフトウェアをバイナリ packages と Ports Collection の両形式からインストールできますが、pkg はバイナリ packages のみをインストールします。

4.4.1. pkg 入門

FreeBSD には、pkg およびマニュアルページをインストールするブートストラップユーティリティが用意されています。
このユーティリティは、FreeBSD 10.*X* 以降で動作するように設計されています。

このブートストラッププロセスは、すべての FreeBSD バージョンおよびアーキテクチャに対応しているわけではありません。
現在対応している一覧は、 https://pkg.freebsd.org/ で確認することができます。
対応していない場合には、 Ports Collection またはバイナリ package から pkg をインストールする必要があります。

システムをブートストラップするには、 以下を実行してください。

```
# /usr/sbin/pkg
```

ブートストラッププロセスに成功するには、 インターネットへの接続が必要です。

port をインストールするには以下を実行してください。

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

古い pkg_* ツールを用いたシステムをアップグレードする際には、新しいツールがすでにインストールされている package を認識するよう、データベースを新しいフォーマットへと変換する必要があります。 pkg をインストールしたら、以下のコマンドを実行して、package
データベースをこれまでの伝統的なフォーマットから新しいフォーマットへと変換してください。

```
# pkg2ng
```

このステップは、
サードパーティ製ソフトウェアがまだインストールされていないような、
新しくインストールされた直後のシステムでは必要ありません。

このステップは非可逆です。 一度 package データベースを pkg フォーマットへと変換したら、伝統的な pkg_* ツールを使うべきではありません。

package データベースを変換する際には、
新しいバージョンへのデータ変換に伴ったエラーが出力されることがあります。
通常、これらのエラーは無視して構いませんが、 pkg2ng 終了後、
変換に失敗したソフトウェアの一覧が表示されます。
これらのソフトウェアを手動で再インストールする必要があります。

FreeBSD のバージョンが 10.*X* より前であれば、 以下の行を /etc/make.conf に追加して、 Ports

Collection がソフトウェアの登録に、伝統的な package のデータベースではなく、pkg を用いるように設定してください。

```
WITH_PKGNG= yes
```

デフォルトでは、pkg は FreeBSD の package ミラー (リポジトリ) のバイナリ package を用います。カスタム package リポジトリの構築については、 Poudriere を用いた package の構築 をご覧ください。

その他の pkg の設定オプションは、pkg.conf(5) に記述されています。

pkg の利用情報は、 pkg(8) マニュアルページや、 `pkg` を引数なしに実行すると表示されます。

各 pkg コマンドの引数は、 コマンドに固有なマニュアルページに記述されています。 たとえば、`pkg install` のマニュアルページを読むには、 以下のコマンドのどちらかを実行してください。

```
# pkg help install
```

```
# man pkg-install
```

以下の節では、pkg を用いた通常のバイナリ package の管理について説明します。各コマンドでは、カスタマイズのために、 多くのオプションが使われています。詳細や、他の例については、 コマンドのヘルプやマニュアルページを参照してください。

4.4.2. Quarterly および Latest Ports ブランチ

`Quarterly` ブランチを使うと、 ユーザは、port および package のインストールおよびアップグレードを、 より予測可能で安定して行うことができます。基本的には、このブランチでは機能のアップデートは行われません。 Quarterly ブランチの目的は、セキュリティに関連する修正 (バージョンアップデートやコミットのバックポートなど)、 バグの修正および ports のコンプライアンスおよびフレームワークの変更の入手です。 Quarterly ブランチは、毎年の毎四半期 (1 月、4 月、7 月および 10 月) のはじめに HEAD から作成されます。 ブランチには、作成された年 (YYYY) をよび四半期 (Q1-4) により名前がつけられます。 たとえば、2016 年 1 月に作成された quarterly ブランチの名前は 2016Q1 となります。 `Latest` ブランチは、 最新バージョンの package をユーザに提供します。

quarterly から latest ブランチに移行するには、 以下のコマンドを実行してください。

```
# cp /etc/pkg/FreeBSD.conf /usr/local/etc/pkg/repos/FreeBSD.conf
```

/usr/local/etc/pkg/repos/FreeBSD.conf ファイルを編集して、`url:` 行の *quarterly* 文字列を *latest* に変更してください。

編集後は、以下のようになります。

```
FreeBSD: {
```

```
  url: "pkg+http://pkg.FreeBSD.org/${ABI}/latest",
  mirror_type: "srv",
  signature_type: "fingerprints",
  fingerprints: "/usr/shared/keys/pkg",
  enabled: yes
}
```

最後に、以下のコマンドを実行して (latest) リポジトリのメタデータからアップデートしてください。

```
# pkg update -f
```

4.4.3. インストールされている package の情報を入手する

オプションを使用しないで `pkg info` を実行すると、 システムにインストールされているすべての package もしくは、 ある特定の package の情報が得られます。

たとえば、インストールされている pkg の情報を調べるには、 以下のように実行してください。

```
# pkg info pkg
pkg-1.1.4_1
```

4.4.4. package のインストールと削除

バイナリ package をインストールするには、 以下のコマンドを使ってください。 ここで *packagename* は、インストールする package の名前です。

```
# pkg install packagename
```

このコマンドは、リポジトリデータを使用して、
インストールすべきソフトウェアのバージョン、および、
インストールされていない依存ソフトウェアがあるかどうかを調べます。 たとえば、curl をインストールするには以下を実行してください。

```
# pkg install curl
Updating repository catalogue
/usr/local/tmp/All/curl-7.31.0_1.txz          100% of 1181 kB 1380 kBps 00m01s

/usr/local/tmp/All/ca_root_nss-3.15.1_1.txz   100% of  288 kB 1700 kBps 00m00s

Updating repository catalogue
The following 2 packages will be installed:

        Installing ca_root_nss: 3.15.1_1
        Installing curl: 7.31.0_1

The installation will require 3 MB more space
```

```
0 B to be downloaded

Proceed with installing packages [y/N]: y
Checking integrity... done
[1/2] Installing ca_root_nss-3.15.1_1... done
[2/2] Installing curl-7.31.0_1... done
Cleaning up cache files...Done
```

新しい package と依存関係から追加された package は、インストール済み package 一覧に表示されます。

```
# pkg info
ca_root_nss-3.15.1_1    The root certificate bundle from the Mozilla Project
curl-7.31.0_1   Non-interactive tool to get files from FTP, GOPHER, HTTP(S) servers
pkg-1.1.4_6 New generation package manager
```

必要のなくなった packages は、pkg delete を使って削除できます。
たとえば、以下のようにして削除できます。

```
# pkg delete curl
The following packages will be deleted:

    curl-7.31.0_1

The deletion will free 3 MB

Proceed with deleting packages [y/N]: y
[1/1] Deleting curl-7.31.0_1... done
```

4.4.5. インストールされている package のアップグレード

以下のコマンドを実行すると、インストールされている packages が最新のバージョンにアップグレードされます。

```
# pkg upgrade
```

このコマンドは、インストールされているソフトウェアのバージョンと、
リポジトリのカタログから利用できるバージョンとを比較し、リポジトリからアップグレードします。

4.4.6. インストールされている package の検証

サードウェア製アプリケーションに対する脆弱性は、定期的に見つかります。脆弱性を調べるために、pkg は、検証機能を持っています。
システムにインストールされているソフトウェアに既知の脆弱性がないかどうかを調べるには、
以下のように実行してください。

```
# pkg audit -F
```

4.4.7. 使われていない **package** の自動削除

package を削除すると、不必要な依存 package が残されることがあります。依存のためにインストールされたが、現在は不必要になった package (リーフ package) は、以下のコマンドで自動的に検出され、削除されます。

```
# pkg autoremove
Packages to be autoremoved:
    ca_root_nss-3.15.1_1

The autoremoval will free 723 kB

Proceed with autoremoval of packages [y/N]: y
Deinstalling ca_root_nss-3.15.1_1... done
```

依存によりインストールされた packages は、 *automatic* package と呼ばれます。 非 automatic packages、 すなわち他の package からの依存ではなく、 明示的にインストールした package の一覧は以下のようにして出力できます。

```
# pkg prime-list
nginx
openvpn
sudo
```

`pkg prime-list` は、 /usr/local/etc/pkg.conf で設定されているエイリアスコマンドです。他にもシステムの package データベースの問い合わせに用いることができる多くのコマンドが用意されています。 たとえば、`pkg prime-origins` コマンドを使うと、 上記で得られた port 一覧のオリジナルの port ディレクトリを知ることができます。

```
# pkg prime-origins
www/nginx
security/openvpn
security/sudo
```

この一覧と ports-mgmt/poudriere または ports-mgmt/synth といったツールを使うと、システムにインストールされているすべての package を再構築できます。

インストールされた package に automatic のマーク付けをするには、 以下のように実行してください。

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

リーフ package や automatic としてマークされた package は、 `pkg autoremove` で選択されます。

インストールされた package を 非 automatic とマークするには、以下のように実行してください。

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

4.4.8. package データベースのリストア

伝統的な package 管理システムとは異なり、pkg には package データベースをバックアップするメカニズムがあります。
この機能はデフォルトで有効に設定されています。

スクリプトによる定期的な package データベースのバックアップを無効にするには、periodic.conf(5) の中で、`daily_backup_pkgdb_enable="NO"` と設定してください。

過去にバックアップした package データベースの中身をリストアするには、以下のコマンドを実行してください。以下のコマンドの */path/to/pkg.sql* については、バックアップのある場所に置き換えて実行してください。

```
# pkg backup -r /path/to/pkg.sql
```

システムの定期的なスクリプトによって取得されたバックアップをリストアする場合には、 リストアの前に展開しておく必要があります。

手動で pkg データベースをバックアップするには、以下のコマンドを実行してください。以下のコマンドの */path/to/pkg.sql* については、適切なファイル名と場所に置き換えて下さい。

```
# pkg backup -d /path/to/pkg.sql
```

4.4.9. 古くなった package の削除

デフォルトでは、pkg は、pkg.conf(5) の `PKG_CACHEDIR` 変数で定義されるキャッシュディレクトリにバイナリ packages を保存します。 インストールされている package の最新のコピーのみが保存されます。 古いバージョンの pkg では、過去にインストールされたすべての package が保存されていました。 これらの古くなったバイナリ package を削除するには、 以下を実行してください。

```
# pkg clean
```

キャッシュ全体を削除するには以下を実行してください。

```
# pkg clean -a
```

4.4.10. package メタデータの変更

FreeBSD Ports Collection では、メジャーバージョン番号が変更になることがあります。これに対応するために、pkg には、 package の情報をアップデートするコマンドが組み込まれています。たとえば、lang/php5 が、 バージョン 5.4 を表すようになり、 lang/php5 を lang/php53 と名前を変更する必要があるような場合に、有用です。

上記の例の package の情報を変更するには、 以下のように実行してください。

```
# pkg set -o lang/php5:lang/php53
```

別の例として、lang/ruby18 を lang/ruby19 にアップデートするには、 以下のようにしてください。

```
# pkg set -o lang/ruby18:lang/ruby19
```

最後の例として、 libglut 共有ライブラリの情報を graphics/libglut から graphics/freeglut へと変更するには、 以下のように実行してください。

```
# pkg set -o graphics/libglut:graphics/freeglut
```

package の情報を変更したら、 情報が変更された package に依存している packages を再インストールすることが重要となります。 依存 packages を再インストールするには、 以下のように実行してください。

```
# pkg install -Rf graphics/freeglut
```

4.5. Ports Collection の利用

Ports Collection は、Makefile, 修正パッチ、 説明文などの一連のファイルのことです。これらのファイルの各セットは、 個々のアプリケーションをコンパイルして FreeBSD にインストールするために用いられ、*port* と呼ばれています。

デフォルトでは、Ports Collection は、/usr/ports 以下のサブディレクトリに置かれます。

Ports Collection をインストールして使用する前に、一般的には、ソフトウェアのインストールに、pkg でダウンロードしたバイナリパッケージと Ports Collection を組み合わせて使うことはあまり良いことではないことを知っておいてください。 pkg は、デフォルトでは ports ツリーの (HEAD ではなく) quarterly ブランチリリースを追いかけます。 HEAD の port と対応する quarterly ブランチの port の依存関係は異なる可能性があり、そのため pkg でインストールされた依存関係と Ports Collection の依存関係の間で競合が起きることがあります。 もし、Ports Collection と pkg を組み合わせて使用しなければならないのであれば、Ports Collection と pkg が同じ

> ports ツリーのブランチを使用していることを必ず確認してください。

Ports Collection は、ソフトウェアのカテゴリを表すディレクトリを持ちます。各カテゴリには、各アプリケーションのサブディレクトリがあります。各アプリケーションのサブディレクトリには、プログラムを FreeBSD 上で正しくコンパイルしてインストールする方法を提供する、*ports* スケルトンと呼ばれるファイルのセットが含まれています。それぞれの port スケルトンには、次のファイルおよびディレクトリが含まれています。

- Makefile:
  このファイルにはアプリケーションのコンパイル方法やシステムのどこにインストールするかを指定する命令文が含まれています。
- distinfo: このファイルには、その port を構築するためにダウンロードする必要があるファイルのファイル名と、チェックサム情報が含まれています。
- files: このディレクトリには FreeBSD 上でプログラムをコンパイルし、インストールするための修正パッチが含まれています。このディレクトリには、その port の構築に必要なその他のファイルが入る場合もあります。
- pkg-descr: このファイルにはプログラムに関する、より詳しい説明文が含まれます。
- pkg-plist: これは、その port によってインストールされる全ファイルのリストです。これにはプログラムを削除する際に、どのファイルを削除すれば良いのかを ports システムに伝える役割もあります。

これらの他に pkg-message や特殊な状況に対応するためのファイルを含む ports もあります。これらのファイルについての詳細および ports の一般的な説明については、port 作成者のためのハンドブック をご覧下さい。

port は実際のソースコード (distfile とも呼ばれます) を含んではいません。port の構築の展開部で、ダウンロードされたソースは自動的に /usr/ports/distfiles に保存されます。

4.5.1. Ports Collection のインストール

port を用いてアプリケーションをコンパイルできるようにするには、まず最初に Ports Collection をインストールする必要があります。FreeBSD のインストール時に Ports Collection をインストールしていない場合には、以下の方法のどれかを用いてインストールしてください。

Procedure: Git を用いる方法

ports ツリーの管理が必要な場合や、ローカルで変更点をメンテナンスする必要がある場合、および FreeBSD-CURRENT を使用している場合には、Git を使って Ports Collection を取得する方法があります。Git のより詳細な説明については、Git Primer を参照してください。

1. Git を使って ports ツリーをチェックアウトする前に、Git をインストールしておく必要があります。ports ツリーがすでにインストールされていれば、以下のようにして Git をインストールできます。

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

```
# make install clean
```

ports ツリーが利用できない場合や、 package の管理に pkg を使っているのであれば、package から Git をインストールできます。

```
# pkg install git
```

2. HEAD ブランチの ports ツリーをチェックアウトしてください。

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

3. または、quarterly ブランチをチェックアウトしてください。

```
# git clone https://git.FreeBSD.org/ports.git -b 2020Q3 /usr/ports
```

4. Git で最初のチェックアウトをした後は、必要に応じて /usr/ports をアップデートしてください。

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

5. 必要に応じて /usr/ports を別の quarterly ブランチにスイッチしてください。

```
# git -C /usr/ports switch 2020Q4
```

4.5.2. ports のインストール

この節では、Ports Collection を利用してプログラムをインストールしたり、システムから削除したりする基本的な手順について説明します。 利用可能な `make` のターゲットや環境変数についての詳細は ports(7) をご覧ください。

いかなる port でも、構築する前には、 前節に書かれているように、Ports Collection をアップデートしてください。
サードパーティ製のソフトウェアをインストールすると、
セキュリティの脆弱性を引き起こす可能性があります。 その port に関連したセキュリティ上の問題がないことを、まずは https://vuxml.freebsd.org/ で確認してください。または、 新しい port をインストールする前に、 `pkg audit -F` を実行してください。 毎日のシステムのセキュリティ確認時に、自動的にセキュリティの検査およびデータベースの更新を行うようにこのコマンドを設定できます。 詳しくは、pkg-audit(8) および periodic(8) を参照してください。

Ports Collection は、ネットワークに接続できることを想定しています。 また、superuser の権限も必要となります。

port をコンパイルしてインストールするには、 インストールしたい port のディレクトリに移動してください。 その後、プロンプトから `make install` と入力してください。すると、次のような出力が現われるはずです。

```
# 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

===>  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
#
```

`lsof` は高い権限で動作するプログラムなので、インストールする時にセキュリティに関する警告が表示されます。
インストールが終わったら、プロンプトが戻ります。

シェルによってはコマンドの実行ファイルを探す時間を短縮するために、 環境変数 `PATH` に登録されている ディレクトリのコマンド一覧をキャッシュするものがあります。 `tcsh` シェルを使っているのであれば、
フルパスを指定することなく新しくインストールしたコマンドを利用できるように、 `rehash` を実行してください。 `sh` シェルを使っているのであれば かわりに `hash -r` を実行してください。詳細については、 あなたの使っているシェルのドキュメントをご覧ください。

インストールの間に、作業用ディレクトリが作成されます。

このディレクトリにはコンパイル時に使用されるすべての一時ファイルが含まれています。
このディレクトリを削除することで、ディスク容量を節約でき、また port を新しいバージョンへアップデートする際に問題が起こる可能性を小さくします。

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

port を構築する際に、`make install clean` と実行することで、これらの余分な手順を省くことができます。

ports のインストールのカスタマイズ

ports の中にはビルドオプションを指定できるものがあります。 このオプションを指定することで、アプリケーションの機能の一部を有効もしくは無効にできます。
また、セキュリティオプションを設定したり、 その他のカスタマイズを行うことができます。
このようなアプリケーションには www/firefox, security/gpgme や mail/sylpheed-claws などがあります。 port が他のカスタマイズ可能なオプションを持つ ports に依存する場合には、デフォルトでは、ユーザに port のオプションをメニューから選択させる設定のため、
何度もユーザとの対話が起こり待たされることがあります。 これを避けるには、まず最初に port スケルトンで `make config-recursive` を実行して設定を一括で行い、その後 `make install [clean]` を実行して port を構築してインストールしてください。

`config-recursive` を実行する際、`all-depends-list` を実行すると、設定すべき ports の一覧を得ることができます。 多くの場合は、すべての依存 ports のオプションが定義され、 ports オプションの画面が表示されなくなり、すべてのオプションが設定されたことを確認できるまで `make config-recursive` を実行すると良いでしょう。

port の構築後、 再びこのメニューを表示させてオプションの追加や削除、設定の変更を行う方法はたくさんあります。 一つ目は port のディレクトリに `cd` で移動し、`make config` と入力する方法です。 別の方法は `make showconfig` を使う方法です。 他の方法は `make rmconfig` の実行です。 このコマンドを実行すると選択されているすべてのオプションが削除され、設定をもう一度やり直すことができます。 これらの方法や他の方法についての詳細は、 ports(7) マニュアルで説明されています。

ports は、いくつかの環境変数を参照する fetch(1) を用いてソースファイルをダウンロードします。 FreeBSD システムがファイアウォールの内側であったり、 FTP/HTTP プロキシを使う場合には、`FTP_PASSIVE_MODE`, `FTP_PROXY`, `FTP_PASSWORD` の環境変数を設定することなります。 対応している環境変数の一覧については fetch(3) をご覧ください。

インターネットに常時接続できないユーザのために `make fetch` コマンドが用意されています。 このコマンドを /usr/ports で実行してすべての distfiles をダウンロードするか、 /usr/ports/net といったカテゴリや、あるスケルトンにおいても実行できます。 ある port がライブラリやその他の ports に依存している場合には、 別のカテゴリの ports の distfiles はダウンロードされないことに注意してください。 port が依存しているすべての distfiles をダウンロードしたければ、`make fetch-recursive` を使ってください。

めったにないことかもしれませんが、ローカルに distfiles のリポジトリがあるような場合に、MASTER_SITES 変数を変更することで Makefile で指定されているダウンロードの場所を変更することができます。設定する場合には、変更先を以下のようにして指定してください。

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

WRKDIRPREFIX 変数と PREFIX 変数を変更することで、作業ディレクトリやターゲットディレクトリをデフォルトのものから変更できます。

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

とすると、ports は /usr/home/example/ports でコンパイルされ、すべて /usr/local 以下にインストールされます。

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

この場合、port のコンパイルは /usr/ports でおこない、/usr/home/example/local にインストールします。そして

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

とすれば両者を組み合わせることが可能です。

これらを環境変数に設定する方法もあります。どのように環境変数を設定するかについては、あなたの使っているシェルのマニュアルページを参照してください。

4.5.3. インストールした ports の削除

インストールされた ports は、`pkg delete` コマンドで削除できます。このコマンドの使用例は、pkg-delete(8) マニュアルページにあります。

あるいは、port のディレクトリにて `make deinstall` を実行することでも削除できます。

```
# 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
```

port が削除されるときに表示されるメッセージを読むことをお勧めします。 もし削除した port に依存するアプリケーションがあった場合には、 その情報が表示されますが、port の削除は行われます。
そのようなケースでは、依存を直すためにアプリケーションを再インストールするとよいでしょう。

4.5.4. ports のアップグレード

ports のインストール後、時間が経過すると、Ports Collection で新しいバージョンのソフトウェアを利用できるようになります。 この章では、どのようにしてアップグレードする必要のあるソフトウェアを判断するか、そしてアップグレードの方法について説明します。

インストールされている ports の新しいバージョンを利用できるかどうかを知るには、まず、 最新の ports ツリーがインストールされていることを確認してください。 これには、「Git を用いる方法」で書かれているアップデートのコマンドを使ってください。 FreeBSD 10 以降のシステム、または、pkg に変換されたシステムでは、 以下のコマンドを実行すると、現在利用可能なバージョンよりも古い ports の一覧が表示されます。

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

FreeBSD 9.*X* より前のシステムでは、 現在利用可能なバージョンよりも古い ports の一覧を表示されるには、以下のコマンドを実行してください。

```
# pkg_version -l "<"
```

アップグレードする前に /usr/ports/UPDATING を、ファイルの頭から、ports を最後にアップデートした日、
もしくはシステムをインストールをした日に最も近い日まで目を通してください
このファイルには port をアップグレードする際にユーザが遭遇するであろう問題や、追加で必要な作業などが記述されています。
例えば、ファイル形式の変更や設定ファイルの場所の変更、
前のバージョンと互換性がなくなったことなどが書かれています。
アップグレードする必要のある ports に関連した手順に注意し、アップグレードする際にはこれらの手順に従ってください。

ports のアップグレードおよび管理ツール

Ports Collection には、実際にアップグレードを行うためのユーティリティがいくつか用意されています。
それぞれのユーティリティは長所と短所を持っています。

歴史的に、最もインストールされ使われているのは、 Portmaster または Portupgrade です。 Synth は新しいユーティリティです。

特定のシステムにおいて、 どのツールを選択するとベストかについては、システム管理者によります。 これらのどのツールでも、使う前には、データのバックアップをとることが推奨されます。

portmaster を用いた ports のアップグレード

ports-mgmt/portmaster は、 インストールされている ports のアップグレードをおこなう、とても小さなユーティリティです。 FreeBSD のベースシステムとしてインストールされているツールだけを使い、 他の ports やデータベースに依存しないように設計されています。 port からこのユーティリティをインストールするには以下のようにしてください。

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

Portmaster は、 ports を 4 つのカテゴリに分類します。

- Root ports: 他の port に依存せず、 他の port からも依存されない ports。
- Trunk ports: 他の port には依存しないが、 他の port から依存されている ports。
- Branch ports: 他の port に依存し、 他の port からも依存されている ports。
- Leaf ports: 他の port に依存するが、 他の port からは依存されない ports。

これらのカテゴリの一覧や、アップデート可能な port の一覧を表示するには以下のようにしてください。

```
# 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
```

以下のコマンドを使うと、 古くなった ports をすべてアップデートします。

```
# portmaster -a
```

Portmaster のデフォルトの設定では、 インストールされている port を削除する前に、バックアップ用の package が作成されます。 このバックアップは、新しいバージョンのインストールに成功すると削除されます。 `-b` を使うと、Portmaster の自動的なバックアップの削除は行いません。 `-i` を追加すると、Portmaster をインタラクティブモードで使用できます。 このモードでは、各 port をアップグレードするかどうかの選択を対話的に行うことができます。
多くのオプションが利用可能です。 portmaster(8) マニュアルページから、それらの使用方法に関する詳細な説明を読んでください。

アップグレードの過程でエラーに遭遇した場合には、 `-f` を使ってすべての ports のアップグレードや再構築を行なってください。

```
# portmaster -af
```

Portmaster を使ってシステムに新しい ports をインストールしたり、 新しい port のコンパイルやインストール前に依存するすべての port をアップグレードできます。この機能を使う時には、 Ports Collection の場所を指定してください。

```
# portmaster shells/bash
```

ports-mgmt/portmaster に関するより多くの情報は、pkg-descr にあります。

Portupgrade を用いた ports のアップグレード

ports-mgmt/portupgrade は、 インストールした ports のアップグレードを行なうためのもう一つのユーティリティです。 このユーティリティは ports を管理するために用いられるアプリケーションのセットをインストールします。 Ruby に依存します。port をインストールするには、以下を実行してください。

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

このユーティリティを使ってアップグレードを行う前に、 `pkgdb -F` を使って、 インストールされている ports の一覧を調べてください。 矛盾が検出された場合には修復してください。

システムにインストールされている port の中で古くなったものをすべてアップデートするには `portupgrade -a` を実行してください。 もし、すべての ports に対して個別にアップグレードするかどうかを確認したいのであれば、 `-i` を追加してください。

```
# portupgrade -ai
```

ports で利用可能なすべてのアプリケーションではなく、ある特定のアプリケーションだけを更新したいのであれば、 `portupgrade pkgname` を実行してください。アップグレードするアプリケーションが依存しているすべての ports をまず先に更新したい場合には、 `-R` を使ってください。

```
# portupgrade -R firefox
```

-P オプションを使うと、 portupgrade は PKG_PATH に登録されているローカルディレクトリから、利用可能な package を探します。 ローカルに利用可能な packages が見つからなければ、リモートサイトから package のダウンロードを試みます。 packages をローカルに見つけることができず、 リモートサイトからもダウンロードできない場合には、portupgrade は ports からインストールを行ないます。 ports を使用したくなければ、-PP オプションを指定してください。 この最後のオプションを設定すると、 もし package が利用できなければ Portupgrade は終了します。

```
# portupgrade -PP gnome3
```

また、ビルドやインストールを行なわず、 distfiles または packages だけをダウンロードしたければ、 -F オプションを指定してください。 利用可能なすべてのオプションについては、 portupgrade(1) のマニュアルを参照してください。

ports-mgmt/portupgrade に関するより多くの情報は、pkg-descr にあります。

4.5.5. ports とディスク容量

Ports Collection を使い続けていると、 そのうちディスクを食いつぶしてしまうでしょう。 ports をビルドしてインストールした後、 ports スケルトンで make clean を実行すると、作業用の work ディレクトリを削除します。 Portmaster を使って port をインストールする場合には、-K を使わなければこのディレクトリは自動的に削除されます。 Portupgrade がインストールされている場合には、 以下のコマンドはローカルの Ports Collection に見つかったすべての work ディレクトリを削除します。

```
# portsclean -C
```

さらに、時間が経つにつれ /usr/ports/distfiles には、古くなったソースファイルがたまっていきます。 Portupgrade を使って、どの ports からも使われていないすべての distfiles を削除するには次のように実行してください。

```
# portsclean -D
```

Portupgrade を使って、システムにインストールされている port から使われていない distfiles をすべて削除することができます。

```
# portsclean -DD
```

もし Portmaster がインストールされているのであれば、以下を実行してください。

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

デフォルトでは、このコマンドはインタラクティブに設定されているため、ユーザに対して distfile を削除すべきかどうかを確認するプロンプトが表示されます。

これらのコマンドに加え、ports-mgmt/pkg_cutleaves は、必要なくなった ports を削除する作業を自動化します。

4.6. Poudriere を用いた package の構築

poudriere は、FreeBSD package を作成したり、試験に用いられる BSD ライセンスのユーティリティです。このユーティリティは、FreeBSD jails を用いて、独立したコンパイル環境を構築します。これらの jail を使って、インストールされている FreeBSD のバージョンとは異なるバージョンの package を作成したり、ホストが amd64 のシステムでは、i386 用の package を構築することもできます。構築された package のレイアウトは公式のミラーと同じです。これらの package は、pkg(8) や他の package 管理ツールで利用できます。

ports-mgmt/poudriere package または port から poudriere をインストールしてください。アプリケーションをインストールすると、サンプルの設定ファイルである /usr/local/etc/poudriere.conf.sample もインストールされます。このファイルを /usr/local/etc/poudriere.conf にコピーして、ローカルの環境に合わせて編集してください。

poudriere を実行するシステムで、必ずしも ZFS を使う必要はありませんが、有用です。ZFS を使う際には、/usr/local/etc/poudriere.conf の中で ZPOOL を指定する必要があります。そして、FREEBSD_HOST を最も近いミラーに設定してください。CCACHE_DIR を定義することで、devel/ccache を使ったコンパイルのキャッシュが可能となり、コンパイルで頻繁に使われるコードの構築時間を短縮できます。poudriere データセットを /poudriere にマウントされた独立したツリーに置くと良いでしょう。他の値はデフォルトの値で十分です。

同時に走らせるコンパイル数の定義には、認識されたコアプロセッサの数が用いられます。RAM もしくはスワップ空間のどちらかの仮想メモリを十分用意してください。
もし、仮想メモリを使い切ってしまったら、jail の構築は中断し、異常なメッセージが表示されることでしょう。

4.6.1. Jails および Port ツリーの初期化

設定が終わったら、poudriere を初期化して、必要とする FreeBSD ツリーおよび jail、そして ports ツリーをインストールしてください。jail の名前を -j、FreeBSD のバージョンを -v で指定してください。FreeBSD/amd64 システムでは、-a を使ってアーキテクチャに i386 または amd64 を設定できます。デフォルトでは、uname で表示されるアーキテクチャに設定されます。

```
# 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
```

一つのコンピュータ上で、 複数の設定、複数の jails、異なる port ツリーから poudriere は port をビルドできます。 これらのコンビネーションのカスタム設定は セット と呼ばれます。詳細については、ports-mgmt/poudriere もしくは ports-mgmt/poudriere-devel をインストール後、poudriere(8) の CUSTOMIZATION の章をご覧下さい。

ここで示される基本設定では、jail, ports そしてセット固有の make.conf を /usr/local/etc/poudriere.d に置いてください。 この例でのファイル名 13amd64-local-workstation-make.conf は、jail 名、port 名そして、セット名の組み合わせで付けられています。 システムの make.conf

と、この新しいファイルは、ビルド時に結合され、構築した jail で用いられる make.conf を作成します。

ビルドする package を 13amd64-local-workstation-pkglist に記載してください。

```
editors/emacs
devel/git
ports-mgmt/pkg
...
```

特定の ports に対し、 オプションや依存を設定してください。

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

最後に packages を構築し、 package リポジトリを生成してください。

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

このコマンドの実行中に `Ctrl` + `t` を押すと、現在のビルド状況が表示されます。 Poudriere は /poudriere/logs/bulk/jailname にあるファイルも構築します。
このファイルをウェブサーバと共に使うことで、 ビルド情報を表示できます。

これが終わると、poudriere リポジトリを package のインストールに利用できるようになります。

poudriere を利用する上でのより多くの情報については、 poudriere(8) およびメインのウェブサイトである https://github.com/freebsd/poudriere/wiki を参照してください。

4.6.2. Poudriere リポジトリを使うための pkg クライアントの設定

カスタムリポジトリと公式のリポジトリの両方を並行して使用することは可能ですが、
公式リポジトリを無効にすると有用な場合があります。
このように設定するには、設定ファイルを作成し、
設定ファイルの中で公式リポジトリを無効にしてください。 /usr/local/etc/pkg/repos/FreeBSD.conf を作成して、以下を含めてください。

```
FreeBSD: {
    enabled: no
}
```

通常は、HTTP 経由で poudriere リポジトリをクライアントコンピュータに公開すると簡単です。 package ディレクトリ (たとえば、 /usr/local/poudriere/data/packages/13amd64) を公開するようにウェブサーバを設定してください。 この例で 13amd64 は構築名です。

もし、package リポジトリの URL が `http://pkg.example.com/13amd64` であれば、リポジトリの設定ファイルである /usr/local/etc/pkg/repos/custom.conf は、 以下のようになります。

```
custom: {
    url: "http://pkg.example.com/13amd64",
    enabled: yes,
}
```

4.7. インストール後の作業

バイナリ package もしくは port のどちらを用いてソフトウェアをインストールするかに関わらず、
サードパーティ製のアプリケーションの多くは、インストール後にある程度の設定を必要とします。
以下のコマンドや場所の情報は、
アプリケーションとともに何がインストールされたかを知るための助けとなるでしょう。

- 多くのアプリケーションでは、デフォルトの設定ファイルが、少なくとも一つは /usr/local/etc にインストールされます。数多くの設定ファイルを持つようなアプリケーションでは、それらのファイルを格納するためにサブディレクトリを作成するものもあります。
  サンプルの設定ファイルは、しばしば .sample といった拡張子がついた名前でインストールされます。設定ファイルを確認し、必要に応じてシステムの要求に合うように編集してください。最初にサンプルファイルを .sample を外した名前のファイルにコピーしてから、編集してください。
- ドキュメントが付属しているアプリケーションは、ドキュメントを /usr/local/share/doc にインストールします。
  また、多くのアプリケーションは、マニュアルページもインストールします。
  これらのドキュメントは、アプリケーションを使い続ける前に見ておくべきものです。
- ある種のアプリケーションでは、サービスを実行するためには、アプリケーションの起動前に、/etc/rc.conf に追加する必要があります。これらのアプリケーションでは、通常、スタートアップスクリプトが /usr/local/etc/rc.d にインストールされます。詳細は、サービスの起動をご覧ください。

設計上、インストールの際に、アプリケーションは、
スタートアップスクリプトを実行しませんし、
アンインストールやアップグレードの際には、
停止のためのスクリプトは実行されません。
起動や停止の決定は、各システム管理者に任されています。

- csh(1) のユーザは、 `rehash` を実行して、 シェルの `PATH` のバイナリリストを再構築してください。
- `pkg info` を使って、アプリケーションと共にインストールされたファイル、マニュアルページ、およびバイナリを調べることができます。

4.8. うまく動作しない ports に遭遇した場合には

port をうまくコンパイルできなかったりインストールできない場合には、 以下を試してください。

1. その port に対する修正案が提出されていないかどうかを 障害報告 (Problem Report) データベースで調べてください。もし提案されていれば、その提案されている修正によって問題を解決できるかもしれません。

2. port の保守担当者に対応してもらいましょう。 port スケルトンで `make maintainer` と入力するか、port の Makefile を読み、保守担当者の電子メールアドレスを調べてください。保守担当者にメールを送る際には、エラーが出力されるまでの出力ログを忘れずに添付してください。

特定の保守担当者が存在せず、かわりにメーリングリストによるグループの管理者が保守している ports があります。そのような場合には、メールアドレスは freebsd-listname@FreeBSD.org のようになります。 メールを送る際には、このことに気をつけてください。

特に ports@FreeBSD.org が保守している ports には、保守担当者がいません。そのかわり、そのメーリングリストを購読する人々からなるコミュニティが、修正や対応をおこなっています。 もっとボランティアが必要です!

メールに対して返信がなければ、FreeBSD 障害報告の書き方 に書かれている手順にしたがい、Bugzilla を使ってバグレポートを提出してください。

3. 自分で直しましょう! ports システムに関する詳細な情報は port 作成者のためのハンドブック にあります。 このセクションを読むと、壊れてしまった port を直したり、 自分で作った port を提出したりできるようになります!

4. pkg によるバイナリ package の管理 に書かれている手順にしたがって、 package をインストールしてください。

Chapter 5. X Window System

5.1. この章では

bsdinstall を用いた FreeBSD のインストールでは、グラフィカルユーザインタフェースは自動的にはインストールされません。
この章では、グラフィカル環境で使われるオープンソースの X Window System を提供する Xorg のインストールおよび設定方法について説明します。その後、デスクトップ環境およびウィンドウマネージャの探し方およびインストール方法について説明します。

Xorg を自動的に設定するインストール方法を希望するユーザは、GhostBSD, MidnightBSD または NomadBSD を参照してください。

Xorg が対応するビデオハードウェアについてのより多くの情報は、x.org のウェブサイトをご覧ください。

この章を読めば以下のことがわかります。

- X Window System のさまざまなコンポーネントと、 それらが互いにどのように連携しているか。
- Xorg のインストールおよび設定方法
- さまざまなウィンドウマネージャおよびデスクトップ環境のインストールおよび設定方法
- Xorg での TrueType® フォントの使い方
- GUI ログイン (XDM) の設定方法

この章を読み始める前に以下のことを理解しておく必要があります。

- アプリケーションのインストール - packages と ports で説明されているサードパーティ製ソフトウェアのインストール方法

5.2. 用語の説明

X Window System のさまざまなコンポーネントについての詳細や、それらがどのようにやり取りするかについてすべて理解する必要はありませんが、
これらのコンポーネントについて基本的なことを知っていると、 強力な武器になるでしょう。

X サーバ

X は最初からネットワークを意識してデザインされており、 "クライアント - サーバ" モデルを採用しています。 このモデルでは、"X サーバ" はキーボードやモニタ、マウスが接続されたコンピュータ上で動きます。
このサーバはディスプレイの表示を管理したり、キーボード、 マウスからの入力を処理したり、タブレットやビデオプロジェクタ等の他の装置からの入出力を処理します。
これは、ある人々を混乱させることがあります。 X での用語は彼らが想定するものとは正反対だからです。 彼らは "X サーバ" は地下にある大きなパワフルなマシンであり、 "X クライアント" が自分たちのデスク上にあると想像するのです。

X クライアント

XTerm や Firefox などの各 X アプリケーションは、 "クライアント" になります。 クライアントは "この座標にウィンドウを描いてください" といったメッセージをサーバへ送り、サーバは "ユーザが OK ボタンを押しました" といったメッセージを送り返します。

家庭や小さなオフィスのような環境では、X サーバと X クライアントは通常同じコンピュータ上で動いています。 X サーバを非力なコンピュータで動かし、X アプリケーションをより高性能なマシンで動かすことも可能です。 この場合、X のクライアントとサーバの通信はネットワーク越しに行なわれます。

ウィンドウマネージャ

X はスクリーン上でウィンドウがどのように見えるべきか、 マウスでそれらをどうやって動かすか、ウィンドウ間を移動するのにどういうキーストロークを使うべきか、
各ウィンドウのタイトルバーはどのように見えるべきか、 クローズボタンを持つべきかどうか、といったことは規定しません。そのかわりに、X ではそういったことを "ウィンドウマネージャ" と呼ばれるアプリケーションに任せます。ウィンドウマネージャはたくさん あります。
これらのウィンドウマネージャの見た目や使い勝手はそれぞれ異なっています。
バーチャルデスクトップをサポートしているものもありますし、
デスクトップを操作するキーストロークをカスタマイズできたり、 "スタート" ボタンやそれに類するものを持っているものもあります。 テーマに対応しており、デスクトップの見た目や使い勝手を完全に変えられるものもあります。 ウィンドウマネージャは Ports Collection の x11-wm カテゴリに用意されています。

それぞれのウィンドウマネージャは異なる設定機構を備えています。
手で設定ファイルを編集しなければならないものや、 設定作業のほとんどを GUI ツールで行うことができるものもあります。

デスクトップ環境

KDE や GNOME は、デスクトップ環境です。
これらは、共通のデスクトップのタスクを実行するための完全なアプリケーションスイートを含んでいます。 オフィススイート、ウェブブラウザやゲームを含んでいるものもあります。

フォーカスポリシ

ウィンドウマネージャは、 マウスのフォーカスポリシに責任を持ちます。
このポリシは、どのウィンドウがアクティブにキーストロークを
受け付けるようにするための方法を提供し、
そして、どのウィンドウがアクティブなのかを示します。

よく知られているフォーカスポリシは "click-to-focus" と呼ばれるものです。 このポリシは、あるウィンドウ内でマウスをクリックすればそのウィンドウがアクティブになる、 というものです。 "focus-follows-mouse" ポリシでは、
マウスポインタの下にいるウィンドウがフォーカスされるというものです。
フォーカスを変えるには他のウィンドウにマウスポインタを動かすだけです。
マウスがルートウィンドウに移動した時には、 このウィンドウがフォーカスされます。 "sloppy-focus" モデルでは、 マウスがルートウィンドウに移動した時には、直前に使われていたウィンドウがフォーカスされています。 sloppy-focus では、ポインタが別のウィンドウに移った時のみフォーカスが変わり、
現在のウィンドウから出ただけでは変わりません。 "click-to-focus" ポリシでは、マウスクリックによりアクティブなウィンドウが選択されます。

ウィンドウは前面に表示され、他のすべてのウィンドウの前にきます。
ポインタが別のウィンドウ上に移動した時でも、
すべてのキーストロークがこのウィンドウに届きます。

それぞれのウィンドウマネージャは、 それぞれのフォーカスポリシに対応しています。
すべてのものは click-to-focus をサポートしていますし、
多くのものは他の方法もサポートしています。
どのフォーカスモデルを利用可能かどうかについては、
ウィンドウマネージャのドキュメントをご覧ください。

ウィジェット

ウィジェットはクリック可能であったり、
他の方法で操作可能なすべてのユーザインタフェース用アイテムを指す用語です。
ボタンやチェックボックス、ラジオボタン、アイコン、リスト、などがそうです。
ウィジェットツールキットはグラフィカルアプリケーションを作成するために使われます。 KDE で使われている Qt や GNOME プロジェクトで使われている GTK+ といった有名なウィジェットセットがあります。
そのため、アプリケーションのルックアンドフィールは、
アプリケーションを作成するのに使われたウィジェットツールキットに依存し、 異なります。

5.3. Xorg のインストール

FreeBSD では、Xorg を package または port からインストールできます。

バイナリ package を使うと早くインストールできますが、
カスタマイズのためのオプションは少なくなります。

```
# pkg install xorg
```

Ports Collection からビルドしてインストールするには、 以下のように実行してください。

```
# cd /usr/ports/x11/xorg
# make install clean
```

どちらの方法でも、完全な Xorg システムがインストールされます。 バイナリ package を用いる方法が、ほとんどのユーザにとってはベストな選択となります。

経験のあるユーザ向けの最小の X システムは、x11/xorg-minimal です。 ほとんどのドキュメント、ライブラリおよびアプリケーションはインストールされません。
アプリケーションによってはこれらの追加の要素が機能する上で必要となります。

5.4. Xorg の設定

5.4.1. クィックスタート

Xorg は、 標準的なほとんどのビデオカード、 キーボード、ポインティングデバイスに対応しています。

ビデオカード、キーボード、入力デバイスは、
自動的に検出されるので、手動の設定は必要ありません。
自動認識に失敗したとき以外は、xorg.conf を作成したり、-configure
プロセスの実行は行わないでください。

1. もし、使用しているコンピュータですでに Xorg が使われているのであれば、
コンフィグレーションファイルを移動するか、削除してください。

```
# mv /etc/X11/xorg.conf ~/xorg.conf.etc
# mv /usr/local/etc/X11/xorg.conf ~/xorg.conf.localetc
```

2. 3D アクセラレータを利用できるシステムでは、 Xorg を実行するユーザを video または wheel
グループに追加して、使用できるようにしてください。 ユーザ *jru*
をどちらのグループでも利用できるようにするには以下のように実行してください。

```
# pw groupmod video -m jru || pw groupmod wheel -m jru
```

3. デフォルトでは twm ウィンドウマネージャがインストールされています。 Xorg が起動すると、
このウィンドウマネージャが立ち上がります。

```
% startx
```

4. 古いバージョンの FreeBSD では、 テキストコンソールに戻れるようにするために、
システムコンソールは vt(4) に設定する必要があります。 Kernel Mode Setting (KMS)
を参照してください。

5.4.2. Accelerated Video のためのユーザグループ

ビデオカードの 3D アクセラレータを有効にするには、 /dev/dri へのアクセスが必要となります。
通常は、X を実行するユーザを video または wheel グループに追加するするだけです。 ここでは、pw(8)
を使ってユーザ *slurms* を video グループ、または video グループが存在しない時に、 wheel
グループに追加しています。

```
# pw groupmod video -m slurms || pw groupmod wheel -m slurms
```

5.4.3. Kernel Mode Setting (KMS)

コンピュータが、コンソールの表示から、 X 用の高解像度の表示へと切り替える時には、 ビデオの出力
mode が設定されている必要があります。 最近の Xorg では、
カーネル内部のシステムを使って効率的にこれらのモードの変換をしています。 古いバージョンの
FreeBSD では、 KMS システムを用いない sc(4) が使用されています。 X
を閉じた後、システムコンソールは動作をしていても、 表示に黒になります。 新しい vt(4)
コンソールではこの問題は起こりません。

以下の行を /boot/loader.conf に追加して vt(4) を有効にしてください。

```
kern.vty=vt
```

5.4.4. コンフィグレーションファイル

通常、この節で説明する手動の設定は必要ありません。 自動認識に失敗したとき以外は、
手動で設定ファイルを作成しないでください。

ディレクトリ

Xorg は、 複数のディレクトリから設定ファイルを探します。 FreeBSD
において、設定ファイルのディレクトリは、 /usr/local/etc/X11/ が推奨されます。
このディレクトリを使うことで、
アプリケーションのファイルをオペレーティングシステムとは区別する事になります。

昔のコンフィグレーションファイルの置き場である /etc/X11/ も機能します。
しかしながら、この場所に置くと、アプリケーションファイルと FreeBSD
システムのファイルが混ざってしまうため、推奨されません。

単一または複数ファイル

複数のファイルを用いて、 各ファイルが特定の部分を設定するようにすると、 古い単一の xorg.conf
を用いるよりも設定が簡単になります。 これらのファイルは、
メインのコンフィグレーションファイルのディレクトリの xorg.conf.d/ サブディレクトリに置かれます。
フルパスは、一般的に /usr/local/etc/X11/xorg.conf.d/ となります。

これらのファイルの例は、この節の後半で説明します。

古い単一の xorg.conf も機能しますが、 xorg.conf.d/
サブディレクトリに複数のファイルで設定する形式に比べると、
柔軟ではなく、わかりにくいものとなります。

5.4.5. ビデオカード

Ports のフレームワークは、X11 が最近のハードウェア上で動作するために必要となる drm
グラフィックドライバを提供します。 そのようなドライバとしては、graphics/drm-kmod
で提供されるドライバがあります。
これらのドライバは、通常プライベートなカーネルのインタフェースを使用するため、PORTS_MODULES
変数を利用して ports システムのドライバを構築することが強く推奨されています。 PORTS_MODULES
に設定されたカーネルモジュールを含む port
は、カーネルを構築するたびに、アップデートされたソースに対応するよう再構築されます。
これにより、カーネルモジュールはカーネルと同期することが保証されます。 カーネルと ports
ツリーは最大限に互換性を持つように共にアップデートされる必要があります。 /etc/make.conf
ファイルで PORTS_MODULES
を設定することで、カーネルを構築する際に、設定されたモジュールも再構築されるようになります。
上級のユーザの中には、カーネルコンフィグレーションファイルに makeoptions
ディレクティブで追加するユーザもいます。 GENERIC を走らせていて、freebsd-update
を使用している場合には、freebsd-update install 実行後に graphics/drm-kmod または x11/nvidia-driver port を構築してください。

/etc/make.conf

```
SYSDIR=path/to/src/sys
PORTS_MODULES=graphics/drm-kmod x11/nvidia-driver
```

この設定はすべてを再構築します。 使用している GPU /グラフィックカードに応じて、ひとつだけ選択したり、別のドライバを選択できます。

Intel KMS driver

Intel® が提供しているほとんどの Intel KMS driver グラフィックカードは、2D および 3D アクセラレーションに対応しています。

ドライバ名: `i915kms`

AMD® が提供している古い Radeon KMS driver グラフィックカードのほとんどは、2D および 3D アクセラレーションに対応しています。

ドライバ名: `radeonkms`

AMD® が提供している新しい Radeon KMS driver グラフィックカードのほとんどは、2D および 3D アクセラレーションに対応しています。

ドライバ名: `amdgpu`

参考として、対応している GPU 一覧を https://en.wikipedia.org/wiki/List_of_Intel_graphics_processing_units または https://en.wikipedia.org/wiki/List_of_AMD_graphics_processing_units でご覧ください。

Intel®

Iron Lake (HD Graphics) および Sandy Bridge (HD Graphics 2000) を含む Ivy Bridge (HD Graphics 2500, 4000, および P4000) までのほとんどの Intel® グラフィックスは、3D acceleration に対応しています。

ドライバ名: `intel`

参考情報については https://en.wikipedia.org/wiki/List_of_Intel_graphics_processing_units をご覧ください。

AMD® Radeon

ATI/Radeon: 2D および 3D acceleration は、 HD6000 シリーズまでのほとんどの Radeon カードで対応しています。

ドライバ名: `radeon`

参考情報については https://en.wikipedia.org/wiki/List_of_AMD_graphics_processing_units をご覧ください。

NVIDIA

NVIDIA: いくつかの NVIDIA ドライバが Ports Collection の x11 カテゴリから利用できます。

ビデオカードのモデルに対応するドライバをインストールしてください。

参考情報については https://en.wikipedia.org/wiki/List_of_Nvidia_graphics_processing_units をご覧ください。

NVIDIA カードに対応しているカーネルは x11/nvidia-driver port または x11/nvidia-driver-xxx port にあります。 最新のカードは、前者の port を使います。 古いカードは、-xxx ports を使うことになります。 ここで xxx はドライバのバージョンを表す 304, 340 または 390 のどれかです。 これらのドライバを使うには、 対応している NVIDIA GPU プロトコル ページを参照して、`-xxx` の部分を埋めてください。
このページには、ドライバの各バージョンが対応しているデバイスの一覧があります。
昔のドライバは i386 および amd64 で動作します。 現在のドライバは amd64 のみに対応しています。 詳細については、 NVIDIAドライバのインストールおよび設定 を参照してください。
ドライバが最大限に安全に動作するように、カーネルを再構築すたびにドライバを再構築することが推奨されていますが、このドライバはプライベートのカーネルインタフェースをほとんど使わないため、通常はカーネルがアップデートしても安全に使用できます。

ハイブリッドグラフィックス

ノートブックコンピュータによっては、
チップセットまたはプロセッサに組み込まれているグラフィックプロセッサユニットの他に、
追加でそれらを持つものがあります。 *Optimus* は、 Intel® と NVIDIA ハードウェアを組み合わせています。 *Switchable Graphics* または、 *Hybrid Graphics* は、 Intel® または AMD® プロセッサと AMD® Radeon GPU を組み合わせています。

これらのハイブリッドなグラフィックシステムの実装は、 システムごとに異なるので、 FreeBSD の Xorg は、 これらのすべてのバージョンについて対応しているわけではありません。

コンピュータによっては、 片方のグラフィックアダプタを無効にしたり、
標準のビデオカードドライバの一つとともに使われる *discrete* モードを選択できるような BIOS オプションを提供しています。 たとえば、Optimus システムでは、NVIDIA GPU を無効にできるものがあります。 その後、Intel® ビデオカードは、 Intel® ドライバで利用できます。

BIOS の設定は、 コンピュータのモデルに依存します。 システムによっては、両方の GPU を有効にできますが、 そのようなシステムの機能を利用するには、 `Device` セッションにおいて、メインの GPU のみを使用するようなコンフィグレーションファイルを作成ことで十分です。

他のビデオカード

Ports Collection の x11-drivers ディレクトリには、
あまり使用されないようなドライバも用意されています。

特定のドライバによりサポートされていないようなカードでも、 x11-drivers/xf86-video-vesa で使用できるかもしれません。 このドライバは、x11/xorg によりインストールされます。
手動でインストールするには、 x11-drivers/xf86-video-vesa としてインストールしてください。
ビデオカードに対して、特定のドライバが見つからない場合には、 Xorg はこのドライバを使うことを試みます。

x11-drivers/xf86-video-scfb も同様に、多くの UEFI および ARM® コンピュータで動くような、使用するカードを特定していないビデオドライバです。

ファイルでビデオドライバを設定する。

コンフィグレーションファイルにおいて Intel®
ドライバを設定するには、以下のようにしてください。

例 *14.* ファイルにおいて *Intel®* ビデオドライバを選択する。

/usr/local/etc/X11/xorg.conf.d/driver-intel.conf

```
Section "Device"
    Identifier "Card0"
    Driver     "intel"
    # BusID    "PCI:1:0:0"
EndSection
```

１つ以上のビデオカードが存在する場合には、 `BusID` 行のコメントを外し、希望するカードを選択するように設定できます。 ビデオカードバス ID は、 `pciconf -lv | grep -B3 display` で表示できます。

コンフィグレーションファイルで、Radeon ドライバを設定するには以下のようにしてください。

例 *15.* ファイルにおいて *Radeon* ビデオドライバを設定する。

/usr/local/etc/X11/xorg.conf.d/driver-radeon.conf

```
Section "Device"
    Identifier "Card0"
    Driver     "radeon"
EndSection
```

コンフィグレーションファイルで VESA ドライバを設定するには、以下のようにしてください。

例 *16.* ファイルで *VESA* ビデオドライバを設定する。

/usr/local/etc/X11/xorg.conf.d/driver-vesa.conf

```
Section "Device"
    Identifier "Card0"
    Driver     "vesa"
EndSection
```

UEFI または ARM® コンピュータを使うために `scfb`
ドライバを設定するには、以下のように設定してください。

例 17. ファイルの中で scfb ビデオドライバを選択する。

/usr/local/etc/X11/xorg.conf.d/driver-scfb.conf

```
Section "Device"
    Identifier "Card0"
    Driver     "scfb"
EndSection
```

5.4.6. モニタ

ほとんどすべてのモニタは、Extended Display Identification Data standard (EDID) に対応しています。 Xorg は EDID を使ってモニタと通信し、対応している解像度とリフレッシュレートを検出します。
そのため、モニタを使用するのに最も適切な設定が選択されます。

モニタにより対応している他の解像度は、 コンフィグレーションファイルに希望する解像度を設定する、または X サーバを起動後、xrandr(1) により選択が可能となります。

xrandr(1) の使用

パラメータを与えずに xrandr(1) を実行すると、ビデオ出力と検出されているモニタのモードを確認できます。

```
% xrandr
Screen 0: minimum 320 x 200, current 3000 x 1920, maximum 8192 x 8192
DVI-0 connected primary 1920x1200+1080+0 (normal left inverted right x axis y axis)
495mm x 310mm
   1920x1200     59.95*+
   1600x1200     60.00
   1280x1024     85.02    75.02    60.02
   1280x960      60.00
   1152x864      75.00
   1024x768      85.00    75.08    70.07    60.00
   832x624       74.55
   800x600       75.00    60.32
   640x480       75.00    60.00
   720x400       70.08
DisplayPort-0 disconnected (normal left inverted right x axis y axis)
HDMI-0 disconnected (normal left inverted right x axis y axis)
```

この出力からは、リフレッシュレート約 60 Hz で、 スクリーン解像度 1920x1200 ピクセルの表示に DVI-0 出力が使用されていることが分かります。 また、DisplayPort-0 および HDMI-0 インタフェースには、 モニタは接続されていません。

xrandr(1) を使用して、 他のディスプレイモードを選択できます。 たとえば、60 Hz で、1280x1024 の表示に変更するには、 以下のように実行してください。

```
% xrandr --output DVI-0 --mode 1280x1024 --rate 60
```

ノートブックコンピュータの外部出力を使用して、
ビデオプロジェクタに接続することがよく行われます。

出力端子のタイプおよび番号は、デバイスごとに異なります。
また、各端子の名前もドライバごとに異なります。 あるドライバが HDMI-1 と呼ぶ出力が、別のドライバでは HDMI1 と呼ばれることもあります。 そのため、最初に xrandr(1) を実行して、利用可能な出力のすべての一覧を表示してください。

```
% xrandr
Screen 0: minimum 320 x 200, current 1366 x 768, maximum 8192 x 8192
LVDS1 connected 1366x768+0+0 (normal left inverted right x axis y axis) 344mm x
193mm
   1366x768      60.04*+
   1024x768      60.00
   800x600       60.32    56.25
   640x480       59.94
VGA1 connected (normal left inverted right x axis y axis)
   1280x1024     60.02 +  75.02
   1280x960      60.00
   1152x864      75.00
   1024x768      75.08    70.07    60.00
   832x624       74.55
   800x600       72.19    75.00    60.32    56.25
   640x480       75.00    72.81    66.67    60.00
   720x400       70.08
HDMI1 disconnected (normal left inverted right x axis y axis)
DP1 disconnected (normal left inverted right x axis y axis)
```

この出力からは、組み込みパネルの LVDS1, 外部出力の VGA1, HDMI1, そして DP1 端子の 4 つの出力を確認できます。

プロジェクタは VGA1 出力に接続されています。 情報を得られたので、xrandr(1) を使ってプロジェクタの標準の解像度に設定し、 デスクトップの右側にスペースを追加できます。

```
% xrandr --output VGA1 --auto --right-of LVDS1
```

この設定において、--auto は、 EDID により検出された解像度とリフレッシュレートを選択します。解像度を正しく検出できていない場合には、 --auto のかわりに、 --mode を使うことで、解像度を固定値を与えることにより設定できます。 たとえば、ほとんどのプロジェクタでは 1024x768 の解像度で使用できるので、 この場合には、--mode 1024x768 のように設定できます。

xrandr(1) は、X を起動する際に、 適切なモードを設定するように、しばしば .xinitrc から実行されます。

モニタ解像度をファイルで設定する。

コンフィグレーションファイルでスクリーンの解像度を 1024x768

と設定するには以下のようにしてください。

例 *18.* スクリーンの解像度をファイルで設定する。

/usr/local/etc/X11/xorg.conf.d/screen-resolution.conf

```
Section "Screen"
    Identifier "Screen0"
    Device     "Card0"
    SubSection "Display"
    Modes      "1024x768"
    EndSubSection
EndSection
```

EDID を持っていないモニタもあります。その場合には、 モニタが対応している周波数の範囲を、 HorizSync および VertRefresh で、指定することで設定できます。

例 *19.* 手動でモニタの周波数を設定する。

/usr/local/etc/X11/xorg.conf.d/monitor0-freq.conf

```
Section "Monitor"
    Identifier   "Monitor0"
    HorizSync    30-83   # kHz
    VertRefresh  50-76   # Hz
EndSection
```

5.4.7. 入力デバイス

キーボード

キーボードレイアウト

キーボード上の標準化されたキーの位置を レイアウト と呼びます。
レイアウトと他の調整可能なパラメータについては、 xkeyboard-config(7) にまとめられています。

アメリカ合衆国のレイアウトがデフォルトです。 他のレイアウトを選択するには、 InputClass で、 XkbLayout および XkbVariant オプションを設定してください。
クラスに対応するすべての入力デバイスに適用できます。

以下の例では、 フランス語のキーボードレイアウトを選択しています。

例 *20.* キーボードレイアウトを選択する。

/usr/local/etc/X11/xorg.conf.d/keyboard-fr.conf

```
Section "InputClass"
```

```
    Identifier  "KeyboardDefaults"
    MatchIsKeyboard "on"
    Option      "XkbLayout" "fr"
EndSection
```

例 *21.* 複数のキーボードレイアウトを選択する。

アメリカ合衆国、スペイン、　　ウクライナのキーボードレイアウトを、　　Alt + Shift によって切り替えるようにするには以下のように設定します。
レイアウトスイッチングコントロールや現在のレイアウトインディケータを改良するには、x11/xxkb または、 x11/sbxkb を使ってください。

/usr/local/etc/X11/xorg.conf.d/kbd-layout-multi.conf

```
Section "InputClass"
    Identifier  "All Keyboards"
    MatchIsKeyboard "yes"
    Option      "XkbLayout" "us, es, ua"
EndSection
```

キーボードから **Xorg** を終了する。

X　　をキーの組み合わせで終了できるように設定できます。
デフォルトでは、幾つかのアプリケーションで、
キーボードコマンドと衝突してしまう可能性があるため、
このキーの組み合わせは設定されていません。　　このオプションを有効にするには、　　キーボードの InputDevice セクションを変更してください。

例 *22.* キーボードからの *X* の終了を有効にする。

/usr/local/etc/X11/xorg.conf.d/keyboard-zap.conf

```
Section "InputClass"
    Identifier  "KeyboardDefaults"
    MatchIsKeyboard "on"
    Option      "XkbOptions" "terminate:ctrl_alt_bksp"
EndSection
```

マウスおよびポインティングデバイス

もし　FreeBSD　12.1　において、　moused(8)　を使わず、　xorg-server　1.20.8 以降を使用しているのであれば、　/etc/sysctl.conf　に、　kern.evdev.rcpt_mask=12 を追加してください。

コンフィグレーションオプションにより、　　多くのマウスパラメータを調整できます。
すべての一覧については、mousedrv(4) をご覧ください。

マウスボタン

xorg.conf のマウス `InputDevice` セクションで、 マウスのボタンの数を設定できます。 ボタンの数を 7 に設定するには、 以下のように設定してください。

例 *23.* マウスボタンの数を設定する。

/usr/local/etc/X11/xorg.conf.d/mouse0-buttons.conf

```
Section "InputDevice"
    Identifier  "Mouse0"
    Option      "Buttons" "7"
EndSection
```

5.4.8. 手動による設定

ハードウェアによっては、Xorg の自動設定で適切な設定が行われなかったり、
自動設定とは別の設定にしたいときがあります。 そのような場合のため、
カスタムコンフィグレーションファイルを作成できます。

自動認識に失敗したとき以外は、 手動で設定ファイルを作成しないでください。
不必要な手動の設定を行った結果、 適切に動作しなくなるということがあります。

検出されたハードウェアをベースとした、 Xorg のコンフィグレーションファイルを作成できます。
このファイルは、 カスタムコンフィグレーションファイルの最初の出発点として有用です。

以下のようにすると xorg.conf が生成されます。

```
# Xorg -configure
```

このコンフィグレーションファイルは、 /root/xorg.conf.new として保存されます。
必要となる変更を行った後、このファイルを (バックグラウンドが表示されるように `-retro` を使って)
テストしてください。

```
# Xorg -retro -config /root/xorg.conf.new
```

新しい設定を調整してテストしたら、 ファイルに分割して、標準の場所である、
/usr/local/etc/X11/xorg.conf.d/ に置いてください。

5.5. Xorg でのフォントの使用

5.5.1. Type1 フォント

Xorg に付いてくるデフォルトのフォントは、
通常のデスクトップパブリッシングアプリケーションにとっては理想的とは言えない程度のものです。
文字を大きくするとジャギーになりプロフェッショナルとは言えないようなものになりますし、

小さなフォントは頭が悪そうに見えます。 しかし、世の中には質の高い Type1 (PostScript®) フォントがいくつかあり、 Xorg ではそれらを簡単に利用することができます。 例えば、URW フォントコレクション (x11-fonts/urwfonts) には高品質の Type1 フォント (Times Roman™, Helvetica™, Palatino™ など) が含まれています。freefont コレクション (x11-fonts/freefonts) にはもっとたくさんのフォントが含まれていますが、 それらは Gimp のようなグラフィックソフトウェアで使用するためのものであり、
スクリーンフォントとしては十分ではありません。 さらに、Xorg は簡単に TrueType® フォントを使うように設定することも可能です。 詳しくは、X(7) のマニュアルページか TrueType® フォント を参照してください。

上記の Type1 フォントコレクションをバイナリ package からインストールする場合には、次のコマンドを実行してください。

```
# pkg install urwfonts
```

あるいは、Ports Collection から構築してインストールするには次のコマンドを実行してください。

```
# cd /usr/ports/x11-fonts/urwfonts
# make install clean
```

freefont や他のコレクションでも同じようにします。 X サーバがこれらのフォントを検出できるようにするには X サーバ設定ファイル (/etc/X11/xorg.conf) の適切な場所に次のような行を加えます。

```
FontPath "/usr/local/share/fonts/urwfonts/"
```

別の方法としては、 X のセッション中に次のようなコマンドラインを実行します。

```
% xset fp+ /usr/local/share/fonts/urwfonts
% xset fp rehash
```

これは動くのですが、X のセッションが終了すると消えてしまいます。 消えないようにするには X の起動時に読み込まれるファイル (通常の `startx` セッションの場合は ~/.xinitrc, XDM のようなグラフィカルなログインマネージャを通してログインする時は ~/.xsession) に加えておきます。
三番目の方法は新しい /usr/local/etc/fonts/local.conf ファイルを使うことです。 これに関しては フォントのアンチエイリアス をご覧ください。

5.5.2. TrueType® フォント

Xorg には、 TrueType® フォントのレンダリング機能が組み込まれています。
この機能を実現するために 2 つの異なるモジュールがあります。 ここでは、freetype の方が他のフォントレンダリングバックエンドと整合性が高いので、
このモジュールを使うことにします。 freetype モジュールを使うためには /etc/X11/xorg.conf ファイルの `"Module"` セクションに以下の行を追加するだけです。

```
Load  "freetype"
```

さて、まずは TrueType® フォント用のディレクトリ (例えば /usr/local/share/fonts/TrueType) を作り、そこに TrueType® フォントをすべて放り込みましょう。 Apple® Mac® の TrueType® フォントは、そのままでは使うことができませんので注意してください。 Xorg で使うには UNIX®/MS-DOS®/Windows® 用のフォーマットでなければなりません。 ファイルを置いたら mkfontscale を使って fonts.dir ファイルを作り、 X のフォントレンダラが新しいファイルがイントールされたことを分かるようにしてください。
`mkfontscale` は package からインストールできます。

```
# pkg install mkfontscale
```

その後、ディレクトリに X フォントファイルのインデックスを作成してください。

```
# cd /usr/local/share/fonts/TrueType
# mkfontscale
```

次に TrueType® フォントのディレクトリをフォントパスに追加します。 Type1 フォントの場合と同じように、

```
% xset fp+ /usr/local/share/fonts/TrueType
% xset fp rehash
```

とするか、もしくは xorg.conf ファイルに `FontPath` 行を追加します。

これで Gimp や LibreOffice といったすべての X アプリケーションから TrueType® フォントを使うことができます。 (高解像度なディスプレイで見るウェブページ上のテキストみたいな) とても小さなフォントや (LibreOffice にあるような) 非常に大きなフォントもかなり綺麗に見えるようになることでしょう。

5.5.3. フォントのアンチエイリアス

/usr/local/share/fonts/ と ~/.fonts/ にあるすべての Xorg のフォントが、Xft に対応しているアプリケーションで自動的にアンチエイリアス表示できるようになりました。 KDE, GNOME および Firefox のような最新のアプリケーションは、Xft に対応しています。

どのフォントがアンチエイリアスされるかを制御するため、
もしくはアンチエイリアスの特性を設定するために、 /usr/local/etc/fonts/local.conf ファイルを作成 (すでに存在しているのなら編集) します。 多くの Xft フォントシステムの高度な機能をこのファイルを使って調整できます。
この節ではいくつか簡単なところだけを紹介します。 詳しくは、fonts-conf(5) をご覧ください。

このファイルは XML 形式でなければなりません。 大文字小文字の区別に注意を払い、すべてのタグが正しく閉じられているか確認してください。 ファイルは一般的な XML ヘッダで始まり、DOCTYPE 定義と `<fontconfig>` タグがその後にきます。

```
<?xml version="1.0"?>
        <!DOCTYPE fontconfig SYSTEM "fonts.dtd">
        <fontconfig>
```

すでに説明したように、 /usr/local/share/fonts/ と ~/.fonts/ にあるすべてのフォントは Xft 対応のアプリケーションで利用できます。 これら 2 つのディレクトリ以外に別のディレクトリを追加したいなら、 /usr/local/etc/fonts/local.conf に以下のような行を追加してください。

```
<dir>/path/to/my/fonts</dir>
```

新しいフォント、 特に新しいフォントディレクトリを追加したら、 フォントキャッシュを再構築してください。

```
# fc-cache -f
```

アンチエイリアスをかけることによって境界が少しぼやけ、 そのためにとても小さなテキストはさらに読みやすくなり、 大きなフォントでは "ギザギザ" が消えるのです。 しかし、普通のテキストにかけた場合には目が疲れてしまうこともあります。 14 ポイント以下のサイズのフォントについて、 アンチエイリアスをかけないようにするには次の行を加えます。

```
        <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>
```

いくつかの等幅フォントは、 アンチエイリアスをかけるとスペーシングがうまくいかなくなる場合があります。 特に KDE でその傾向があるようです。 解決策の一つとして、そういったフォントのスペーシングを 100 に設定する方法があります。 そうするためには次の行を加えてください。

```
    <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>
```

(これは固定サイズのフォントに "mono" という一般的な別名をつけます) そして以下を追加します。

```
    <match target="pattern" name="family">
        <test qual="any" name="family">
            <string>mono</string>
        </test>
        <edit name="spacing" mode="assign">
            <int>100</int>
        </edit>
    </match>
```

Helvetica の様なある種のフォントは、アンチエイリアスすると問題が起こるでしょう。
たいてい、フォントが縦に半分に切られて表示されます。
最悪の場合、アプリケーションがクラッシュします。これを回避するには、以下を local.conf に追加します。

```
    <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>
```

local.conf の編集を終えたら、ファイルの末尾が </fontconfig> タグで終わるようにしてください。
これを行わなければ、変更は無視されるでしょう。

ユーザは自分だけの設定を各自の ~/.config/fontconfig/fonts.conf に追加できます。
このファイルもこれまでの説明と同じく XML 形式を使います。

最後に一つ。LCD スクリーンではサブピクセルサンプリングが必要な場合があります。
これは、基本的には (水平方向に分かれている) 赤、緑、

青の各コンポーネントを別々に扱うことによって水平方向の解像度を良くするというもので、
劇的な結果が得られます。 これを有効にするには local.conf ファイルに次の行を加えます。

```
    <match target="font">
           <test qual="all" name="rgba">
               <const>unknown</const>
           </test>
           <edit name="rgba" mode="assign">
               <const>rgb</const>
           </edit>
       </match>
```

ディスプレイの種類にもよりますが、 `rgb` ではなく `bgr` や `vrgb`、もしくは `vbgr` の場合もあるので、 試してみて最も良いものを使ってください。

5.6. X ディスプレイマネージャ

Xorg は、 ログインセッションの管理に用いることのできる X ディスプレイマネージャ XDM を提供しています。XDM はどのディスプレイサーバに接続するかを選択でき、ログイン名とパスワードの組み合わせなど認証情報を入力できるグラフィカルなインタフェースを提供しています。

この章では、FreeBSD 上での X ディスプレイマネージャの設定方法について説明します。
デスクトップ環境によっては、 各環境独自のグラフィカルログインマネージャを提供しています。
GNOME ディスプレイマネージャの設定方法については、GNOME を参照してください。 また、KDE ディスプレイマネージャの設定方法については、KDE を参照してください。

5.6.1. XDM の設定

XDM をインストールするには、 x11/xdm package または port を使ってください。
インストール後、コンピュータの起動時に、XDM を起動するように設定するには、/etc/rc.conf に以下の行を追加してください。

```
xdm_enable="YES"
```

デフォルトでは、XDM は 9 番目の仮想端末で起動します。

XDM の設定用ディレクトリは /usr/local/etc/X11/xdm です。 このディレクトリには XDM の振る舞いや見た目を変更するために用いられるファイルや、XDM
の動作中にデスクトップを設定するためのスクリプトやプログラムがあります。 XDM 設定ファイルには、これらのフィアルの機能についてまとめられています。
これらのファイルの正確な文法や使用方法については、xdm(8) に記述されています。

表 7. *XDM* 設定ファイル

ファイル	説明
Xaccess	XDM に接続するためのプロトコルは X Display Manager Connection Protocol (XDMCP) と呼ばれます。 このファイルにはリモートのマシンからの XDMCP 接続をコントロールするためのルールセットが書かれます。 デフォルトでは、どのクライアントからの接続も拒否します。
Xresources	このファイルは、XDM ディスプレイの chooser およびログインスクリーンを設定します。 デフォルトの設定は、シンプルな長方形のログインウィンドウで、 コンピュータのホスト名がログインウィンドウの上部に大きなフォントで表示され、 その下に "Login:" および "Password:" のプロンプトが表示されます。 このファイルのフォーマットは Xorg のドキュメントで記述されている app-defaults ファイルのものと同じです。
Xservers	これは、chooser がログインの選択肢として提供するローカルおよびリモートのディスプレイの一覧です。
Xsession	ユーザのログイン時に XDM により実行されるデフォルトのセッションスクリプトです。 ~/.xsession に置かれているカスタマイズされたセッションスクリプトが優先されます。
Xsetup_*	これらは chooser やログインインタフェースが表示される前に自動的に実行されるアプリケーションです。 それぞれのディスプレイに対して、Xsetup_* (* はローカルのディスプレイ番号) という名前のついたスクリプトがあります。 典型的な使い方は `xconsole` のようなバックグラウンドで動かすプログラムを一つか二つ起動することです。
xdm-config	このマシンで動いているすべてのディスプレイのグローバルな設定
xdm-errors	このファイルにはサーバプログラムからのエラーが書き出されます。 XDM が起動しようとしているディスプレイがなんらかの理由でハングした場合、 このファイルのエラーメッセージを見てください。 これらのメッセージは各ユーザの ~/.xsession-errors ファイルにもセッション毎に書き出されます。
xdm-pid	現在動いている XDM のプロセス ID。

5.6.2. リモートアクセスの設定

デフォルトでは、XDM を使ってログインできるのは、同じシステムのユーザのみです。
あるディスプレイサーバに他のシステムのユーザが接続できるようにするためには、
アクセスコントロールのルールを編集し、 コネクションリスナを有効にする必要があります。

XDM が他のリモートコネクションを待ち受けるようにするためには、 /usr/local/etc/X11/xdm/xdm-config の `DisplayManager.requestPort` 行を、行頭に `!` を置くことでコメントアウトしてください。

```
! SECURITY: do not listen for XDMCP or Chooser requests
! Comment out this line if you want to manage X terminals with xdm
DisplayManager.requestPort:     0
```

変更点を保存して、XDM を再起動してください。
リモートアクセスを制限するには、/usr/local/etc/X11/xdm/Xaccess
にある例を参考にしたり、詳細について xdm(8) を参照してください。

5.7. デスクトップ環境

この節では、良く使われている 3 つのデスクトップ環境を FreeBSD
上でにインストールする方法について解説します。 デスクトップ環境とは、
単なるウィンドウマネージャから完全なデスクトップアプリケーションスイートまでカバーします。
Ports Collection の x11-wm カテゴリには、 100 を超えるデスクトップ環境が用意されています。

5.7.1. GNOME

GNOME はユーザフレンドリなデスクトップ環境です。
アプリケーションを起動したりステータスを表示するパネル、
デスクトップ、ツールおよびアプリケーション群、
そしてアプリケーションが互いにうまくやり取りできるような仕組みが含まれています。 FreeBSD 上の GNOME に関するもっと詳しい情報は、https://www.FreeBSD.org/gnome で見ることができます。
このウェブサイトには、FreeBSD での GNOME のインストール、設定、管理に関する多くの情報があります。

このデスクトップ環境は、package からインストールできます。

```
# pkg install gnome
```

ports から GNOME を構築するには、以下のコマンドを実行してください。 GNOME は大きなアプリケーションなので、 コンパイルには高速のコンピュータでも時間がかかります。

```
# cd /usr/ports/x11/gnome
# make install clean
```

GNOME を使用するには、 /proc ファイルシステムをマウントする必要があります。 以下を /etc/fstab に追加して、 システムの起動中にこのファイルシステムをマウントするように設定してください。

```
proc           /proc       procfs  rw  0   0
```

GNOME は、メッセージバスおよびハードウェアアブストラクションに D-Bus を使います。これらのアプリケーションは、GNOME の依存として自動的にインストールされます。/etc/rc.conf の中で、システムの起動時にスタートするように有効にしてください。

```
dbus_enable="YES"
```

インストール後、GNOME を起動するように Xorg を設定してください。最も簡単な方法は、GNOME ディスプレイマネージャ GDM を使うことです。GDM は、GNOME package または port の一部としてインストールされます。有効にするには、以下の行を /etc/rc.conf に追加してください。

```
gdm_enable="YES"
```

GNOME のすべてのサービスを、起動するようにしておくと良いでしょう。このように設定するには、以下の行を /etc/rc.conf に追加してください。

```
gnome_enable="YES"
```

システムを再起動すると、GDM が自動的に起動します。

GNOME を起動するもう一つの方法は、.xinitrc を適切に設定した後で、コマンドラインから `startx` と入力する方法です。.xinitrc が既にある場合には、ウィンドウマネージャを起動する行を /usr/local/bin/gnome-session を起動するように変更してください。このファイルが存在しなければ、次のコマンドで作成してください。

```
% echo "exec /usr/local/bin/gnome-session" > ~/.xinitrc
```

3 つめの方法は、XDM をディスプレイマネージャとして使う方法です。この場合は、実行可能な .xsession というファイルを作成してください。

```
% echo "exec /usr/local/bin/gnome-session" > ~/.xsession
```

5.7.2. KDE

KDE はもう一つの使いやすいデスクトップ環境です。このデスクトップは、統一されたルックアンドフィール、標準化されたメニューおよびツールバー、キーバインディング、カラースキーム、国際化、
一元化されたダイアログベースのデスクトップ設定とともに、
アプリケーションのスイートを提供します。KDE の詳細については http://www.kde.org/ をご覧ください。KDE に関する FreeBSD 特有の情報については、http://freebsd.kde.org をご覧ください。

KDE package をインストールするには以下のように実行してください。

```
# pkg install x11/kde5
```

KDE port を構築するには、以下のコマンドを使ってください。 port のインストールでは、インストールするアプリケーションを選択するためのメニューが表示されます。 KDE は大きなアプリケーションなので、 高速のコンピュータでもコンパイルには時間がかかります。

```
# cd /usr/ports/x11/kde5
# make install clean
```

KDE では、 /proc ファイルシステムをマウントする必要があります。 以下の行を /etc/fstab に追加して、システム起動時にこのファイルシステムが自動的にマウントされるように設定してください。

```
proc            /proc       procfs  rw  0   0
```

KDE は、 メッセージバスおよびハードウェアアブストラクションに D-Bus を使います。これらのアプリケーションは、KDE の依存として自動的にインストールされます。 /etc/rc.conf の中で、システムの起動時にスタートするように有効にしてください。

```
dbus_enable="YES"
```

KDE Plasma 5 から KDE のディスプレイマネージャ KDM の開発は終了しました。かわりに推奨されているのが SDDM です。 インストールするには、以下を実行してください。

```
# pkg install x11/sddm
```

その後、以下の行を /etc/rc.conf に追加してください。

```
sddm_enable="YES"
```

KDE Plasma を起動するもう一つの方法は、 コマンドラインから `startx` を実行する方法です。このコマンドを実行するには、~/.xinitrc に以下の行を追加してください。

```
exec ck-launch-session startplasma-x11
```

KDE Plasma を起動する 3 つめの方法は、 XDM を利用する方法です。この方法を使うには、以下のようにして実行可能な ~/.xsession を作成してください。

```
% echo "exec ck-launch-session startplasma-x11" > ~/.xsession
```

KDE Plasma を起動した後は、ビルトインヘルプシステムから、さまざまなメニューおよびアプリケーションの使用方法などのより詳しい情報を参照できます。

5.7.3. Xfce

Xfce は GNOME で使われている GTK+ ツールキットをベースにしたデスクトップ環境ですが、より軽量、
シンプルでかつ効率的でありながら使いやすいデスクトップ環境です。
すべての設定が可能で、メニュー、アプレットおよびアプリケーションランチャを含むメインパネル、ファイルマネージャ、サウンドマネージャを提供し、テーマに対応しています。
速くて軽く、効率的なため、古いマシンや遅いマシン、メモリの限られたマシンに向いています。Xfce に関する詳しい情報は http://www.xfce.org で得られます。

Xfce package をインストールするには、次のように実行してください。

```
# pkg install xfce
```

また、port を構築するには以下のようにしてください。

```
# cd /usr/ports/x11-wm/xfce4
# make install clean
```

Xfce は、メッセージバスに D-Bus を使います。これらのアプリケーションは Xfce の依存として自動的にインストールされます。/etc/rc.conf において、システム起動時に起動するように有効にしてください。

```
dbus_enable="YES"
```

GNOME や KDE とは異なり、Xfce は、ログインマネージャを提供していません。コマンドラインから `startx` を実行して Xfce を起動するには、以下のコマンドを使って、~/.xinitrc を作成してください。

```
% echo ". /usr/local/etc/xdg/xfce4/xinitrc" > ~/.xinitrc
```

もう一つの方法は XDM を用いる方法です。この方法を使うには、実行可能な .xsession を作成してください。

```
% echo ". /usr/local/etc/xdg/xfce4/xinitrc" > ~/.xsession
```

5.8. Compiz Fusion のインストール

魅力的な 3D 効果を使うと、デスクトップコンピュータを使う楽しさがさらに増えることでしょう。

Compiz Fusion のインストールは簡単ですが、設定の際には、port の文書には記載されていないような作業が必要となることがあります。

5.8.1. FreeBSD nVidia ドライバの設定

デスクトップ効果は、 グラフィックカードに極めて高い負荷をかけることがあります。 nVidia ベースのグラフィックカードにおいて、 良いパフォーマンスを出すには、 プロプリエタリなドライバが必要となります。
他のグラフィックカードを使っているユーザは、この節を飛ばし、 xorg.conf の設定に進んでください。

必要となる nVidia ドライバについては、 この問題に関する FAQ を参照して決めてください。

使用しているカードに対する適切なドライバが決まれば、 インストール作業は他の package をインストールするのと同じように簡単です。

たとえば、 最新のドライバをインストールするには以下のように実行してください。

```
# pkg install x11/nvidia-driver
```

このドライバはカーネルモジュールを作成するので、このモジュールをシステムの起動時に読み込むように設定する必要があります。 sysrc(8) を使用して起動時にモジュールを読み込むようにしてください。

```
# sysrc kld_list+="nvidia"
```

または、以下の行を /boot/loader.conf に追加してください。

```
nvidia_load="YES"
```

動作しているカーネルに、 カーネルモジュールを今すぐ読み込ませるには、 `kldload nvidia` のようなコマンドを実行してください。 しかしながら、Xorg のバージョンによっては、
起動時にドライバが読み込まれていないと正しく動かないもありますので、
注意してください。/boot/loader.conf を編集後は、再起動してください。
/boot/loader.conf
を間違って設定してしまうと、システムは適切に起動しない可能性があります。

読み込まれたカーネルモジュールを使うには、 通常は、xorg.conf ファイルの一つの行をプロプリエタリなドライバを使うように変更するだけです。

/etc/X11/xorg.conf において、 以下の行を探し出してください。

```
Driver      "nv"
```

この行を以下のように変更してください。

```
Driver      "nvidia"
```

いつものように GUI を起動すると、nVidia のスプラッシュが表示されます。
すべてはこれまで通りに動作するはずです。

5.8.2. デスクトップ効果のための xorg.conf の設定

Compiz Fusion を有効にするには /etc/X11/xorg.conf を変更する必要があります。

コンポジット効果を有効にするには、 以下のセクションを追加してください。

```
Section "Extensions"
    Option         "Composite" "Enable"
EndSection
```

以下のような "Screen" セクションの場所を見つけてください。

```
Section "Screen"
    Identifier     "Screen0"
    Device         "Card0"
    Monitor        "Monitor0"
    ...
```

("Monitor" の後に) 次の二つの行を追加してください。

```
DefaultDepth    24
Option         "AddARGBGLXVisuals" "True"
```

あなたが使用したいと考えているスクリーン解像度に対応する "Subsection" を探してください。 たとえば、1280x1024 を使用する予定であれば、 次のようなセクションを探してください。 もし希望の解像度の subsection がなければ、 手動でそのエントリを追加してください。

```
SubSection     "Display"
    Viewport    0 0
    Modes      "1280x1024"
EndSubSection
```

デスクトップコンポジットで 24 ビットのカラーが必要であれば、上述の subsection を以下のように変更してください。

```
SubSection     "Display"
    Viewport    0 0
    Depth       24
    Modes      "1280x1024"
EndSubSection
```

最後に、"Module" セクションに "glx" および "extmod" モジュールが読み込まれるように設定されていることを確認してください。

```
Section "Module"
```

```
    Load          "extmod"
    Load          "glx"
    ...
```

前述の設定は、 x11/nvidia-xconfig を (root 権限で) 実行することで自動的に設定できます。

```
# nvidia-xconfig --add-argb-glx-visuals
# nvidia-xconfig --composite
# nvidia-xconfig --depth=24
```

5.8.3. Compiz Fusion のインストールおよび設定

Compiz Fusion のインストールは、 他の package と同様に簡単です。

```
# pkg install x11-wm/compiz-fusion
```

インストールが終了したら、グラフィックデスクトップを起動して、
端末から以下のコマンドを通常のユーザで実行してください。

```
% compiz --replace --sm-disable --ignore-desktop-hints ccp &
% emerald --replace &
```

使っているウィンドウマネージャ (GNOME では、Metacity) が、 Compiz Fusion に置き換えられるため、 画面は数秒間ちらつきます。 Emerald がウィンドウデコレーション (たとえば、閉じる、最小化、最大化ボタンタイトルバーなど) を取り扱います。

このコマンドをスクリプトに変換して、 (たとえば GNOME デスクトップの "Sessions" に追加して) 起動時に自動的に実行されるようにすることもできます。

```
#! /bin/sh
compiz --replace --sm-disable --ignore-desktop-hints ccp &
emerald --replace &
```

これを、たとえば start-compiz という名前でホームディレクトリに保存して、
以下のように実行可能にしてください。

```
% chmod +x ~/start-compiz
```

GUI を使って、このスクリプトを (GNOME デスクトップの System, Preferences, Sessions にある) Startup Programs に追加してください。

すべての希望する効果と設定を選択するには、 (もう一度通常のユーザで) Compiz Config Settings Manager を実行してください。

```
% ccsm
```

GNOME では、 System, Preferences メニューから選択することも出来ます。

ビルドの際に "gconf support" を選択していたのであれば、 gconf-editor を使って apps/compiz 以下を見ることで、 これらの設定を確認することも出来ます。

5.9. トラブルシューティング

もしマウスが動作しなければ、 先へ進む前にマウスの設定を行う必要があります。 最近の Xorg では、デバイスの自動認識のため、 xorg.conf の InputDevice セクションは無視されます。 古い設定の記述を利用するには、 このファイルの ServerLayout もしくは、 ServerFlags セクションに以下の行を追加してください。

```
Option "AutoAddDevices" "false"
```

これで、以前のバージョンのように、入力デバイスを (キーボードレイアウトの変更のように) 必要なオプションを用いて設定できるようになります。

この章には部分的に古くなった情報が含まれています。 FreeBSD のデスクトップ設定に HAL デーモン (hald) はもう使われません。

すでに説明したように、デフォルトで hald デーモンがキーボードを自動的に認識します。
キーボードレイアウトやモデルを正しく認識しない場合でも、 GNOME, KDE もしくは Xfce のようなデスクトップ環境が、 キーボードの設定ツールを提供しています。 しかしながら、 setxkbmap(1) ユーティリティや hald の設定ルールを利用することで、 キーボードのプロパティを直接設定できます。

たとえば、フランス語のレイアウトの PC 102 キーボードを使いたい場合には、 hald のキーボード設定ファイル x11-input.fdi を作成し、 /usr/local/etc/hal/fdi/policy ディレクトリに保存してください。 このファイルは以下を含んでいる必要があります。

```
<?xml version="1.0" encoding="utf-8"?>
<deviceinfo version="0.2">
  <device>
    <match key="info.capabilities" contains="input.keyboard">
      <merge key="input.x11_options.XkbModel"
type="string">pc102</merge>
      <merge key="input.x11_options.XkbLayout" type="string">fr</merge>
    </match>
  </device>
</deviceinfo>
```

このファイルがすでに存在する場合には、

キーボードの設定に関する部分をただ単にコピーし、 ファイルに追加してください。

hald がこのファイルを読み込むように、 コンピュータを再起動してください。

X 端末やスクリプトから以下のコマンドラインを実行することでも、同様に設定できます。

```
% setxkbmap -model pc102 -layout fr
```

/usr/local/share/X11/xkb/rules/base.lst には、利用可能なキーボード、レイアウトおよびオプションの一覧があります。

xorg.conf.new 設定ファイルを好みに合うように調整できます。 emacs(1) や ee(1) のようなテキストエディタでファイルを開いてください。 古いモニタや、通常とは異なるモデルで、同期周波数の自動認識に対応していない場合には、 以下のような設定を xorg.conf.new の "Monitor" セクションの下に加えてください。

```
Section "Monitor"
        Identifier   "Monitor0"
        VendorName   "Monitor Vendor"
        ModelName    "Monitor Model"
        HorizSync    30-107
        VertRefresh  48-120
EndSection
```

ほとんどのモニタは同期周波数の自動認識に対応しているので、
これらの値を手動で入力する必要はありません。 自動認識に対応していないモニタでは、ダメージの可能性を避けるため、 メーカーが提供している値のみを入力してください。

X はモニタが対応していれば DPMS (Energy Star) 機能を使うことができます。 xset(1) プログラムでタイムアウトをコントロールしたり、
強制的にスタンバイ、サスペンドや電源オフにすることができます。 モニタの DPMS 機能を有効にしたい場合は、 "Monitor" セクションに次の行を加えてください。

```
Option       "DPMS"
```

xorg.conf.new 設定ファイルはエディタで開いたままにしておき、デフォルトの解像度と色数を好みで選んでください。 "Screen" セクションで定義されます。

```
Section "Screen"
        Identifier "Screen0"
        Device     "Card0"
        Monitor    "Monitor0"
        DefaultDepth 24
        SubSection "Display"
                Viewport  0 0
                Depth     24
```

```
                Modes     "1024x768"
        EndSubSection
EndSection
```

DefaultDepth というキーワードは 実行時のデフォルトの色数について記述するためのものです。 Xorg(1) のコマンドラインスイッチ -depth が使用された場合はそちらが優先されます。 Modes というキーワードは、 与えられた色数におけるデフォルトの解像度を記述しておくためのものです。 ターゲットのシステムのグラフィックハードウェアによって定義されている、 VESA スタンダードモードのみがサポートされていることに注意してください。
上の例ではデフォルトの色数はピクセルあたり 24 ビットであり、 この色数での解像度は 1024 ピクセル× 768 ピクセルです。

最後に、設定ファイルを保存し、 上の例にあるようにテストしてみてください。

トラブルシューティングの過程で助けとなるツールのひとつに Xorg のログファイルがあります。 これには、Xorg サーバが検知したデバイスそれぞれについての情報があります。 Xorg のログファイル名は /var/log/Xorg.0.log という形式です。実際のログファイル名は Xorg.0.log から Xorg.8.log のように変わります。

すべてうまくいったなら、設定ファイルを Xorg(1) が見つけることができる共通の場所に置きます。 これは、通常は /etc/X11/xorg.conf や /usr/local/etc/X11/xorg.conf です。

```
# cp xorg.conf.new /etc/X11/xorg.conf
```

これで Xorg の設定は完了です。 startx(1) ユーティリティで Xorg を起動できます。 xdm(8) を使って Xorg サーバを起動することもできます。

5.9.1. Intel® i810 グラフィックチップセットの設定

Intel® i810 統合チップセットを設定するには、 Xorg にカードを制御させるために AGP プログラミングインタフェースである agpgart が必要になります。 詳しくは、agp(4) ドライバのマニュアルページをご覧ください。

このドライバを用いることで、
他のグラフィックボードと同様に設定を行うことができるようになります。 カーネルに agp(4) ドライバが組み込まれていないシステムでは、 このモジュールを kldload(8) を使って読み込もうとしても動作しないことに注意してください。 このドライバは、 起動時にカーネル内に存在するようにカーネル内部に組み込むか、 /boot/loader.conf を使わなければなりません。

5.9.2. ワイドスクリーンフラットパネルの追加

この節では、設定に関する幾分高度な知識を必要とします。
これまでに述べた標準ツールを使って設定に失敗する場合は、 ログファイルを参照してください。 ログファイルには、 設定のために有用な情報が十分含まれています。 テキストエディタを使用する必要があるでしょう。

現在のワイドスクリーン（WSXGA，WSXGA+，WUXGA，WXGA，WXGA+ など）は、16:10 や 10:9 形式、または (問題を含む可能性のある) 他のアスペクト比に対応しています。 以下は、16:10 アスペクト比のスクリーン解像度の例です。

- 2560x1600
- 1920x1200
- 1680x1050
- 1440x900
- 1280x800

これらの解像度のひとつを以下のように "Screen" セクション の 可能な Mode に追加してください。

```
Section "Screen"
Identifier "Screen0"
Device     "Card0"
Monitor    "Monitor0"
DefaultDepth 24
SubSection "Display"
    Viewport  0 0
    Depth     24
    Modes     "1680x1050"
EndSubSection
EndSection
```

Xorg は、I2C/DDC 情報を通してワイドスクリーンの解像度に関する情報を取得できるので、モニタの周波数や解像度の範囲を把握しています。

もし、これらの ModeLines がドライバに存在しないのであれば、Xorg にヒントを与えなければならないでしょう。 ModeLine を手動で設定するのに十分な情報を /var/log/Xorg.0.log から得ることができます。 以下のような情報を探してください。

```
(II) MGA(0): Supported additional Video Mode:
(II) MGA(0): clock: 146.2 MHz   Image Size:  433 x 271 mm
(II) MGA(0): h_active: 1680  h_sync: 1784  h_sync_end 1960 h_blank_end 2240 h_border:
0
(II) MGA(0): v_active: 1050  v_sync: 1053  v_sync_end 1059 v_blanking: 1089 v_border:
0
(II) MGA(0): Ranges: V min: 48  V max: 85 Hz, H min: 30  H max: 94 kHz, PixClock max
170 MHz
```

これは EDID と呼ばれる情報です。 この情報を用いて ModeLine を作成するには、正しい順番に数字を入力するだけです。

```
 ModeLine <name> <clock> <4 horiz. timings> <4 vert. timings>
```

この例では Monitor セクション の ModeLine は以下のようになります。

```
Section "Monitor"
Identifier      "Monitor1"
VendorName      "Bigname"
ModelName       "BestModel"
ModeLine        "1680x1050" 146.2 1680 1784 1960 2240 1050 1053 1059 1089
Option          "DPMS"
EndSection
```

以上の簡単な編集作業が終わったら、 新しいワイドスクリーンモニタ上で X が動作するでしょう。

5.9.3. Compiz Fusion 使用時のトラブルシューティング

Compiz Fusion をインストールし、説明されたようにコマンドを実行すると、
ウィンドウのタイトルバーやボタンが表示されません。 何が問題でしょうか？

おそらく /etc/X11/xorg.conf の設定が行われていていないのでしょう。
このファイルを詳細に確認してください。特に `DefaultDepth` および `AddARGBGLXVisuals`
ディレクティブを確認してください。

Compiz Fusion を起動するコマンドを実行すると、**X** サーバがクラッシュし、
コンソールに戻ります。何が問題でしょうか？

/var/log/Xorg.0.log ファイルを確認すると、 X の起動時のエラーメッセージを探し出すことができます。
多くの場合は、以下のようなものです。

```
(EE) NVIDIA(0):     Failed to initialize the GLX module; please check in your X
(EE) NVIDIA(0):     log file that the GLX module has been loaded in your X
(EE) NVIDIA(0):     server, and that the module is the NVIDIA GLX module.  If
(EE) NVIDIA(0):     you continue to encounter problems, Please try
(EE) NVIDIA(0):     reinstalling the NVIDIA driver.
```

これは通常 Xorg をアップグレードした時に起きる現象です。 x11/nvidia-driver package
をインストールして glx を再構築してください。

Part II: 日々の生活

第 1 部では基礎的なことがらを説明したので、 ハンドブックの第 2 部では FreeBSD でよく使われる機能について説明します。 各章の内容は以下のとおりです。

- ブラウザ、生産的なツール、ドキュメントビューアといった、
  人気があって便利なデスクトップアプリケーションの紹介
- FreeBSD で利用可能なマルチメディアツールの紹介
- 特別な機能を有効にするために、 カスタム FreeBSD カーネルを構築する手順の説明
- デスクトップおよびネットワーク接続両方のプリンタの設定に関する、 印刷システムの詳細な説明
- FreeBSD システムで Linux アプリケーションを実行する方法

これらの章では、読み飛ばしを推奨しているものもあります。
これについてはそれぞれの章の始めにある概要に書かれています。

Chapter 6. デスクトップアプリケーション

6.1. この章では

FreeBSD は性能や安定性によりサーバとして人気がある一方で、日々のデスクトップとしての利用にも適しています。 packages や ports から 37000 を超えるアプリケーションを利用できるので、
さまざまなアプリケーションを動かせるようにカスタマイズしたデスクトップを作り上げることができます。 この章では、ウェブブラウザ、生産的なソフトウェア、ドキュメントビューア、および財務管理ソフトウェアといった、
数多くのデスクトップアプリケーションのインストール方法について説明します。

一から構築するのではなく、 事前に構築されたデスクトップバージョンの FreeBSD をお望みのユーザは、FuryBSD, GhostBSD および MidnightBSD をご覧ください。

この章の読者は、以下のことを理解しておく必要があります。

- package または ports を用いたサードパーティ製ソフトウェアのインストール方法 (アプリケーションのインストール - packages と ports)。
- X およびウィンドウマネージャのインストール方法 (X Window System)。

マルチメディア環境を整える方法については マルチメディア を参照してください。

この文書は英語で書かれている原文をそのまま邦訳したものです。
必ずしも各アプリケーションで日本語が扱えるとは限らないことに注意してください。
日本語に対応したアプリケーションは、Ports Collection の japanese ディレクトリにあるかもしれません。

6.2. ブラウザ

FreeBSD では Web ブラウザは事前にインストールされていません。 そのかわり、Ports Collection の www カテゴリには数多くの Web ブラウザ が用意されており、 多くのプログラムを packages からインストールしたり、 Ports Collection からコンパイルできます。

KDE や GNOME デスクトップ環境には、 それぞれ HTML ブラウザが用意されています。 これらのデスクトップ環境を設定するための情報については 「デスクトップ環境」 を参照してください。

軽量なブラウザには、 www/dillo2, www/links, および www/w3m といったものがあります。

この節では、広く使われている以下の web ブラウザのインストール方法について説明します。 もし、アプリケーションがリソースを大量に消費したり、 ports からのコンパイルに時間がかかったり、他の ports に大きく依存する場合には、そのことについても触れます。

アプリケーション名	必要なリソース	**port** からのインストール	備考
Firefox	中	重	FreeBSD, Linux® および地域化されたバージョンを利用できます。
Konqueror	中	重	KDE ライブラリを必要とします。
Chromium	中	重	Gtk+ を必要とします。

6.2.1. Firefox

Firefox は、 標準に準拠した HTML 表示エンジン、タブブラウジング、ポップアップブロック、拡張性、高い安全性などが特徴のオープンソースのブラウザです。 Firefox は Mozilla のコードベースから派生したブラウザです。

最新の Firefox の package をインストールするには以下のように入力してください。

```
# pkg install firefox
```

Firefox 延長サポート版 (ESR: Extended Support Release) を利用したい場合には、かわりに以下のように入力してください。

```
# pkg install firefox-esr
```

かわりにソースコードから希望の firefox をコンパイルすることもできます。 この例では www/firefox をビルドしますが、 `firefox` の部分は、 インストールする ESR やローカライズに置き換えることもできます。

```
# cd /usr/ports/www/firefox
# make install clean
```

6.2.2. Konqueror

Konqueror はブラウザであると同時に、ファイルマネージャおよびマルチメディアビューアの役割も果たします。 Konqueror は、KHTML とともに WebKit にも対応しています。WebKit は Chromium など最近のブラウザの多くで採用されているレンダリングエンジンです。

Konqueror は、 以下のように入力して package からインストールできます。

```
# pkg install konqueror
```

Ports Collection からインストールするには、 以下のように入力してください。

```
# cd /usr/ports/x11-fm/konqueror/
# make install clean
```

6.2.3. Chromium

Chromium は、 オープンソースのブラウザのプロジェクトで、 より安全かつより高速、より安定したウェブブラウジングを目指しています。 Chromium は、タブブラウジング、ポップアップブロック、拡張機能などの機能を持っています。 Chromium は、Google Chrome ウェブブラウザがベースとしているオープンソースのプロジェクトです。

Chromium は、 以下のように入力することで package からインストールできます。

```
# pkg install chromium
```

または、Ports Collection を用いて ソースから Chromium をコンパイルしてインストールできます。

```
# cd /usr/ports/www/chromium
# make install clean
```

Chromium の実行可能ファイルは、 /usr/local/bin/chrome です。/usr/local/bin/chromium ではありません。

6.3. 生産的なアプリケーション

生産的なアプリケーションということになると、 ユーザはしばしばオフィススイートや、使いやすい文書作成ソフトウェアを求めるでしょう。
デフォルトの生産的なアプリケーションはありませんが、 KDE のような デスクトップ環境はオフィススイートを提供しています。
インストールされているウィンドウマネージャにかかわらず、FreeBSD では、いくつものオフィススイート、 グラフィカルな文書作成ソフトウェアを利用できます。

この節では、 以下の人気のある生産的なソフトウェアのインストール方法について説明します。もし、アプリケーションがリソースを大量に消費したり、 ports からのコンパイルに時間がかかったり、もしくは他の ports に大きく依存する場合には、 そのことについても触れます。

アプリケーション名	必要なリソース	**port** からのインストール	実行に必要となる主な環境
Calligra	少	重	KDE
AbiWord	少	軽	Gtk+ または GNOME
Gimp	少	重	Gtk+
Apache OpenOffice	多	莫大	JDK™ および Mozilla
LibreOffice	やや多	莫大	Gtk+ または KDE/GNOME または JDK™

6.3.1. Calligra

KDE デスクトップには、 KDE 環境以外でも利用可能なオフィススイートがあります。 Calligra には、他のオフィススイートと同様に、 標準的なアプリケーションが含まれています。 Words は文書作成ソフトウェア、 Sheets は表計算ソフトウェア、 Stage はプレゼンテーションソフトウェア、そして Karbon は図形描画ソフトウェアです。

FreeBSD では package または port から editors/calligra をインストール出来ます。 package からインストールするには次のようにします。

```
# pkg install calligra
```

package を入手できない場合は、かわりに Ports Collection を利用してください。

```
# cd /usr/ports/editors/calligra
# make install clean
```

6.3.2. AbiWord

AbiWord は、Microsoft® Word のような見た目や操作感を持つフリーの文書作成ソフトウェアです。速く、多くの機能を持ち、ユーザフレンドリです。

AbiWord は、 Microsoft® .rtf のような独自仕様を含む多くの形式のファイルを読み書きできます。

AbiWord package をインストールするには、以下のようにしてください。

```
# pkg install abiword
```

package を入手できない場合は、 Ports Collection からコンパイルしてください。

```
# cd /usr/ports/editors/abiword
# make install clean
```

6.3.3. GIMP

画像を描画したり写真を修正することに関して、 GIMP は洗練された編集プログラムです。単純にお絵かきソフトウェアとして使うこともできますし、
高品質な写真の加工ツールとしても使えます。 多くのプラグインに対応しており、スクリプトインタフェースを特徴としています。 GIMP はさまざまな形式のファイルを読み書きでき、スキャナやタブレットとのインタフェースにも対応しています。

package をインストールするには、以下のようにしてください。

```
# pkg install gimp
```

もしくは、Ports Collection を利用してください。

```
# cd /usr/ports/graphics/gimp
# make install clean
```

Ports Collection の graphics カテゴリ (freebsd.org/ja/ports/) には、GIMP に関連したプラグイン、ヘルプファイルおよびユーザマニュアルなどがあります。

6.3.4. Apache OpenOffice

Apache OpenOffice は、Apache Software Foundation のインキュベータプロジェクトとして開発が行われているオープンソースのオフィススイートです。
Apache OpenOffice は、完全なオフィススイートに必須のアプリケーション (文書作成ソフトウェア、表計算ソフトウェア、
プレゼンテーションソフトウェア、そして図形描画ソフトウェア) をひととおり揃えています。
ユーザインタフェースは他のオフィススイートと似ており、
広く用いられているさまざまな形式のファイルを読み書きできます。
多くの言語で利用でき、インタフェース、スペルチェッカ、 辞書は国際化されています。

Apache OpenOffice の文書作成ソフトウェアは、ネイティブの XML ファイル形式を採用することでポータビリティや柔軟性を高めています。
表計算ソフトウェアにはマクロ機能があり、 外部データベースと接続することもできます。 Apache OpenOffice は、 Windows®, Solaris™, Linux®, FreeBSD および Mac OS® X において安定してネイティブに動作しています。 Apache OpenOffice についてのより詳しい情報は、openoffice.org をご覧ください。 また、porting.openoffice.org/freebsd/ から、FreeBSD 特有の情報を参照してください。

Apache OpenOffice package をインストールするには、以下のように入力してください。

```
# pkg install apache-openoffice
```

package をインストールしたら、以下のコマンドを入力して Apache OpenOffice を起動してください。

```
% openoffice-X.Y.Z
```

ここで *X.Y.Z* は、 インストールされている Apache OpenOffice のバージョン番号です。 Apache OpenOffice の初回起動時に、いくつかの質問が行われ、 ユーザのホームディレクトリに .openoffice.org フォルダが作成されます。

希望の Apache OpenOffice の packages を利用できない場合には、port を利用する方法もあります。しかしながら、コンパイルには大きなディスクスペースと、 本当にかなり長い時間を必要とします。

```
# cd /usr/ports/editors/openoffice-4
# make install clean
```

地域化されたバージョンをビルドするには、
上記のコマンドの代わりに以下を実行して下さい。

```
# make LOCALIZED_LANG=your_language install clean
```

your_language を正しい言語 ISO コードに置き換えてください。
サポートされている言語コードは、同じ port ディレクトリにある
files/Makefile.localized に書かれています。

6.3.5. LibreOffice

LibreOffice は、documentfoundation.org
が開発しているフリーソフトウェアのオフィススイートです。
他のメジャーなオフィススイートと互換性があり、 さまざまなプラットフォームで利用できます。
Apache OpenOffice.org からの新しいフォークで、 完全なオフィススイートに必須のアプリケーション
(文書作成ソフトウェア、表計算ソフトウェア、
プレゼンテーションソフトウェア、図形描画ソフトウェア、
データベース管理ソフトウェア、数式エディタ) をすべて揃えています。 多くの言語で利用でき、
インタフェース、スペルチェッカ、辞書は国際化されています。

LibreOffice の文書作成ソフトウェアは、 ネイティブのファイル形式に XML を採用することで
ポータビリティや柔軟性を高めています。 表計算ソフトウェアにはマクロ機能があり、
外部データベースと接続することもできます。 LibreOffice は、 Windows®, Solaris™, Linux®,
FreeBSD, Mac OS® X において安定してネイティブに動作しています。 LibreOffice
についての詳しい情報は、libreoffice.org をご覧ください。

英語版の LibreOffice package をインストールするには、以下のように入力してください。

```
# pkg install libreoffice
```

Ports Collection の edtors カテゴリ (freebsd.org/ja/ports/) カテゴリには、地域化された LibreOffice
が用意されています。 地域化された package をインストールするには、 `libreoffice` を地域化された
package 名に置き換えてください。

package をインストールしたら、以下のコマンドで LibreOffice を起動してください。

```
% libreoffice
```

初回起動時には、いくつかの質問が行われ、 ユーザのホームディレクトリに .libreoffice
フォルダが作成されます。

希望の LibreOffice の packages を利用できない場合には、port からコンパイルする方法もあります。
しかしながら、コンパイルには大きなディスクスペースと、 本当にかなり長い時間を必要とします。
以下の例では、英語版をコンパイルします。

```
# cd /usr/ports/editors/libreoffice
```

```
# make install clean
```

地域化されたバージョンをビルドしたいのなら、 希望の言語の port ディレクトリに cd コマンドで移動してください。 対応している言語は、Ports Collection の editors カテゴリ (freebsd.org/ja/ports/) にあります。

6.4. ドキュメントビューア

UNIX® の出現以降、 いくつかの新しい文書形式が広く使われるようになりました。
基本システムには、それらの文書が要求するビューアがないかもしれません。
この節ではそれらのドキュメントビューアのインストール方法について説明します。

アプリケーション名	必要なリソース	**port** からのインストール	実行に必要になる主な環境
Xpdf	少	軽	FreeType
gv	少	軽	Xaw3d
Geeqie	少	軽	Gtk+ または GNOME
ePDFView	少	軽	Gtk+ または GNOME
Okular	少	重	KDE

6.4.1. Xpdf

FreeBSD 向けの軽い PDF ビューアを使いたいのなら Xpdf を試してみてください。
これは少ないリソースで動作するビューアで、軽くて効率的です。 標準の X フォントを利用し、 他の X ツールキットを必要としません。

Xpdf の package をインストールするには次のコマンドを入力してください。

```
# pkg install xpdf
```

package を入手できない場合は、 Ports Collection を利用してください。

```
# cd /usr/ports/graphics/xpdf
# make install clean
```

インストールが完了したら xpdf を起動してください。
メニューを表示するにはマウスの右ボタンを押してください。

6.4.2. gv

gv は PostScript® と PDF のビューアです。これは ghostview をベースとしていますが、 Xaw3d ウィジットツールキットによってより良い外観になっています。 gv は向きや用紙のサイズ、拡大縮小、アンチエイリアスなどたくさんの設定可能な機能を持っています。
ほとんどすべての操作をキーボードかマウスのどちらかだけで行なうことができます。

package から gv をインストールするには次のようにします。

```
# pkg install gv
```

package を利用できない場合には、Ports Collection を使ってください。

```
# cd /usr/ports/print/gv
# make install clean
```

6.4.3. Geeqie

Geeqie は、メンテナンスが行われていない GQView プロジェクトからのフォークで、開発を進めることと、
これまでに作成されたパッチを統合することを目指しています。 Geeqie は、クリックひとつで画像ファイルを開いたり、外部エディタを起動したり、
サムネイル画像を作成できるような画像管理ソフトウェアです。
また、スライドショーや基本的なファイル操作機能も備えており、 画像のコレクションの管理や、重複したファイルを見つけることが簡単にできます。 Geeqie は全画面表示、および国際化にも対応しています。

Geeqie package をインストールするには次のコマンドを入力してください。

```
# pkg install geeqie
```

package を入手できない場合は、 Ports Collection を利用してください。

```
# cd /usr/ports/graphics/geeqie
# make install clean
```

6.4.4. ePDFView

ePDFView は軽量な PDF ドキュメントビューアです。 このビューアは、 Gtk+ および Poppler ライブラリのみを使います。 このソフトウェアは、現在開発中ですが、ほぼすべての PDF ファイル (暗号化されたものを含む) を開くことが可能で、ドキュメントのコピーを保存でき、 CUPS を用いた印刷にも対応しています。

package から ePDFView をインストールするには以下のようにしてください。

```
# pkg install epdfview
```

package が利用できないようでしたら、 Ports Collection を使ってインストールしてください。

```
# cd /usr/ports/graphics/epdfview
```

```
# make install clean
```

6.4.5. Okular

Okular は、 KDE の KPDF をベースとした一般的なドキュメントビューアです。 このビューアは、PDF, PostScript®, DjVu, CHM, XPS, および ePub といった、多くの形式のファイルを開くことができます。

package で Okular をインストールするには、以下のようにしてください。

```
# pkg install okular
```

package が利用できないようでしたら、 Ports Collection を使ってインストールしてください。

```
# cd /usr/ports/graphics/okular
# make install clean
```

6.5. 財務管理ソフトウェア

FreeBSD のデスクトップで個人的な財務管理ができるように、強力で簡単に使えるアプリケーションが用意されています。 それらのアプリケーションの中には Quicken や Excel などの広く行き渡った形式のファイルと互換性があるものもあります。

この節では次のアプリケーションについて説明します。

アプリケーション名	必要なリソース	**port** からのインストール	実行に必要になる主な環境
GnuCash	少	重	GNOME
Gnumeric	少	重	GNOME
KMyMoney	少	重	KDE

6.5.1. GnuCash

GnuCash は、 GNOME の一部で、使いやすくかつ強力なアプリケーションとしてエンドユーザに提供されています。 GnuCash を使えば、収入や支出、銀行口座、あるいは株を管理できます。 直観的なインタフェースを特徴としていますが、高度な機能も提供しています。

GnuCash は洗練された登録機能、 階層構造の勘定システム、多くのキーボードショートカット、自動補完機能を提供しています。 単一のトランザクションをより小さな要素に分解できます。 GnuCash は、 Quicken の QIF ファイルの読み込みやマージができます。また、国際的な日付および通貨形式も扱えます。

GnuCash package をインストールするには次のようにしてください。

```
# pkg install gnucash
```

package が手に入らなければ、Ports Collection を使ってください。

```
# cd /usr/ports/finance/gnucash
# make install clean
```

6.5.2. Gnumeric

Gnumeric は、GNOME コミュニティによって開発されている表計算ソフトウェアです。セルの書式に従ってユーザの入力を自動的に推測する便利な機能や、多くのシーケンスに対する自動補完機能があります。Excel, Lotus 1-2-3, Quattro Pro といった広く行き渡っている多くの形式のファイルを読みこめます。多くの関数を内蔵しており、数値、通貨、日付、時間などのよく使うセルの書式が利用できます。

Gnumeric package をインストールするには次のように入力してください。

```
# pkg install gnumeric
```

package が手に入らなければ、Ports Collection を使ってください。

```
# cd /usr/ports/math/gnumeric
# make install clean
```

6.5.3. KMyMoney

KMyMoney は、KDE コミュニティが作成している個人用財務管理アプリケーションです。KMyMoney は、商用の個人用財務管理ソフトウェアに見られる重要な機能を提供することを目指しています。また、使いやすい複式簿記機能も特徴です。KMyMoney は標準の Quicken QIF ファイルをインポート可能で、投資履歴や複数通貨の取扱い、財政状況のレポートを提供します。

package から KMyMoney をインストールするには次のようにします。

```
# pkg install kmymoney-kde4
```

package が手に入らない場合は、Ports Collection を使ってください。

```
# cd /usr/ports/finance/kmymoney-kde4
# make install clean
```

Chapter 7. マルチメディア

7.1. この章では

FreeBSD は数多くの種類のサウンドカードに対応しており、FreeBSD システムで原音に忠実な出力を楽しむことができます。 これには録音機能と、MPEG Audio Layer 3 (MP3) や Waveform Audio File (WAV), Ogg Vorbis などをはじめとした多くの形式の音楽の再生機能が含まれます。 加えて FreeBSD の Ports Collection には、録音した音楽を編集したり、音響効果を加えたり、接続された MIDI 機器を制御するためのアプリケーションが用意されています。

FreeBSD ではビデオファイルおよび DVD の再生もできます。 FreeBSD の Ports Collection には、さまざまなビデオメディアをエンコード、 変換、再生するアプリケーションが用意されています。

この章では FreeBSD 上でサウンドカード、ビデオの再生、TV チューナカード、スキャナを設定する方法について説明します。
また、これらのデバイスを使うためのアプリケーションについても説明します。

この章を読むと、以下のことがわかります。

- FreeBSD でのサウンドカードの設定方法
- サウンドの設定に関するトラブルシューティング
- MP3 およびその他の形式の音声を再生、エンコードする方法
- FreeBSD システムでのビデオ再生の準備
- DVD, .mpg および .avi ファイルを再生する方法
- CD および DVD の情報をファイルに抽出する方法
- TV カードの設定方法
- MythTV を FreeBSD にインストールして設定する方法
- 画像スキャナの設定方法
- Bluetooth ヘッドホンの設定方法

この章を読む前に、以下のことを理解しておく必要があります。

- アプリケーションのインストール方法 (アプリケーションのインストール - packages と ports)

7.2. サウンドカードの設定

設定をはじめる前に、サウンドカードのモデル、 そのカードが使用しているチップを確認してください。 FreeBSD は サウンドカードに幅広く対応しています。 使用しているカードが対応しているかどうか、どの FreeBSD ドライバを使うかについて、ハードウェアノート の対応オーディオデバイスの一覧を確認してください。

サウンドデバイスを使うためには、デバイスドライバを読み込まなければいけません。 もっとも簡単な方法は kldload(8) を使ってサウンドカードのカーネルモジュールを読み込むことです。 次の例は、Intel 仕様のビルトインオーディオチップセットのドライバを読み込む例です。

```
# kldload snd_hda
```

このドライバを起動時に読み込むように設定するためには、 /boot/loader.conf にドライバを追加してください。 このドライバの場合は以下の行になります。

```
snd_hda_load="YES"
```

他に利用可能な読み込み可能なサウンドモジュールは /boot/defaults/loader.conf に記載されています。 どのドライバを利用すればいいか確かでなければ、 snd_driver モジュールを読み込んでください。

```
# kldload snd_driver
```

snd_driver モジュールは、 一般に使用されるカードに対応したドライバをまとめて一度に読み込むメタドライバです。 このドライバを使用すれば、速やかに正しいドライバを探し出すことができます。 /boot/loader.conf ファイルを使用して、 すべてのサウンドドライバを読み込むこともできます。

snd_driver メタドライバの読み込み後に、 どのドライバがサウンドカードに選択されたのかを知るには、 `cat /dev/sndstat` と入力してください。

7.2.1. サウンドに対応したカスタムカーネルを設定する

この節は、 サウンドカードのドライバをカーネルへ静的に組み込もうと考えているユーザ向けです。 カーネル再構築の詳細は FreeBSD カーネルのコンフィグレーション を参照してください。

サウンドに対応したカスタムカーネルを使うときには、 オーディオフレームワークドライバをカーネルコンフィグレーションファイルに追加してください。

```
device sound
```

次に、サウンドカードに対応したドライバを追加します。 前節の Intel 仕様のビルトインオーディオチップセットの例では、 カスタムカーネルコンフィグレーションファイルに以下の行を追加してください。

```
device snd_hda
```

ドライバのマニュアルページを読んで、 ドライバが使用するデバイス名を調べてください。

PnP 非対応の ISA サウンドカードでは、 IRQ および I/O ポートの設定を /boot/device.hints に指定する必要があるかもしれません。 システムの起動時に、loader(8) はこのファイルを読み、設定情報をカーネルに渡します。 たとえば、PnP 非対応の古い Creative SoundBlaster® 16 (ISA 接続) には `snd_sb16` とともに snd_sbc(4) ドライバを使用します。 このカードを使用する場合には、 カーネルコンフィグレーションファイルに以下の行を追加してください。

```
device snd_sbc
device snd_sb16
```

もしカードが 0x220 I/O port と IRQ 5 を使用している場合には、/boot/device.hints に以下の行を追加してください。

```
hint.sbc.0.at="isa"
hint.sbc.0.port="0x220"
hint.sbc.0.irq="5"
hint.sbc.0.drq="1"
hint.sbc.0.flags="0x15"
```

/boot/device.hints に用いるべき構文は、sound(4) および、サウンドカードの各ドライバのマニュアルページに記載されています。

これまでの設定はデフォルトのものです。 カードを使用する状況によっては、IRQ やその他の設定を変更する必要があるかもしれません。 このカードについての詳細は、snd_sbc(4) をご覧ください。

7.2.2. サウンドのテスト

必要となるモジュールを読み込むか、カスタムカーネルで再起動すると、
サウンドカードが検出されます。 確認をするには、dmesg | grep pcm と実行してください。
この例は、ビルトイン Conexant CX20590 チップセットを搭載したシステムのものです。

```
pcm0: <NVIDIA (0x001c) (HDMI/DP 8ch)> at nid 5 on hdaa0
pcm1: <NVIDIA (0x001c) (HDMI/DP 8ch)> at nid 6 on hdaa0
pcm2: <Conexant CX20590 (Analog 2.0+HP/2.0)> at nid 31,25 and 35,27 on hdaa1
```

サウンドカードの状態は、 以下のコマンドを使用して確認することもできます。

```
# cat /dev/sndstat
FreeBSD Audio Driver (newpcm: 64bit 2009061500/amd64)
Installed devices:
pcm0: <NVIDIA (0x001c) (HDMI/DP 8ch)> (play)
pcm1: <NVIDIA (0x001c) (HDMI/DP 8ch)> (play)
pcm2: <Conexant CX20590 (Analog 2.0+HP/2.0)> (play/rec) default
```

この出力は、サウンドカードによって異なります。 pcm デバイスがなければ、
適切なデバイスドライバが読み込まれているか、
カーネルに追加されてコンパイルされているかどうかを確認してください。
次の節では、良くある問題とその解決方法をリストアップしています。

すべてうまくいけば、サウンドカードが FreeBSD で機能するでしょう。 CD または DVD ドライブのオーディオ出力端子がサウンドカードと適切に接続されていれば、cdcontrol(1) を使ってドライブ内のオーディオ CD を再生できます。

```
% cdcontrol -f /dev/acd0 play 1
```

オーディオ CD は特別なエンコーディングが行われているため、 mount(8) を使ってマウントすべきではありません。

audio/workman のように、よりよいインタフェースを提供するさまざまなアプリケーションがあります。 audio/mpg123 port をインストールして MP3 オーディオファイルを聞くことができます。

手っ取り早くカードをテストするには、 /dev/dsp デバイスにデータを送ってみてください。

```
% cat filename > /dev/dsp
```

ここで filename は、どのような形式のファイルでも構いません。
このコマンドラインを実行すると雑音が発生するはずです。
これにより、サウンドカードが動作していることを確認できます。

/dev/dsp* デバイスノードは、必要に応じて自動的に作成されます。 デバイスノードが使用されていない場合には存在せず、 ls(1) の出力に表示されません。

7.2.3. Bluetooth サウンドデバイスの設定

Bluetooth デバイスへの接続についての説明は、この章の範囲外です。 詳細については Bluetooth をご覧ください。

FreeBSD のサウンドシステムで Bluetooth サウンドシンクを動かすには、最初に audio/virtual_oss をインストールしてください。

```
# pkg install virtual_oss
```

audio/virtual_oss を使うには、 カーネルに cuse が読み込まれている必要があります。

```
# kldload cuse
```

システムのスタートアップ時に cuse を読み込むには、以下のコマンドを実行してください。

```
# echo 'cuse_load=yes' >> /boot/loader.conf
```

audio/virtual_oss でヘッドホンをサウンドシンクとして使うには、 Blueooth オーディオデバイスに接続後、 仮想デバイスを作成する必要があります。

```
# virtual_oss -C 2 -c 2 -r 48000 -b 16 -s 768 -R /dev/null -P
/dev/bluetooth/headphones -d dsp
```

この例において、 *headphones* は、 /etc/bluetooth/hosts に記載されているホスト名です。 代わりに BT_ADDR を使えます。

詳細については、virtual_oss(8) をご覧ください。

7.2.4. サウンドカードの問題についてのトラブルシューティング

良くあるエラーメッセージ は、 良くあるエラーメッセージとその解決法の一覧です。

表 *8.* 良くあるエラーメッセージ

エラー	解決方法
`sb_dspwr(XX) timed out`	使用する I/O ポートが適切に設定されていません。
`bad irq XX`	使用する IRQ が正しく設定されていません。 サウンドカードの IRQ と設定した IRQ が同じかどうか確かめてください。
`xxx: gus pcm not attached, out of memory`	デバイスを使用するのに十分なメモリを確保できません。
`xxx: can't open /dev/dsp!`	`fstat | grep dsp` と入力して、 他のアプリケーションがデバイスを使用しているか調べてください。 注目すべきトラブルメーカは esound と KDE のサウンド機能です。

最近のグラフィックカードの中には、 HDMI を利用するため、 グラフィックカード自身がサウンドカードを持つものがあります。 このようなサウンドデバイスには、 時としてサウンドカードより若い番号が付けられることがあります。 そのような場合には、 サウンドカードをデフォルトプレイバックデバイスとして利用できません。
このことが原因かどうかを確認するには、dmesg を実行して pcm を探してください。 以下のような出力を得るかもしれません。

```
...
hdac0: HDA Driver Revision: 20100226_0142
hdac1: HDA Driver Revision: 20100226_0142
hdac0: HDA Codec #0: NVidia (Unknown)
hdac0: HDA Codec #1: NVidia (Unknown)
hdac0: HDA Codec #2: NVidia (Unknown)
hdac0: HDA Codec #3: NVidia (Unknown)
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
...
```

この例では、グラフィックカード (NVidia) には、サウンドカード (Realtek ALC889) より若い番号が付けられています。
サウンドカードをデフォルトのプレイバックデバイスとして利用するには、 hw.snd.default_unit をプレイバックで使用するユニット番号に変更してください。

```
# sysctl hw.snd.default_unit=n
```

ここで、n は使用するサウンドデバイスの番号です。 この例では 4 です。 /etc/sysctl.conf に以下の行を入れると、 設定の変更が常に反映されるようになります。

```
hw.snd.default_unit=4
```

7.2.5. 複数音源の利用

同時に再生することのできる音源を複数実装していることは、 多くの場合望ましいことです。 FreeBSD では、"仮想サウンドチャネル" を使ってカーネル内でサウンドを合成することにより、サウンドカードの再生を多重化することができます。

仮想チャネルの数を決めるのに三つの sysctl(8) 変数を設定できます。

```
# sysctl dev.pcm.0.play.vchans=4
# sysctl dev.pcm.0.rec.vchans=4
# sysctl hw.snd.maxautovchans=4
```

この例では四つの仮想チャネルを設定しています。 これは通常利用する上で十分実用的な数です。 dev.pcm.0.play.vchans=4 と dev.pcm.0.rec.vchans=4 は、デバイスが取り付けられた後で設定できます。 これらは pcm0 が再生や録音のために持っている仮想チャネルの数です。 hw.snd.maxautovchans は、 kldload(8) を用いて認識された新しいデバイスの仮想チャネル数です。 pcm モジュールはハードウェアドライバとは独立して読み込むことができるので、 hw.snd.maxautovchans は、オーディオデバイスが取り付けられた時に、デバイスに与えられる仮想チャネルの数を表しています。 より詳細な情報については pcm(4) を参照してください。

デバイスを使用しているときに仮想チャンネルの数を変更することはできません。
まず、ミュージックプレーヤやサウンドデーモンといった
デバイスを使用しているすべてのプログラムを終了してください。

/dev/dsp0 を必要とするプログラムが意識しなくても、 適切な pcm デバイスが自動的に設定されます。

7.2.6. ミキサチャネルの初期値を設定する

各ミキサチャネルの初期値は pcm(4) ドライバのソースコードにハードコーディングされています。 mixer(8) および他のサードパーティ製のアプリケーションやデーモンによって、サウンドカードのミキサレベルを変更できますが、 永続的な解決方法ではありません。 そのかわり以下の例のように、 適切な値を /boot/device.hints ファイルに記述することによって、ドライバレベルでミキサの初期値を設定することができます。

```
hint.pcm.0.vol="50"
```

この例では、pcm(4) が読み込まれたと同時に、 ボリュームチャネルの初期値を 50 に設定します。

7.3. MP3 オーディオ

この節では、FreeBSD で利用できる MP3 プレイヤや、オーディオ CD トラックを吸い出す方法、 および MP3 のエンコード、 デコードの方法について説明します。

7.3.1. MP3 プレイヤ

Audacious は 人気のあるグラフィカルな MP3 プレイヤです。 Winamp スキンや追加のプラグインに対応しています。 Audacious のプレイリスト、 グラフィックイコライザ等のインタフェースは直感的です。 Winamp を使いなれている人は簡単に Audacious を使えるでしょう。 FreeBSD では、Audacious は multimedia/audacious の port または package からインストールできます。 Audacious は、XMMS の子孫です。

audio/mpg123 package もしくは port は、 は代替となる コマンドライン上の MP3 プレイヤです。インストールしたら、再生する MP3 ファイルをコマンドラインから指定してください。 もしシステムが、複数のオーディオデバイスを搭載しているのであれば、 サウンドデバイスを同様に指定してください。

```
# mpg123 -a /dev/dsp1.0 Foobar-GreatesHits.mp3
High Performance MPEG 1.0/2.0/2.5 Audio Player for Layers 1, 2 and 3
        version 1.18.1; written and copyright by Michael Hipp and others
        free software (LGPL) without any warranty but with best wishes

Playing MPEG stream from Foobar-GreatestHits.mp3 ...
MPEG 1.0 layer III, 128 kbit/s, 44100 Hz joint-stereo
```

他の MP3 プレイヤも Ports Collection から利用できます。

7.3.2. CD オーディオトラックの抽出

CD 全体または CD トラックを MP3 に変換する前に、CD 上のオーディオデータをハードディスク上に抽出する必要があります。 これは raw CD Digital Audio (CDDA) データを WAV ファイルにコピーすることで行われます。

sysutils/cdrtools スイートからインストールされる cdda2wav ツールを用いて、CD からオーディオデータを抽出できます。

CD をドライブにいれて次のコマンドを root 権限で実行すると、 CD 全体をトラックごとに個々の WAV ファイルに抽出できます。

```
# cdda2wav -D 0,1,0 -B
```

この例では、-D 0,1,0 は SCSI デバイス 0,1,0 が抽出する CD を表します。 cdrecord -scanbus を使って、

システムの適切なデバイスパラメータを取得してください。

個々のトラックを抽出するには、 次のように -t でトラックを指定してください。

```
# cdda2wav -D 0,1,0 -t 7
```

範囲を指定して、 一番目から七番目のトラックまで抽出したい場合、 次のようにします。

```
# cdda2wav -D 0,1,0 -t 1+7
```

ATAPI (IDE) CDROM ドライブから抽出するには、 SCSI ユニット番号をデバイス名に置き換えて指定します。 たとえば IDE ドライブから七番目のトラックを抽出するには、 次のようにします。

```
# cdda2wav -D /dev/acd0 -t 7
```

または、「オーディオ CD の複製」 で説明されているように、dd を使って ATAPI ドライブ上のオーディオトラックを展開できます。

7.3.3. MP3 のエンコードとデコード

lame は、 ポピュラーな MP3 エンコーダです。 audio/lame port からインストールできます。 特許の問題から、package は利用できません。

次のコマンドを実行すると、抽出した WAV ファイル audio01.wav を使って audio01.mp3 に変換します。

```
# lame -h -b 128 --tt "曲名" --ta "アーティスト名" --tl "アルバム名" \
--ty "年" --tc "コメント" --tg "ジャンル" audio01.wav audio01.mp3
```

ここで指定している 128 kbits は、MP3 の標準のビットレートです。 160 kbits または 192 kbits のビットレートは、 さらに高音質を提供します。 ビットレートが高くなるにつれて作成される MP3 ファイルは多くのディスク領域を消費します。 -h オプションを指定すると "低速高品質" モードとなります。 --t ではじまるオプションは ID3 タグを設定します。 このタグにはたいてい曲の情報が含まれており、 MP3 ファイルに格納されます。 Lame のマニュアルを参照すれば、 他のエンコーディングのオプションが見つかるでしょう。

MP3 からオーディオ CD を作成するには、 まず非圧縮のファイル形式に変換しなければなりません。 XMMS は WAV 形式へ変換できますが、 mpg123 は raw Pulse-Code Modulation (PCM) オーディオデータに変換します。

mpg123 を使って audio01.mp3 を変換するには、PCM ファイルを指定してください。

```
# mpg123 -s audio01.mp3 > audio01.pcm
```

XMMS を使って MP3 を WAV 形式に変換するには、 以下の手順に従ってください。

Procedure: XMMS を使って WAV 形式に変換する

1. XMMS を起動します。
2. 右クリックで XMMS メニューを表示します。
3. `Options` から `Preferences` を選択します。
4. Output Plugin を "Disk Writer Plugin" に変更します。
5. `Configure` を押します。
6. 非圧縮ファイルを書き出すディレクトリを入力、 または選択します。
7. 普段通り XMMS へ MP3 ファイルを読み込みます。 音量は 100% でイコライザの設定はオフにします。
8. `Play` を押します。 XMMS は MP3 を再生しているかのように表示しますが、音声はきこえません。 実際には MP3 をファイルに出力しています。
9. 終了したら、再び MP3 を聴けるように Output Plugin を以前のように元に戻すのを忘れないでください。

WAV と PCM 形式は、 cdrecord で利用できます。 WAV ファイルを使用する場合、それぞれのトラックの先頭に小さなノイズが入るのに気づくでしょう。 これは WAV ファイルのヘッダ情報です。 audio/sox port または package を使うとヘッダ情報を削除できます。

```
% sox -t wav -r 44100 -s -w -c 2 track.wav track.raw
```

FreeBSD での CD 作成の詳しい情報は 「光メディア (CD & DVD) の作成と使用」 を参照してください。

7.4. ビデオ再生

ビデオ再生のための設定をはじめる前に、
ビデオカードのモデルおよびチップセットを確認する必要があります。 Xorg はさまざまなビデオカードに対応していますが、
すべてのカードがビデオ再生に性能を発揮できるとは限りません。 利用しているビデオカードの Xorg サーバが対応している拡張機能のリストを得るには、 Xorg を実行中に `xdpyinfo` を実行してください。

さまざまなプレイヤやオプションを試すのに、 テストファイルとして小さな MPEG ファイルを用意しておくのはよい考えです。 いくつかの DVD アプリケーションは DVD メディアを /dev/dvd として初期設定しているか、ハードコーディングしているので、次のように適切なデバイスにシンボリックリンクを張っておくと便利かもしれません。

```
# ln -sf /dev/cd0 /dev/dvd
```

devfs(5) の仕様により、
このように手動で作成されたリンクはシステムを再起動すると消えてしまいます。
システムの起動時にこれらのシンボリックリンクを自動的に作成するには、 /etc/devfs.conf に下記の設定を追加してください。

```
link cd0 dvd
```

特別な機能を必要とする DVD の抽出には、 DVD デバイスへの書き込み権限が必要です。

Xorg インタフェースの使う共有メモリを拡張するために、 以下の sysctl(8) 変数の値を増やすことが推奨されています。

```
kern.ipc.shmmax=67108864
kern.ipc.shmall=32768
```

7.4.1. ビデオ機能の決定

Xorg においてビデオ表示性能を改善する方法はいくつかあり、正しく動作するかどうかはハードウェアに大きく依存しています。 下記に説明したどの方法でも、ハードウェアが変わると品質が変わるでしょう。

よく知られたビデオインタフェースは次の通りです。

1. Xorg: 共有メモリを用いた通常の出力
2. XVideo: 特別なアクセラレータによって、 drawable オブジェクトに直接ビデオを表示する Xorg インタフェースの拡張機能です。
   この拡張を使うことで廉価なコンピュータでも高品質の再生が可能になります。 次の節では、この拡張が動作していることの確認方法について説明します。
3. SDL: Simple Directmedia Layer は、さまざまなオペレーティングシステムの間でサウンドとグラフィックスを効果的に利用したクロスプラットホームアプリケーションを開発することを目的としたレイヤです。 SDL はハードウェアに対する低レベルの抽象的概念を提供し、 時には Xorg インタフェースを使用するよりも効果的なことがあります。 FreeBSD では、SDL は、 devel/sdl20 package または port によりインストールできます。
4. DGA: Direct Graphics Access は、 プログラムが Xorg サーバを介せず直接フレームバッファを変更することを可能にする Xorg の拡張機能です。
   低レベルのメモリマッピングが実行できることを期待しているので、 この機能を使うプログラムは `root` 権限で実行されなければなりません。 DGA 機能拡張は dga(1) によってテストとベンチマークができます。 `dga`
   実行中はキーボードを押せばいつでもディスプレイ色が変更されます。 中止するには `q` を押します。
5. SVGAlib: 低レベルコンソールグラフィックレイヤ

XVideo

この拡張機能が動作しているかどうかを調べるには、 `xvinfo` を使います。

```
% xvinfo
```

以下のような結果が得られたならば、カードは XVideo に対応しています。

```
X-Video Extension version 2.2
  screen #0
  Adaptor #0: "Savage Streams Engine"
    number of ports: 1
    port base: 43
    operations supported: PutImage
    supported visuals:
      depth 16, visualID 0x22
      depth 16, visualID 0x23
    number of attributes: 5
      "XV_COLORKEY" (range 0 to 16777215)
              client settable attribute
              client gettable attribute (current value is 2110)
      "XV_BRIGHTNESS" (range -128 to 127)
              client settable attribute
              client gettable attribute (current value is 0)
      "XV_CONTRAST" (range 0 to 255)
              client settable attribute
              client gettable attribute (current value is 128)
      "XV_SATURATION" (range 0 to 255)
              client settable attribute
              client gettable attribute (current value is 128)
      "XV_HUE" (range -180 to 180)
              client settable attribute
              client gettable attribute (current value is 0)
    maximum XvImage size: 1024 x 1024
    Number of image formats: 7
      id: 0x32595559 (YUY2)
        guid: 59555932-0000-0010-8000-00aa00389b71
        bits per pixel: 16
        number of planes: 1
        type: YUV (packed)
      id: 0x32315659 (YV12)
        guid: 59563132-0000-0010-8000-00aa00389b71
        bits per pixel: 12
        number of planes: 3
        type: YUV (planar)
      id: 0x30323449 (I420)
        guid: 49343230-0000-0010-8000-00aa00389b71
        bits per pixel: 12
        number of planes: 3
        type: YUV (planar)
      id: 0x36315652 (RV16)
        guid: 52563135-0000-0000-0000-000000000000
        bits per pixel: 16
        number of planes: 1
        type: RGB (packed)
        depth: 0
        red, green, blue masks: 0x1f, 0x3e0, 0x7c00
      id: 0x35315652 (RV15)
```

```
        guid: 52563136-0000-0000-0000-000000000000
        bits per pixel: 16
        number of planes: 1
        type: RGB (packed)
        depth: 0
        red, green, blue masks: 0x1f, 0x7e0, 0xf800
      id: 0x31313259 (Y211)
        guid: 59323131-0000-0010-8000-00aa00389b71
        bits per pixel: 6
        number of planes: 3
        type: YUV (packed)
      id: 0x0
        guid: 00000000-0000-0000-0000-000000000000
        bits per pixel: 0
        number of planes: 0
        type: RGB (packed)
        depth: 1
        red, green, blue masks: 0x0, 0x0, 0x0
```

リストにある形式、YUV2, YUV12 などが XVideo のすべての実装で存在するとは限りません。
対応している形式が少ないために、
あるプレイヤでは悪影響が出るかもしれないことにも注意してください。

出力が以下のような場合、

```
X-Video Extension version 2.2
screen #0
no adaptors present
```

カードはおそらく XVideo に対応していないのでしょう。
このことはディスプレイでビデオを表示するのに、 ビデオカードおよびプロセッサによっては、
計算上の要求を満たすことがより困難になることを意味します。

7.4.2. ビデオを扱う **ports** および **packages**

この節では Ports Collection で利用可能な、
ビデオの再生に使用できるソフトウェアについて紹介します。

MPlayer および **MEncoder**

MPlayer はコマンドラインのビデオプレイヤで、
高速性と柔軟性をもたらすグラフィカルなインタフェースも持っています。 MPlayer
の他のグラフィカルなフロントエンドも Ports Collection からインストールできます。

MPlayer は multimedia/mplayer package または port からインストールできます。
いくつかのコンパイル時のオプションを設定することができ、また、
構築の際にさまざまなハードウェアのチェックがおこなわれます。 そのため、package
からインストールを行わず、 port から構築することを好むユーザもいます。

port を構築する際に、メニューのオプションは、port

にコンパイル時にオプションとしてどの形式に対応するかを決定するため、 見ておく必要があります。オプションが選択されていなければ、 MPlayer はその形式のビデオ形式を表示することは出来ません。矢印キーとスペースキーを使って必要な形式を選択してください。 選択が終わったら、`Enter` を押して、port の構築とインストールを続けてください。

デフォルトでは、この packege または port は、 `mplayer` コマンドラインユーティリティと `gmplayer` グラフィカルユーティリティを構築します。 ビデオをエンコードする必要があれば、multimedia/mencoder port をコンパイルしてください。 ライセンスの制限のため、 MEncoder の package は利用できません。

MPlayer を初めて起動すると、 各自のホームディレクトリ内に ~/.mplayer が作成されます。このサブディレクトリには、
ユーザ固有の設定ファイルのデフォルトバージョンが含まれています。

この節では、一般的な使用法についてのみ説明します。 数多くのオプションの完全な説明については、mplayer(1) のマニュアルに記載されています。

testfile.avi というファイルを再生するには、以下の例のように、 `-vo` とともに、ビデオインタフェースを指定してください。

```
% mplayer -vo xv testfile.avi
```

```
% mplayer -vo sdl testfile.avi
```

```
% mplayer -vo x11 testfile.avi
```

```
# mplayer -vo dga testfile.avi
```

```
# mplayer -vo 'sdl:dga' testfile.avi
```

ビデオ再生の相対的性能は多くの要因に依存し、 ハードウェアに応じて著しく変わると思われるので、これらのオプションをすべて試してみる価値はあるでしょう。

DVD を再生するには、 testfile.avi を `dvd://N -dvd-device DEVICE` に置き換えてください。 <N> には再生するタイトル番号を、 *DEVICE* は DVD のデバイスノードを指定します。 たとえば、/dev/dvd から 2 番目のタイトルを再生するには以下のようにします。

```
# mplayer -vo xv dvd://3 -dvd-device /dev/dvd
```

デフォルトの DVD デバイスは、 MPlayer port の構築時に `WITH_DVD_DEVICE=/path/to/desired/device` を追加することでで定義できます。デフォルトでは、デバイスは /dev/cd0 です。 詳細はこの port の Makefile.options をご覧ください。

停止、休止、 再生などをするにはキーバインディングを使ってください。
キーバインディングの一覧を見るには、mplayer -h を実行するか、もしくは、mplayer(1)
を読んでください。

再生に関する追加のオプションがあります。 全画面モードにする -fs -zoom オプションと、
性能を向上させる -framedrop オプションです。

よく使用するオプションについては、各ユーザの .mplayer/config に以下のように追加してください。

```
vo=xv
fs=yes
zoom=yes
```

mplayer を使って、 DVD タイトルを .vob に抽出できます。 DVD から 2
番目のタイトルをダンプするには次のようにします。

```
# mplayer -dumpstream -dumpfile out.vob dvd://2 -dvd-device /dev/dvd
```

出力された out.vob ファイルは MPEG 形式です。

UNIX® ビデオについて、 高レベルのノウハウを得たいと考えている方は mplayerhq.hu/DOCS
をご覧ください。技術的な情報があります。 このドキュメントは、
バグを報告する前に、読むべきものです。

mencoder を使う前に、mplayerhq.hu/DOCS/HTML/en/mencoder.html
を読んでオプションに慣れておくのはよい考えです。
品質向上、低ビットレート、形式変換をする方法が無数にあります。
これらの要素の調節具合で、性能が良かったり悪かったりするなど、 結果に違いが出るかもしれません。
コマンドラインオプションを不適切に組合せると、 mplayer
でさえ再生できない出力ファイルを作成してしまいます。

```
はじめは単純なファイルのコピーです。
```

```
% mencoder input.avi -oac copy -ovc copy -o output.avi
```

したがって、単にファイルを抽出したいときには、 mplayer に -dumpfile をつけます。

input.avi を音声に MPEG3 エンコードを使用して MPEG4 コーデックに変換するには、まず最初に
audio/lame port をインストールしてください。 ライセンスの制限により、package は利用できません。
インストールしたら、以下のように入力してください。

```
% mencoder input.avi -oac mp3lame -lameopts br=192 \
     -ovc lavc -lavcopts vcodec=mpeg4:vhq -o output.avi
```

これは mplayer や xine といったアプリケーションで再生可能な出力ファイルを作成します。

DVD タイトルを直接再エンコードするためには、上記のコマンドラインの input.avi を dvd://1 -dvd-device /dev/dvd に置き換えて、root 権限で実行します。期待する結果を得るには何度か繰り返すことになるので、かわりにタイトルをファイルにダンプして、ファイルに対して作業することをおすすめします。

xine ビデオプレイヤ

xine は、再利用可能な基本ライブラリと、プラグインで拡張できる実行可能なモジュールを提供するビデオプレイヤです。multimedia/xine package または port からインストールできます。

実用上、xine を使用するには高速なビデオカードとともに高速な CPU があるか、またはビデオカードが XVideo 拡張に対応している必要があります。XVideo インタフェースとともに xine ビデオプレイヤを使うのが最良です。

デフォルトでは、xine プレイヤは GUI 付きで起動するでしょう。メニューを使用して特定のファイルを開くことができます。

xine は、再生するファイル名を指定することで、コマンドラインから実行することもできます。

```
% xine -g -p mymovie.avi
```

xine-project.org/faq には、より多くの情報やトラブルシューティングがあります。

Transcode ユーティリティ

Transcode は、ビデオおよびオーディオファイルを再エンコードするためのツール一式です。Transcode を使えば、stdin/stdout ストリームインタフェースとともにコマンドラインツールを用いることで、ビデオファイルの統合や、壊れたファイルの修復ができます。

FreeBSD では、Transcode は、multimedia/transcode package もしくは port からインストールできます。多くのユーザは port からコンパイルすることを好みます。port では、コンパイルで有効にするサポートやコーデックを指定するコンパイルオプションのメニューを利用できるためです。オプションを選択しないと、Transcode は、その形式をエンコード出来ないでしょう。矢印キーとスペースバーを使って、必要とするフォーマットを選択してください。選択が終わったら、Enter を押して、port のコンパイルとインストールを続けてください。

この例では、DivX ファイルを PAL MPEG-1 (PAL VCD) に変換する使用例を示します。

```
% transcode -i input.avi -V --export_prof vcd-pal -o output_vcd
% mplex -f 1 -o output_vcd.mpg output_vcd.m1v output_vcd.mpa
```

作成された MPEG ファイル、output_vcd.mpg は、MPlayer を使って再生できます。また、multimedia/vcdimager および sysutils/cdrdao といったユーティリティを使って、ファイルを CD メディアに書き込むことでビデオ CD も作成できます。

transcode のマニュアルページに加え、transcoding.org/cgi-bin/transcode から、更なる情報や使用例を得てください。

7.5. TV カードの設定

TV カードを使用することで、 TV 放送をコンピュータで見ることができます。 これらの多くのカードは RCA または S-video 入力端子を備えており、 FM ラジオチューナを装備したカードもあります。

FreeBSD は、Brooktree Bt848/849/878/879 をビデオキャプチャチップに採用した PCI TV カードに bktr(4) ドライバで対応しています。 このドライバは、ほとんどの Pinnacle PCTV ビデオカードに対応しています。 TV カードを購入する前に、対応しているチューナの一覧について、bktr(4) を参照してください。

7.5.1. ドライバを読み込む

カードを使用するには、bktr(4) ドライバを読み込む必要があります。
起動時に自動的に読み込むためには、 /boot/loader.conf に以下の行を追加してください。

```
bktr_load="YES"
```

あるいは、カスタムカーネルに TV ビデオカードへのサポートを静的に組み込むこともできます。
この場合には、 次の行をカーネルコンフィギュレーションファイルに追加してください。

```
device   bktr
device  iicbus
device  iicbb
device  smbus
```

カードコンポーネントは I2C バス経由で連結されているため、 bktr(4) ドライバに加えてこれらのデバイスが必要になります。
編集したら新しいカーネルを構築し、インストールします。

チューナが適切に検出されたかどうかを確認するため、 システムを再起動してください。
起動時のメッセージに TV カードが以下のように認識されるでしょう。

```
bktr0: <BrookTree 848A> mem 0xd7000000-0xd7000fff irq 10 at device 10.0 on pci0
iicbb0: <I2C bit-banging driver> on bti2c0
iicbus0: <Philips I2C bus> on iicbb0 master-only
iicbus1: <Philips I2C bus> on iicbb0 master-only
smbus0: <System Management Bus> on bti2c0
bktr0: Pinnacle/Miro TV, Philips SECAM tuner.
```

これらのメッセージはハードウェアに応じて異なります。 必要であれば、sysctl(8) や、カーネルコンフィギュレーションファイルオプションで、
検知されたいくつかのパラメータを変更できます。 たとえば、チューナを Philips SECAM チューナとして検知されるようにするには、
カーネルコンフィギュレーションファイルに以下の行を追加します。

```
options OVERRIDE_TUNER=6
```

または、直接 sysctl(8) を使用して変更します。

```
# sysctl hw.bt848.tuner=6
```

利用可能な sysctl(8) パラメータおよびカーネルオプションについては bktr(4) を参照してください。

7.5.2. 便利なアプリケーション

TV カードを使用するためには、 以下のアプリケーションの一つをインストールする必要があります。

- multimedia/fxtv はウィンドウ内に TV 映像を映します。 画像/音声/ビデオを取り込むこともできます。
- multimedia/xawtv も同様の機能を持った TV アプリケーションです。
- audio/xmradio は TV カードに搭載された FM ラジオチューナを使用するためのアプリケーションです。

他にも多くのアプリケーションが FreeBSD の Ports Collection に収録されています。

7.5.3. トラブルシューティング

TV カードに関する問題が起きたときには、bktr(4) が本当にビデオキャプチャチップおよびチューナに対応しているか、オプションが正しく設定されているかどうかをまず確認してください。 TV カードに関するサポートや質問に関しては、 FreeBSD multimedia メーリングリスト メーリングリストを参照してください。

7.6. MythTV

MythTV は、広く使われているオープンソースの Personal Video Recorder (PVR) アプリケーションです。 この節では、FreeBSD に MythTV をインストールし、設定する方法について説明します。 MythTV の使用法に関するより詳細な情報については、mythtv.org/wiki をご覧ください。

MythTV は、フロントエンドおよびバックエンドを必要とします。 これらは、同じシステム上でも、異なるコンピュータ上でも動かすことが可能です。

フロントエンドについては、 multimedia/mythtv-frontend package または port から FreeBSD にインストールできます。 X Window System で説明されているように、 Xorg をインストールして設定する必要もあります。 このシステムは X-Video Motion Compensation (XvMC) に対応し、 オプションとして、Linux Infrared Remote Control (LIRC)-互換のリモートに対応したビデオカードを持っていることが理想的です。

FreeBSD にバックエンドとフロントエンドの両方をインストールするには、 multimedia/mythtv package または port を使ってください。 MySQL™ データベースサーバも必要となりますが、自動的に依存でインストールされます。オプションで、

チューナカードと録音したデータを保存するためのストレージが必要です。

7.6.1. ハードウェア

MythTV は、 エンコーダやチューナなどのビデオ入力デバイスへのアクセスに Video for Linux (V4L) を用います。 FreeBSD では、USB DVB-S/C/T カードにおいて最もよく動作します。 なぜならば、このカードは、 V4L ユーザランドアプリケーションを提供する multimedia/webcamd package または port により良くサポートされているためです。 webcamd により対応している Digital Video Broadcasting (DVB) カードは、MythTV で動作するはずです。 動作することが知られているカードの一覧が wiki.freebsd.org/WebcamCompat にあります。 Hauppauge カードのドライバもまた、 multimedia/pvr250 および multimedia/pvrxxx port として利用可能ですが、 標準的ではないドライバのインタフェースを提供しており、 0.23 より後の MythTV では動作しません。 ライセンスの制限により、package は利用できません。 そのため、これらの ports はコンパイルをしなければなりません。

wiki.freebsd.org/HTPC ページは、DVB ドライバのすべての一覧を提供しています。

7.6.2. MythTV バックエンドの設定

バイナリ package を使って MythTV をインストールしてください。

```
# pkg install mythtv
```

あるいは、Ports Collection からインストールするには、 以下のように実行してください。

```
# cd /usr/ports/multimedia/mythtv
# make install
```

インストールが終わったら、 MythTV データベースを設定してください。

```
# mysql -uroot -p < /usr/local/share/mythtv/database/mc.sql
```

その後、バックエンドを設定してください。

```
# mythtv-setup
```

最後にバックエンドを起動してください。

```
# sysrc mythbackend_enable=yes
# service mythbackend start
```

7.7. 画像スキャナ

FreeBSD では、画像スキャナに対するアクセスは SANE (Scanner Access Now Easy)

によって実現されており、 FreeBSD の Ports Collection で提供されています。 SANE はスキャナのハードウェアにアクセスするために FreeBSD デバイスドライバを使用します。

FreeBSD は SCSI 接続および USB 接続のスキャナのどちらにも対応しています。 スキャナのインタフェースに依存して、異なるドライバが必要となります。 設定を始める前に、 SANE がスキャナに対応していることを確認してください。
対応しているスキャナに関してのより詳細な情報については、http://www.sane-project.org/sane-supported-devices.html をご覧ください。

この節では、FreeBSD がどのようにしてスキャナを認識するかについて説明します。 その後、FreeBSD システム上で SANE を設定して使用する方法の概要について説明します。

7.7.1. スキャナの確認

GENERIC カーネルには USB スキャナに対応するためのデバイスドライバが搭載されています。 カスタムカーネルを使用する際には、
以下の行がカーネルコンフィグレーションファイルにあることを確認してください。

```
device usb
device uhci
device ohci
device ehci
device xhci
```

USB スキャナが認識されたかを確認するには、 スキャナを接続して、dmesg を利用し、 システムメッセージバッファで、 スキャナが認識されているかどうかを確認してください。 認識されていたら、以下のようなメッセージが表示されます。

```
ugen0.2: <EPSON> at usbus0
```

この例では、EPSON Perfection® 1650 USB スキャナが /dev/ugen0.2 上で認識されています。

スキャナのインタフェースが SCSI であれば、 どの SCSI コントローラボードを使用するかを知ることが重要です。 使用する SCSI チップセットによって、 カスタムカーネルコンフィグレーションファイルを調整する必要があります。 GENERIC カーネルは、 一般に使用される SCSI コントローラのほとんどに対応しています。 /usr/src/sys/conf/NOTES ファイルを読んで、 適切な行をカーネルコンフィグレーションファイルに追加してください。 また、SCSI アダプタドライバに加えて、
以下の行をカスタムカーネルコンフィグレーションファイルに記述する必要があります。

```
device scbus
device pass
```

デバイスがメッセージバッファに出力されていることを確認してください。

```
pass2 at aic0 bus 0 target 2 lun 0
pass2: <AGFA SNAPSCAN 600 1.10> Fixed Scanner SCSI-2 device
```

```
pass2: 3.300MB/s transfers
```

システムを起動する際にスキャナの電源を入れてなければ、 `camcontrol` を使用して SCSI バスをスキャンし、 以下のように手動でデバイスを検出させることもできます。

```
# camcontrol rescan all
Re-scan of bus 0 was successful
Re-scan of bus 1 was successful
Re-scan of bus 2 was successful
Re-scan of bus 3 was successful
```

すると、スキャナは SCSI デバイスの一覧に現れるでしょう。

```
# camcontrol devlist
<IBM DDRS-34560 S97B>              at scbus0 target 5 lun 0 (pass0,da0)
<IBM DDRS-34560 S97B>              at scbus0 target 6 lun 0 (pass1,da1)
<AGFA SNAPSCAN 600 1.10>           at scbus1 target 2 lun 0 (pass3)
<PHILIPS CDD3610 CD-R/RW 1.00>     at scbus2 target 0 lun 0 (pass2,cd0)
```

FreeBSD における SCSI デバイスについての詳細は、 scsi(4) および camcontrol(8) をご覧ください。

7.7.2. SANE の設定

SANE システムは、 バックエンド (graphics/sane-backends) を経由してスキャナに対するアクセスを提供します。
バックエンドが対応している画像スキャナについては、http://www.sane-project.org/sane-supported-devices.html を参照してください。 グラフィカルなスキャニングインタフェースは、 Kooka (graphics/kooka) または XSane (graphics/xsane) といったサードパーティ製のアプリケーションによって提供されています。 SANE のバックエンドは、スキャナを試すには十分です。

バイナリ package から、バックエンドをインストールするには、 以下のように実行してください。

```
# pkg install sane-backends
```

あるいは、Ports Collection からインストールするには、 以下のように実行してください。

```
# cd /usr/ports/graphics/sane-backends
# make install clean
```

graphics/sane-backends port または package をインストールしたら、 `sane-find-scanner` コマンドを使用して、 SANE システムで検出されているスキャナを確認してください。

```
# sane-find-scanner -q
```

```
found SCSI scanner "AGFA SNAPSCAN 600 1.10" at /dev/pass3
```

この出力から、スキャナインタフェースの種類とシステムに接続されているスキャナが使用するデバイスノードがわかります。
ベンダ名や製品のモデル名は表示されないかも知れません。

いくつかの USB スキャナではファームウェアを読み込む必要がある場合があります。
詳細については、sane-find-scanner(1) および sane(7) を参照してください。

次に、スキャナがフロントエンドで認識されるか調べてください。 SANE のバックエンドには scanimage が付属します。このコマンドを使用すると、デバイスの一覧を表示したり画像を取得することができます。 スキャナデバイスの一覧を表示するには、-L オプションを使ってください。 以下の最初の例は、SCSI スキャナ用のもので、 次の例は、USB スキャナ用のものです。

```
# scanimage -L
device `snapscan:/dev/pass3' is a AGFA SNAPSCAN 600 flatbed scanner
# scanimage -L
device 'epson2:libusb:/dev/usb:/dev/ugen0.2' is a Epson GT-8200 flatbed scanner
```

2 番目の出力において、 epson2 がバックエンド名で、 libusb:000:002 は /dev/ugen0.2 を意味し、スキャナが使用するデバイスノードです。

scanimage がスキャナの認識に失敗した場合には、 以下のようなメッセージが表示されます。

```
# scanimage -L

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).
```

このような場合には、/usr/local/etc/sane.d/ にあるバックエンドの設定ファイルを編集して、使用するスキャナデバイスを設定してください。 例えば、認識されなかったスキャナのモデルが、EPSON Perfection® 1650 で、epson2 バックエンドを使っているのであれば、/usr/local/etc/sane.d/epson2.conf を編集してください。 編集作業を行う際には、使用するインタフェースとデバイスノードを指定する行を追加します。
この例では、以下の行を追加します。

```
usb /dev/ugen0.2
```

編集を保存し、
適切なバックエンド名とデバイスノードでスキャナが認識されたかどうかを確認してください。

```
# scanimage -L
device 'epson2:libusb:000:002' is a Epson GT-8200 flatbed scanner
```

scanimage -L を実行してスキャナが認識されたことがわかれば、設定は終了です。スキャナを使用する準備ができました。

scanimage を使用してコマンドラインから画像を取得することができますが、GUI を使用して画像を取得できることが望ましいでしょう。Kooka や xsane といったアプリケーションは、広く使われているスキャニングフロントエンドです。これらには、さまざまなスキャニングモード、色補正、バッチスキャンなど先進的な機能があります。XSane は、GIMP のプラグインとして使用することもできます。

7.7.3. スキャナの許可属性

スキャナにアクセスするには、
ユーザはスキャナが使用するデバイスノードへの読み込み権限と書き込み権限が必要です。
今回の例では、USB スキャナは /dev/ugen0.2 デバイスノードを使用しています。
このデバイスノードは、/dev/usb/0.2.0 へのシンボリックリンクです
シンボリックリンクとデバイスノードは、それぞれ wheel および operator グループが所有しています。
ユーザをこれらのグループに加えると、スキャナを使用できるようになりますが、ユーザを wheel に追加することは、セキュリティの観点からお勧めできません。良い方法は、スキャナデバイスにアクセスできるグループを作成することです。

この例では、usb という名前のグループを作成します。

```
# pw groupadd usb
```

その後、シンボリックリンク /dev/ugen0.2 および、/dev/usb/0.2.0 デバイスノードに対して、usb グループが利用できるように書き込みの許可属性 0660 または 0664 を設定してください。/etc/devfs.rules に次の行を追加すれば設定できます。

```
[system=5]
add path ugen0.2 mode 0660 group usb
add path usb/0.2.0 mode 0666 group usb
```

デバイスを追加したり外すことにより、デバイスノードが変わることがあります。
そのため、すべての USB デバイスにアクセスしたい場合には、
代わりに以下のルールセットを使ってください。

```
[system=5]
add path 'ugen*' mode 0660 group usb
add path 'usb/*' mode 0666 group usb
```

このファイルの詳細については、devfs.rules(5) を参照してください。

つぎに、/etc/rc.conf でルールセットを有効にしてください。

```
devfs_system_ruleset="system"
```

そして、devfs(8) システムを再起動してください。

```
# service devfs restart
```

最後に、スキャナを利用するユーザを usb
グループに追加してスキャナを利用できるようにしてください。

```
# pw groupmod usb -m joe
```

詳細については、pw(8) をご覧ください。

Chapter 8. FreeBSD カーネルのコンフィグレーション

8.1. この章では

カーネルは FreeBSD オペレーティングシステムの中核をなすものです。
カーネルは、メモリ管理、セキュリティ制御の強制、ネットワーク、
ディスクアクセスなどを担っています。 FreeBSD
の大部分は動的に構成することができるようになっていますが、
まだ、時にはカスタムカーネルを設定してコンパイルする必要があります。

この章では、以下のことを扱っています。

- いつカスタムカーネルの構築が必要になるか。
- ハードウェア一覧の作成方法。
- カーネルコンフィグレーションファイルのカスタマイズの方法。
- カーネルコンフィグレーションファイルから新しいカーネルを構築する方法。
- 新しいカーネルのインストール方法。
- うまく行かないときの問題解決法。

この章で表示されているすべてのコマンドは、`root` 権限で実行する必要があります。

8.2. なぜカスタムカーネルを作るか?

伝統的に、FreeBSD はモノリシック (monolithic) カーネルを使っていました。
このカーネルは、単一の巨大なプログラムで、 扱えるデバイスは固定されていて、
カーネルの振る舞いを変えたければ構築してコンピュータを再起動し、
新しいカーネルを動かさなければなりませんでした。

今日では、FreeBSD カーネルのかなりの機能はモジュールに含まれるようになり、
必要に応じて動的にカーネルに組み込んだり外したりできるようになりました。 この移行により、
動作しているカーネルが新しいハードウェアに迅速に対応したり、
カーネルに新たな機能を取り入れられるようになります。 このようなカーネルは、モジュラ (modular)
カーネルと呼ばれます。

しかしながら、 いまだにいくらかは静的にカーネルを構成する必要があります。
機能がカーネルとあまりに密接に結びついているため、
動的に組み込むことができない場合があるためです。 環境によっては、セキュリティの観点から、
カーネルモジュールを読み込んだり外すことができず、
必要となる機能を静的にカーネルにコンパイルしなければならない場合もあります。

システムに合わせたカーネルを構築することは、多くの場合、 高度な知識を持つ BSD
ユーザが避けて通ることのできない通過儀礼です。 この作業は多くの時間を必要としますが、FreeBSD
システムに利益をもたらします。 広範囲のハードウェアをサポートしなければならない GENERIC
カーネルとは異なり、カスタムカーネルは、
使用しているコンピュータのハードウェアのみをサポートするように、
必要のない機能を省くことができます。これは、 次にあげるような利益をもたらします。

- 素早く起動します。 カーネルはシステム上にあるハードウェアしか検出しないので、システムの起動にかかる時間を短くできます。
- メモリの消費量を減らすことができます。 システムに合わせたカーネルは、使用しない機能やデバイスドライバを含まないので、 大抵 GENERIC カーネルより少ないメモリしか消費しません。 カーネルコードは常に物理メモリ上に存在し、アプリケーションはその容量分のメモリを使用できないので、 これは重要なことです。したがって、メモリが少ないシステムでは、 カーネルの再構築は重要です。
- 追加のハードウェアをサポートします。 カスタムカーネルは、GENERIC カーネルに存在しないデバイスのサポートを追加することができます。

カスタムカーネルを構築する前に、再構築する理由を考えてください。
ある特定のハードウェアに対応する必要がある場合に、
そのハードウェアに対応するためのモジュールがすでに用意されていることがあります。

カーネルモジュールは /boot/kernel にあります。 モジュールによっては kldload(8) により、すでに実行中のカーネルに動的に読み込まれています。
ほとんどのカーネルドライバには、読み込み可能なモジュールやマニュアルページが用意されています。
たとえば、ath(4)
ワイヤレスネットワークドライバのマニュアルページには以下のような記述があります。

```
Alternatively, to load the driver as a module at boot time, place the
following line in loader.conf(5):

    if_ath_load="YES"
```

/boot/loader.conf に if_ath_load="YES" を追加すると、
起動時にモジュールが読み込まれるようになります。

対応するモジュールが /boot/kernel に存在しないこともあります。
特定のサブシステムでは、ほとんど多くの場合存在しません。

8.3. システムのハードウェアについて知る

カーネルコンフィグレーションファイルの編集を始める前に、
コンピュータのハードウェア一覧を作成すると良いでしょう。 デュアルブートシステムでは、現在インストールされている別のオペレーティングシステムの設定を調べることで、
一覧を作成できます。 たとえば、Microsoft® の デバイスマネージャは、インストールされているデバイスに関する情報を持っています。

Microsoft® Windows® のバージョンによっては、 システム アイコンを使って、デバイスマネージャ にアクセスできます。

インストールされているオペレーティングシステムが FreeBSD だけであれば、dmesg(8) を使い、起動時に検出されたハードウェアの一覧を調べてください。 FreeBSD のほとんどのデバイスドライバにはマニュアルページが用意され、
対応しているハードウェアの一覧を提供しています。 たとえば、以下の行は、psm(4) ドライバがマウスを検出したことを示しています。

```
psm0: <PS/2 Mouse> irq 12 on atkbdc0
psm0: [GIANT-LOCKED]
psm0: [ITHREAD]
psm0: model Generic PS/2 mouse, device ID 0
```

このハードウェアはシステムに存在するので、
カスタムカーネルコンフィグレーションファイルからこのドライバを外さないでください。

dmesg が起動時の検出結果を表示しない場合には、かわりに /var/run/dmesg.boot で出力を確認してください。

ハードウェアを見つけるためのもうひとつのツールは、より冗長な出力を行う pciconf(8) です。
たとえば、以下のようになります。

```
% 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
```

この出力は、ath ドライバがワイヤレスイーサネットデバイスにあることを示しています。

man(1) を -k フラグで実行すると、有用な情報を得ることができます。たとえば、
ある特定のデバイスブランドや名前を含むマニュアルページの一覧を表示するには、
以下のように実行してください。

```
# man -k Atheros
ath(4)                   - Atheros IEEE 802.11 wireless network driver
ath_hal(4)               - Atheros Hardware Access Layer (HAL)
```

ハードウェアの一覧を作成したら、この一覧を利用して、
カスタムカーネルのコンフィグレーションファイルを編集している時に、
インストールされているハードウェアのドライバが削除されていないことを確認してください。

8.4. コンフィグレーションファイル

カスタムカーネルのコンフィグレーションファイルを作成し、カスタムカーネルを構築するには、
FreeBSD の全ソースツリーがまずインストールされている必要があります。

もし /usr/src/ が存在していなかったり、空であれば、
カーネルのソースはインストールされていません。「Git の利用」で説明した Git
を使ってソースをインストールしてください。

ソースをインストールしたら、/usr/src/sys を確認して下さい。
このディレクトリには、いくつものサブディレクトリがあります。
その中には、サポートされている各アーキテクチャ amd64, i386, powerpc および sparc64

のサブディレクトリがあります。
各アーキテクチャのディレクトリ内部にあるファイルはすべてそのアーキテクチャでのみ使用されます。
残りのコードは、アーキテクチャに依存しない、　すべてのプラットフォームで共有されるコードです。
サポートされている各アーキテクチャには、　conf　サブディレクトリがあり、　そのアーキテクチャ用の GENERIC カーネルコンフィグレーションファイルが用意されています。

この　GENERIC　は編集しないでください。
かわりに、このファイルを別名でコピーし、コピーを編集してください。
慣習として、この名前はすべて大文字でつづられます。もし、　いくつかの異なるハードウェアの FreeBSD　マシンを扱うなら、　この名前にホスト名を含めるとよいでしょう。ここでは、例として MYKERNEL　という名前の　amd64　アーキテクチャ用の　GENERIC　コンフィグレーションファイルのコピーを作成します。

```
# cd /usr/src/sys/amd64/conf
# cp GENERIC MYKERNEL
```

これで、MYKERNEL　を　ASCII　テキストエディタで編集できます。
初心者に対してより簡単なエディタである　ee　も　FreeBSD　とともにインストールされていますが、デフォルトのエディタは vi です。

コンフィグレーションファイルのフォーマットはシンプルです。
各行はデバイスやサブシステム、引数、または簡単な説明を含んでいます。　#　に続くテキストはすべてコメントとして扱われ、無視されます。
カーネルからデバイスもしくはサブシステムのサポートを外すには、対応する行の最初に　#　を入れてください。 理解していない行に対しては、# を追加したり削除しないでください。

デバイスやオプションのサポートを外すことは簡単で、
その結果、カーネルを壊すことがあります。　たとえば　ata(4)　ドライバをカーネルコンフィグレーションファイルから除くと、　ATA　ディスクドライバを用いているシステムは起動しません。
確信が持てないものについては、 カーネルにサポートを残したままにしてください。

このファイルで与えられる説明の他に、　そのアーキテクチャの　GENERIC　と同じディレクトリにある NOTES　にも説明があります。　アーキテクチャに依存しないオプションについては、/usr/src/sys/conf/NOTES をご覧ください。

カーネルコンフィグレーションファイルの編集を終えたら、 ファイルのバックアップを /usr/src 以外の場所に保存してください。

または、カーネルコンフィグレーションファイルは他の場所において、
シンボリックリンクを張る方法もあります。

```
# cd /usr/src/sys/amd64/conf
# mkdir /root/kernels
# cp GENERIC /root/kernels/MYKERNEL
# ln -s /root/kernels/MYKERNEL
```

コンフィグレーションファイルでは include ディレクティブを利用できます。
コンフィグレーションファイルに他のファイルを取り込むことができるので、
すでに存在するファイルに対する小さな変更の管理が簡単にできます。
オプションやドライバの追加が少しだけの場合には、以下の例のように GENERIC からの差分による管理が可能になります。

```
include GENERIC
ident MYKERNEL

options         IPFIREWALL
options         DUMMYNET
options         IPFIREWALL_DEFAULT_TO_ACCEPT
options         IPDIVERT
```

この方法では、ローカルのコンフィグレーションファイルには、ローカルにある GENERIC カーネルとの差分が記述されています。アップグレードが行われると、GENERIC に追加された新しい機能は、(nooptions や nodevice によって外されない限り) ローカルのカーネルにも反映されます。 コンフィグレーションの構成要素に関する包括的な一覧と説明は config(5) にあります。

利用可能なすべてのオプションを含むファイルを構築するには、以下のコマンドを root 権限で実行してください。

```
# cd /usr/src/sys/arch/conf && make LINT
```

8.5. カスタムカーネルの構築とインストール

カスタムコンフィグレーションファイルを編集して保存したら、
カーネルのソースコードを以下の手順でコンパイルしてください。

Procedure: カーネルの構築

+ . 以下のディレクトリに移動してください。

+

```
# cd /usr/src
```

+ . カスタムコンフィグレーションファイルの名前を指定して新しいカーネルをコンパイルします。

+

```
# make buildkernel KERNCONF=MYKERNEL
```

+ .

指定したカーネルコンフィグレーションファイルでコンパイルされた新しいカーネルをインストールします。 以下のコマンドは、新しいカーネルを /boot/kernel/kernel に、 今までのカーネルを /boot/kernel.old/kernel という名前で保存します。

+

```
# make installkernel KERNCONF=MYKERNEL
```

+ . 新しいカーネルを使うために、 システムをシャットダウンして再起動してください。 うまく行かない場合は、カーネルが起動しない を参照してください。

デフォルトでは、カスタムカーネルを構築すると、 すべてのカーネルモジュールが再構築されます。 カーネルのアップデートをより早く行いたい、または、 カスタムモジュールのみを構築したいといった場合は、 カーネルの構築を開始する前に、以下のように /etc/make.conf を編集してください。

例として、以下の変数は、 デフォルトのすべてのモジュールを構築する設定を変更し、 構築するモジュール一覧を指定します。

```
MODULES_OVERRIDE = linux acpi
```

また、以下の変数は、構築を行わないモジュールを指定します。

```
WITHOUT_MODULES = linux acpi sound
```

他の変数については、make.conf(5) を参照してください。

8.6. 問題が起きた場合には

カスタムカーネルを作る際に起こりうるトラブルは、 次の 4 種類に分けられます。

config コマンドの失敗

config で失敗した時には、 トラブルの起きた行番号が出力されます。 たとえば、次のように出力された場合には、 17 行目が正しく入力されているかどうか、 GENERIC や NOTES と比較して修正してください。

```
config: line 17: syntax error
```

make コマンドの失敗

make が失敗した場合には、 通常、カーネルコンフィグレーションファイルにおいて、 config がとらえられなかったような間違いをしています。 コンフィグレーションファイルを見直してください。 それでも問題を解決することができなければ、 FreeBSD general questions メーリングリスト へカーネルコンフィグレーションファイルを添付して送ってください。

カーネルが起動しない

新しいカーネルが起動しなかったり、デバイスの認識をしない場合でもあわてないでください! さいわい、FreeBSD には利用できないカーネルから復帰する洗練されたメカニズムがあります。FreeBSD のブートローダで起動したいカーネルを選択してください。システムの起動メニューが表示されている時に、"Escape to a loader prompt" オプションを選択するとアクセスできます。プロンプトで `boot kernel.old` か他の正常に起動するカーネルを入力してください。

問題のないカーネルで起動した後、コンフィグレーションファイルを調べ、再び構築を試みてください。/var/log/messages にはすべての成功した起動時のカーネルメッセージの記録があり、これは問題を解決するための助けになる情報の一つでしょう。また、dmesg(8) は現在の起動時のカーネルメッセージを出力します。

カーネルのトラブルシューティングを行う時には、GENERIC といった正常に起動するカーネルのコピーを保存するようにしてください。*kernel.old* は新しいカーネルをインストールする時に、その一つ前にインストールした、うまく動かないかもしれないカーネルで上書きされてしまうため、起動するカーネルを保存しておくことは重要です。
できる限り早く以下のようにして、正しく起動するカーネルを含むディレクトリ名に変更してください。

```
# mv /boot/kernel /boot/kernel.bad
# mv /boot/kernel.good /boot/kernel
```

カーネルは動きますが **ps(1)** は動きません!

システムユーティリティの構築されたバージョンと異るバージョンのカーネルをインストールした場合、たとえば -CURRENT のソースから構築したカーネルを -RELEASE システム上にインストールするような場合には、ps(1) や vmstat(8) のような多くのシステムステータスコマンドは動かなくなります。
修正するには、カーネルと同じバージョンのソースツリーで world を再構築し、インストールしてください。
カーネルとそれ以外で異なるバージョンを組み合わせてオペレーティングシステムを使用することは推奨されていません。

Chapter 9. プリンタの利用

9.1. この章では

FreeBSD は古いインパクトプリンタから最新のレーザープリンタまで幅広いプリンタが利用でき、実行しているアプリケーションから高品質な印刷出力が行えます。

FreeBSD はネットワーク上のプリンタサーバとして動作するように設定することもできます。この機能は、他の FreeBSD コンピュータや、Windows® や Mac OS® ホストから印刷ジョブを受け取ることができます。 FreeBSD は印刷ジョブを 1 つずつ処理することを保証します。 また、どのユーザやマシンが最も多く印刷しているかの統計を取り、どの印刷物が誰の物か表示する "バナー" ページの作成などを行うことができます。

この章を読めば以下のことがわかります。

- FreeBSD プリントスプーラの設定方法。
- 入力ドキュメントをプリンタが扱える印刷フォーマットへ変換するなどといった、特別な印刷ジョブを別に取り扱うための印刷フィルタのインストール方法。
- 印刷物へのヘッダやバナーの適用方法。
- 他のコンピュータに接続されたプリンタで印刷する方法。
- ネットワークに直接接続されたプリンタで印刷する方法。
- 印刷ジョブの上限サイズや特定のユーザからの印刷拒否といった、 プリンタの制限の制御方法。
- 印刷の統計とプリンタの使用状況の取得方法。
- 印刷問題のトラブルシューティング方法。

この章を読み始める前に以下を済ませておいてください。

- 新しいカーネルの設定とインストール方法について理解すること (FreeBSD カーネルのコンフィグレーション)。

9.2. はじめに

FreeBSD でプリンタを使うために、それらを LPD スプーリングシステム、 または単に LPD としても知られる Berkeley ラインプリンタスプーリングシステムで動作するように設定できます。これは FreeBSD での標準的なプリンタ制御システムです。 この章では、LPD を紹介し、その設定方法について説明します。

あなたがすでに LPD やその他のプリンタスプーリングシステムに詳しいのなら、 基本的な設定 まで読み飛ばしてもかまいません。

LPD はホストのプリンタに関するあらゆることを制御します。ここで言う制御としては、次のことがあげられます。

- ホストに接続されたプリンタ、あるいはネットワーク上の他ホストに接続されたプリンタに対するアクセス制御を行ないます。
- ファイルをプリントする要求に対して許可を与えます。 この要求は特にジョブと呼ばれています。

- 各々のプリンタのキューを管理することにより、
  複数のユーザがあるプリンタに対して同時にアクセスすることを防ぎます。
- ヘッダページ (バナーまたは バーストページとしても知られています)
  をプリントすることができます。 これにより、
  プリントアウトの山の中から自分がプリントしたジョブを見つけやすくなります。
- シリアルポートに接続したプリンタ用に通信パラメータを管理します。
- ネットワーク経由で他のホスト上の LPD スプーラにジョブを送ることができます。
- 様々なプリンタ言語やプリンタの能力に応じてジョブの形式を整えるため、
  特別なフィルタを起動することができます。
- プリンタの使用に対して課金を行なうことができます。

設定ファイル (/etc/printcap) を通して、専用のフィルタプログラムを用いることにより、
多種多様なプリンタ機器に対して、上述の機能の全部または一部を LPD
システムに行なわせることができます。

9.2.1. どうしてスプーラを使うべきなのか

あなたのシステムを利用するのがあなた一人だけだとしても、
スプーラは有用ですし、使用すべきです。その理由は以下のとおりです。

- LPD はジョブをバックグラウンドで処理します。
  データがプリンタに送信されるまで待つ必要がなくなります。
- LPD ではジョブをフィルタを通してプリントすることが簡単にできます。
  これにより、印刷物のヘッダに時刻や日付を入れたり、 特別なファイル形式 (TeX の DVI
  ファイルなど) をプリンタが処理できる形式に変更することができ、
  これらの作業を手動で行なう必要がなくなります。
- プリント処理を行なうフリー、 または商用のプログラムのほとんどは、
  システムのスプーラとやりとりするように作られています。
  スプーリングシステムをセットアップすることで、 今後加えるかもしれない、あるいは、
  すでに持っている別のソフトウェアをより簡単にサポートすることができるでしょう。

9.3. 基本的な設定

LPD スプーリングシステムを用いてプリンタを使用するためには、 プリンタ機器と LPD
用ソフトウェアの両方を準備する必要があります。
本文書では次の二段階のレベルに分けて説明をします。

- プリンタを接続する方法、 プリンタにどのように通信するかを LPD に指示する方法や、
  プレインテキストをプリンタで印字する方法については、 プリンタの簡単な設定をご覧ください。
- 様々な形式のファイルを印字する方法、 ヘッダページを印字する方法、
  ネットワーク経由でプリンタに印字する方法、 プリンタを制御する方法、
  プリンタの使用に対する課金を行なう方法についてはプリンタ設定上級編をご覧ください。

9.3.1. プリンタ設定導入編

この節では、プリンタ機器やプリンタを使用するための LPD

用ソフトウェアを設定する方法について述べます。 この節の概要は次のとおりです。

- プリンタ機器の設定では、
  プリンタをコンピュータに接続するためのヒントがいくつか書かれています。
- ソフトウェアの設定では、 LPD のスプーラ設定ファイル (/etc/printcap)
  の設定方法について書かれています。

データをプリンタに送るのにコンピュータのローカルインタフェースではなく、
ネットワークプロトコルを使用する場合は、
ネットワークにおけるデータストリームインタフェースを持つプリンタをご覧ください。

この節のタイトルは "プリンタ設定導入編" ですが、 実際の設定はかなり複雑です。
プリンタをコンピュータに接続し、 LPD スプーラを起動させることは一番困難な作業です。
ヘッダページを出力させたり課金したりするオプションの設定は、
一度プリンタがうまく動くようになればとても簡単です。

プリンタ機器の設定

この節では、プリンタに PC を接続するための様々な方法について説明しています。
ここでは、ポートやケーブルの種類、 FreeBSD
がプリンタとの通信に必要なカーネルコンフィグレーションについても言及しています。

もしプリンタが既に接続されていて、
他のオペレーティングシステム上でプリンタからの印字に成功している場合は、
ソフトウェアの設定まで読み飛ばすことが多分できるでしょう。

ポートとケーブル

今日 PC 用に売られているプリンタには通常、 次の 3 つのインタフェースのうち、どれか 1
つ以上がついてきます。

- シリアルインタフェース (RS-232 または COM ポートとも呼ばれます) は、
  コンピュータにあるシリアルポートを使ってプリンタにデータを送信します。
  シリアルインタフェースはコンピュータ業界で共通して使用されています。
  そのケーブルは容易に手に入りますし、簡単に自作することもできます。
  シリアルインタフェースの場合は時々、
  特別なケーブルや何か複雑な通信方式選択の設定が必要になることがあります。 ほとんどの PC
  のシリアルポートは通信速度が最大で 115200 bps であり、
  大きな画像を印刷するのには実用的ではありません。
- パラレルインタフェースではプリンタにデータを送信するために、
  コンピュータにあるパラレルポートを使用します。 パラレルインタフェースは PC
  業界ではよく使われており、 RS-232 シリアルよりも速いです。 ケーブルの入手は容易ですが、
  自作するのはシリアルよりも困難です。 パラレルインタフェースには通常、通信方式の選択はなく、
  設定は極めて単純です。

  パラレルインタフェースは "セントロニクス" インタフェースとして知られています。
  これは、プリンタ用のコネクタタイプとして採用された後に名付けられました。
- USB インタフェースは、Universal Serial Bus (汎用シリアルバス) の略で、パラレルや RS-232
  シリアルよりさらに速く動作します。 ケーブルは単純で安価です。USB は、印刷目的には RS-232
  シリアルやパラレルよりも向いていますが、UNIX® システムでは十分対応されていません。

この問題を回避する手としては、多くのプリンタがそうですが、USB とパラレルの両方のインタフェースを備えたプリンタを購入することが挙げられます。

パラレルインタフェースでは、普通は(コンピュータからプリンタへの)単方向通信のみを行なうのに対して、 シリアルおよび USB インタフェースは双方向通信を行ないます。FreeBSD でも IEEE1284 準拠のケーブルを使えば、 最近のパラレルポート (EPP や ECP) とプリンタの多くで双方向通信を行なうことができます。

パラレルポート経由のプリンタとの双方向通信には、通常 2 つの方法のどちらかが使われます。一つ目の方法は、プリンタが使用しているプロプライエタリな言語を話す FreeBSD 用に作成されたプリンタドライバを使うものです。これはインクジェットプリンタではよく使われる方法で、インクの残量やその他の状態の情報を知らせるのに使えます。 二つ目の方法は、プリンタが PostScript® に対応している時に使われます。

PostScript® ジョブは、実際にはプリンタに送信されるプログラムです。印字作業を行う必要は必ずしありませんし、プログラムの結果を直接コンピュータに返してもよいのです。PostScript® プリンタでは双方向通信を使って PostScript® プログラムのエラーや紙づまりといった問題をコンピュータに報告します。ユーザはそれらの情報を知りたいと思うかも知れません。また、PostScript® プリンタで課金作業をもっとも効率よく行なうためには、双方向通信が必要となります。この方法ではまず、プリンタの現在のページカウント (起動してから今まで何枚の紙を印字したか) の情報を得ます。次に、ユーザのジョブを実行し、終了後、再びページカウントを得ます。この二つの数を差によって、 課金対象となる紙の枚数を知ることができるのです。

パラレルポート

プリンタをパラレルインタフェースを使って接続する場合は、セントロニクスケーブルでプリンタとコンピュータを接続してください。詳しい説明はプリンタやコンピュータに付属する説明書に書かれているはずです。

その際、 どのパラレルポートを使用したかを覚えておいてください。 FreeBSD では最初のポートは ppc0、二番目が ppc1 であり、三番目以降も同様に続きます。プリンタのデバイス名にも同じ形式が使われており、最初のパラレルポートに接続されたプリンタは /dev/lpt0 などとなります。

シリアルポート

シリアルインタフェースを使ってプリンタを使う場合は、適切なシリアルケーブルでプリンタとコンピュータを接続してください。詳しい説明はプリンタ、コンピュータ、あるいは両方に付属する説 明書に書かれているはずです。

"適切なシリアルケーブル" が良くわからないときは、 次のどれかを試してみてください。

- モデム用ケーブルでは、 それぞれのピンは他方のコネクタの対応するピンと線でつながっています。このタイプのケーブルは "DTE-DCE" 間ケーブルとしても知られています (訳注: 日本ではストレートケーブルという名前で売られています)。
- ヌルモデム用ケーブルでは、あるピンは対応するピンとを接続していますが、あるピン (たとえば、データ送信用とデータ受信用のピン) が交差して接続したり、

いくつかのピンは内部で短絡していたりします。 このタイプのケーブルは、 "DTE-DTE" 間ケーブルと呼ばれています (訳注: 日本ではクロスケーブルという名前で売られています)。

- A シリアルプリンタ用ケーブルは、 ある特定のプリンタで必要とされるものです。 ヌルモデムケーブルと似ていますが、 内部で短絡させる代わりに、 ある信号を他方側に送るために使用しています。

この他に、 プリンタ用の通信パラメータを設定する必要があります。 通常、プリンタのフロントパネルや DIP スイッチによって制御します。 コンピュータとプリンタの双方で設定できる最高の通信速度 [bps] (ビット/秒、 ボーレートと示されているときもある) を選んでください。そして、データビット (7 または 8)、 パリティ (偶/奇/なし)、ストップビット (1 または 2) を選んでください。 そして、フローコントロールの有無 (制御なし、または XON/XOFF ("イン・バンド" または "ソフトウェア" フローコントロールとも呼ばれる)) を選びます。 以下に続くソフトウェアの設定のために、 ここでの設定を覚えておいてください。

ソフトウェアの設定

本節では FreeBSD の LPD スプーリングシステムで印字をおこなうために 必要となるソフトウェアの設定について説明しています。

本節の概要は次のようになります。

1. プリンタで使用するポートのために、必要があれば、 カーネルの書き変えをおこないます。「カーネルの変更」で、 このためにしなくてはならないことを説明しています。
2. パラレルポートを使用している場合は、 パラレルポートのための通信モードを設定します。 詳細は、 「パラレルポートの通信モードを設定する」 で説明しています。
3. オペレーティングシステムからプリンタにデータが送ら れているかをテストします。「プリンタとの通信状況を調べる」で、 どのようにテストするかの提案をいくつかおこなっています。
4. ファイル/etc/printcapを変更し、 LPD の設定をおこないます。 この節で、どのように変更するかを説明しています。

カーネルの変更

オペレーティングシステムのカーネルの コンパイルをおこなうことによって、 指定されたデバイスが機能するようになります。シリアル、 または、パラレルインタフェースをプリンタで使用する場合、 必要なデバイスがこの指定の中に含まれていなくてはなりません。 したがって、 必要なデバイスがカーネルに組み込まれていない場合、 追加のシリアル、または、パラレルポートをサポートするために、 カーネルの再コンパイルが必要となるかもしれません。

シリアルポートが現在使用しているカーネルで サポートされているかどうかを調べるためには、 次のように入力します。

```
# grep sioN /var/run/dmesg.boot
```

ここで、*N* はシリアルポートの番号を示し、この番号は 0 から始まります。
次のような出力があった場合、 カーネルはそのポートをサポートしています。

```
sio2 at port 0x3e8-0x3ef irq 5 on isa
 sio2: type 16550A
```

パラレルポートが現在使用しているカーネルで サポートされているかどうかを調べるためには、
次のように入力します。

```
# grep ppcN /var/run/dmesg.boot
```

ここで、*N* はパラレルポートの番号を示し、この番号は 0 から始まります。
次のような出力があった場合、 カーネルはそのポートをサポートしています。

```
ppc0: <Parallel port> at port 0x378-0x37f irq 7 on isa0
ppc0: SMC-like chipset (ECP/EPP/PS2/NIBBLE) in COMPATIBLE mode
ppc0: FIFO with 16/16/8 bytes threshold
```

上記の出力が得られない場合、プリンタを使うため、 オペレーティングシステムにパラレル、または、
シリアルポートを認識し、使用できるようにするためには カーネルを変更する必要があります。

シリアルポートをサポートさせるには、「FreeBSD カーネルのコンフィグレーション」の節をご覧く
ださい。パラレルポートをサポートさせる場合も、その節と、 あわせて、
この節に続く節もご覧ください。

パラレルポートの通信モードを設定する

パラレルインタフェースを使用している場合、FreeBSD では、 割り込み駆動型にするか、
プリンタとの通信の状況をカーネルに監視させるかのいずれかを選択できます。 FreeBSD
の汎用プリンタデバイスドライバ (lpt(4)) は ppbus(4) システムを利用しています。 これは ppc(4)
ドライバを使ってパラレルポートのチップセットを制御します。

- GENERIC カーネルでは割り込み駆動方式がデフォルトになっています。 この方式では、
  オペレーティングシステムはプリンタがデータを受け付けられるかどうかを調べるために、 IRQ
  ラインを一つ使用します。
- 監視方式では、
  オペレーティングシステムにプリンタがもっとデータを受け付けられるかどうかを繰り返し尋ねるよ
  うに指示します。 そして、受け付けるという応答を受けたとき、
  カーネルはさらなるデータを送信します。

割り込み駆動方式は一般的にいくらか高速になりますが、貴重な IRQ ラインを一つ消費します。 HP
の新しいプリンタの一部には、明らかに何かしらのタイミングの問題 (まだ正確にはわかっていません)
で割り込みモードでは正常に動作しないものがあると言われています。
これらのプリンタにはポーリングモードが必要になります。
どちらかうまく機能する方を使ってください。 一部のプリンタはどちらの方式でも動作しますが、
割り込みモードでは苦痛を感じるほど低速です。

通信モードを設定するためには 2 つの方法があります。 1 つはカーネルを変更することで、もう一つは lptcontrol(8) プログラムを使用する方法です。

カーネルを設定することによって、 通信モードを変更する。

1. カーネルコンフィグレーションファイルを変更します。 ppc0 のエントリを探してください。 2 番目のパラレルポートを設定するときは、代わりに ppc1 を使います。 以下、3 番目のポートは ppc2 となっていきます。
   - 割り込み駆動方式にする場合は、 /boot/device.hints ファイルの以下の行を編集して、 *N* を適切な IRQ 番号に置き換えてください。

     ```
     hint.ppc.0.irq="N"
     ```

     カーネルの設定ファイルには ppc(4) ドライバも入れなければなりません。

     ```
     device ppc
     ```

   - ポーリングモードを使用する場合は、 /boot/device.hints ファイルの以下の行を削除してください。

     ```
     hint.ppc.0.irq="N"
     ```

     場合によっては、これだけでは FreeBSD でポートをポーリングモードにするには十分ではないことがあります。 多くの場合これは acpi(4) ドライバと併せて動作します。 これはデバイスのプローブとアタッチを行うので、プリンタポートへのアクセスモードを制御できます。 問題を修正するために acpi(4) の設定を確認してください。
2. ファイルをセーブし、config プログラムを起動し、カーネルの構築、インストールをおこないます。そして、リブートしてください。詳細は、「FreeBSDカーネルのコンフィグレーション」を参照してください。

lptcontrol(8) で通信モードを設定する場合

1. lptN をイベント駆動方式に設定する場合は、 次のように入力します。

   ```
   # lptcontrol -i -d /dev/lptN
   ```

2. lptN を監視方式に設定する場合は、次のように入力します。

   ```
   # lptcontrol -p -d /dev/lptN
   ```

これらのコマンドを /etc/rc.local ファイルに追加しておくと、システムをブートする度に通信モードを設定することができます。詳細については、lptcontrol(8) をご覧ください。

プリンタとの通信状況を調べる

スプーリングシステムの設定に進む前に、オペレーティングシステムがプリンタにデータを送ることに成功しているかどうかを確かめるべきでしょう。これにより、印字がうまくいかないとき、プリンタとの通信が問題なのか、スプーリングシステムが問題なのかを分けて調べることがかなり容易になります。

プリンタをテストするためには、プリンタに何かのテキストを送信してみます。送信した文字をすぐに印字してくれるプリンタには、lptest(1) コマンドを使うと有用です。このコマンドは印字可能な 96 文字の ASCII 文字すべてを 96 行生成します。

PostScript® (または他の言語に対応した) プリンタの場合は、もっと巧妙なテストが必要になります。次のような、簡単な PostScript® プログラムを使えば十分でしょう。

```
%!PS
100 100 moveto 300 300 lineto stroke
310 310 moveto
/Helvetica findfont 12 scalefont setfont
(Is this thing working?) show
showpage
```

上の PostScript® コードはファイルに保存し、以降の節で例として示されているように利用することができます。

このドキュメントでプリンタ用言語を参照するときは、PostScript® のような言語を仮定しており、Hewlett Packard の PCL は考慮していません。PCL は非常に機能的なのですが、プレインテキストにエスケープシーケンスを混ぜることができます。PostScript® ではプレインテキストを直接印字することはできません。このような種類のプリンタ言語に対しては、特別な対応をおこなわなければなりません。

パラレルポートのプリンタとの接続を調べる

この節では、FreeBSD がパラレルポートに接続されたプリンタと通信できているかどうかを調べる方法について説明し ています。

パラレルポートのプリンタをテストするために

1. su(1) コマンドで `root` になります。
2. プリンタにデータを送ります。
   - プリンタがプレインテキストを印字できる場合、lptest(1) コマンドを使います。次のように入力してください。

```
# lptest > /dev/lptN
```

ここで、*N* はパラレルポートの番号で、番号は 0 から始まります。

- プリンタが PostScript® か他のプリンタ言語を使用している場合、そのプリンタに簡単なプログラムを送信してください。次のように入力します。

```
# cat > /dev/lptN
```

そして、一行一行、 プログラムを慎重に入力して 下さい。RETUREN または ENTER キーを入力してしま うと、その行は編集できなくなります。プログラムの入力が終わったら、CONTROL+D か、あなたが設定しているファイル終了のキーを押してください。

もしくは、プログラムを入力したファイルがある 場合は、次のように入力してください。

```
# cat file > /dev/lptN
```

ここで、*file* はプログラムが格納されていて、 プリンタに送信するファイルの名前です。

これで何かが印刷されるはずです。 印字されたテキストがおかしくても心配は無用です。それについては、後で修正します。

シリアルポートのプリンタとの接続を調べる

この節では、FreeBSD がシリアルポートに接続されたプリンタと通信できているかどうかを調べる方法について述べられ ています。

シリアルポートのプリンタをテストするために

1. su(1) コマンドで `root` になります。
2. /etc/remote ファイルを編集します。次のエントリを加えてください。

```
printer:dv=/dev/port:br#bps-rate:pa=parity
```

ここで、*port* シリアルポート (`ttyu0`、 `ttyu1` など) のデバイスエントリで、 *bps-rate*はプリンタとの通信の転送速度[bit/秒]、 *parity*はプリ ンタとの通信で必要とされるパリティ (`even`、`odd`、 `none`、 `zero`のいずれか) を表わしていま す。

次の例は、 プリンタをシリアルケーブルでパリティなし、転送速度 19200 bps で第 3 番目のシリアルポートに接続した場 合です。

```
printer:dv=/dev/ttyu2:br#19200:pa=none
```

3. tip(1) コマンドでプリンタと接続します。 次のように入力してください。

```
# tip printer
```

これがうまくいかなかった場合は、 /etc/remoteを編集して、 /dev/ttyuN の代わりに /dev/cuaaN を試してみてください。

4. プリンタにデータを送ります。
   - プリンタがプレインテキストを印字できる場合、 lptest(1) コマンドを使います。 次のように入力してください。

```
% $lptest
```

   - プリンタが PostScript® か他のプリンタ言語を使用している場合、 そのプリンタに簡単なプログラムを入力します。 一行一行、プログラムを慎重に入力してください。 バックスペースキーや他の編集用のキーは、 プリンタの制御コードに割り当てられているかもしれません。 プログラムが終了したことをプリンタに伝えるための特別なファイル終了キーを 入力する必要があるかもしれません。 PostScript® プリンタの場合、 CONTROL+D を入力します。

     もしくは、プログラムを入力したファイルがある場合は、 次のように入力してください。

```
% >file
```

     ここで、*file* はプログラムが格納されているファイル名です。 tip(1) コマンドでファイルを送信した後は、 ファイル終了を表わすキーを入力する必要があります。

これで何かがプリントされることでしょう。 印字されたテキストがおかしくても心配しなくても構いません。 それについては、後で修正します。

スプーラに許可を与える: **/etc/printcap** ファイル

ここまでで、プリンタはコンピュータに接続され、(必要なら) プリンタと通信できるようにカーネルを変更し、 簡単なデータをプリンタに送信することができているはずです。 これで、LPD にプリンタへのアクセスを制御させる設定をおこなう準備が整いました。

LPD の設定は /etc/printcap を編集することでおこないます。 LPD スプーリングシステムは スプーラが使われる毎にこのファイルを参照します。 そのため、ファイルを更新するとすぐにその変更が反映されます。

printcap(5) ファイルの書式は簡単です。 /etc/printcap の編集はお好みのテキストエディタをお使いください。このファイルの書式は、 /usr/shared/misc/termcap や /etc/remote といった他のケイパビリティファイルと一致しています。 この書式 についての詳細な情報については

cgetent(3) をご覧ください。

スプーラの単純な設定法は、 次のステップでおこないます。

1. プリンタに名前 (と簡単な別名 2 ～ 3 個) を付け、それを /etc/printcap ファイルに記述します。これについては、「プリンタに名前を付ける」 を参照してください。
2. sh の項目を追加することで、 ヘッダページの出力を禁止します (デフォルトは許可)。これについては、「ヘッダページの印字を禁止する」 を参照してください。
3. スプール用のディレクトリを作成し、その位置を sd 項目で指定します。これについては、「スプーリングディレクトリの作成」 を参照してください。
4. プリンタを使用するために /dev エントリを設定し、/etc/printcap の lp 項目でそのエントリを指定します。これについては、「プリンタデバイスの特定」を参照してください。プリンタをシリアルポートに接続した場合は、ms# の項目を設定する必要があります。こちらについては、「スプーラのための通信パラメータの設定」 を参照してください。
5. プレインテキスト用の入力フィルタのインストールをおこないます。「テキストフィルタのインストール」 を参照してください。
6. lpr(1) コマンドで何かを印字することで設定のテストをおこないます。 印字してみよう と トラブルシューティング を参照してください。

PostScript® プリンタのような、 プリンタ言語を使用しているプリンタには、プレインテキストを直接印字させることができません。 上にアウトラインを示し、以下の節で説明する簡単な設定方法の説明では、そのようなプリンタを設置している場合は、プリンタが認識できるファイルだけを印字の対象としているという 仮定をしています。

多くの場合、 利用者はシステムに設置されているプリンタすべてでプレインテキストが印字できることを期待しています。 印字作業をおこなうために LPD のインタフェースを利用するプログラムでも、 通常、そのような仮定を置きます。プリンタ言語を使用するプリンタを設置しており、 そのプリンタ言語で記述されたジョブと、これに加えて、 プレインテキストのジョブも印字できるようにしたいならば、上で示した簡単な設定方法に加えて、 さらなる設定をおこなうことを強くお勧めします。すなわち、自動的にプレインテキストから PostScript® (もしくは、 他のプリンタ言語) に変換するプログラムをインストールしてください。「プレインテキストのジョブを PostScript® プリンタで印字する」 で、それをどのようにおこなえばよいのかが説明されています。

日本語を印字したい場合は、プリンタ言語を使用していない「日本語プリンタ」についても、プリンタ固有のエスケープシーケンスを送る必要があります。また、漢字コードをプリン タが設定しているものに変換したりする必要があり、各プリンタ毎に、日本語用のフィルタが必要になります。

プリンタに名前を付ける

最初の (簡単な) ステップで、プリンタの名前を考えます。

プリンタには別名をいくつか付けることもできるので、機能的な名前でも風変わりな名前でもどちらを選んでもまったく 問題はありません。

少なくとも1つのプリンタには、 /etc/printcap の中で、 lp という別名を持たせるべきでしょう。この名前はデフォルトのプリンタ名になっています。 ユーザが環境変数 PRINTER を設定しておらず、かつ、LPD コマンドのコマンドラインで プリンタの名前が指定されていない場合、lp がデフォルトのプリンタ名となり、 そのプリンタに出力されます。

それから、これは共通の慣習ですが、プリンタの最後の別名には、メーカーやモデル名を含むプリンタの完全な名称をつけることに なっています。

名前と別名のいくつかを決めたら、/etc/printcap ファイルに設定します。プリンタ名は一番左のカラムから書き始めます。別名はそれぞれ縦棒によって区切られ、最後の別名の後ろにコロンを置きます。

次の例では、2 台のプリンタ (Diablo 630 ラインプリンタと Panasonic KX-P4455 PostScript® レーザライタプリンタ) が定義 されている /etc/printcap のスケルトンを記しています。

```
#
#  /etc/printcap for host rose
#
rattan|line|diablo|lp|Diablo 630 Line Printer:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:
```

この例では、最初のプリンタに rattan という名前と別名として、line、 diablo、lp そして Diablo 630 Line Printer が付けられています。別名とし て lp があるので、このプリンタはデフォルトのプリンタとなっ ています。2 番目は bamboo と名付けられ、別名として、ps と PS、S、 panasonic、Panasonic KX-P4455 PostScript v51.4 が付けられています。

ヘッダページの印字を禁止する

LPD スプーリングシステムでは、デフォルトでジョブ毎に ヘッダページを印字します。ヘッダページにはジョブを要求したユーザ名、 ジョブが送られたホスト名、そして、ジョブの名前が素晴らしい大きな文字で印字されています。残念なことに、この余分なテキストすべてが、簡単なプリンタ設定法のデバッグの際に紛れ込んできてしまいます。
このため、ヘッダページの出力を禁止しておきます。

ヘッダページの出力を禁止するには、/etc/printcap にあるプリンタのエントリに sh の項目を追加します。次に、sh を加えた /etc/printcap の例を示します。

```
#
#  /etc/printcap for host rose - no header pages anywhere
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :sh:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:
```

この書式を正しく使うための注意をしておきます。 最初の行は左端のカラムから始めます。それに続く行は字下げします。最後の行以外のすべての行は、 行末にバックスラッシュを記述します。

スプーリングディレクトリの作成

スプーラの簡単な設定の次のステップでは、 スプーリングディレクトリを作成します。プリンタに送られるジョブは、 その印字が終了するまでこのディレクトリに置かれます。また、他のたくさんのスプーラもこのディレクトリにファイルを置きます。

様々な事情によりスプーリングディレクトリは、通常、慣例 として /var/spool の下に置きます。また、スプーリングディレクトリの内容は バックアップをする必要はありません。 mkdir(1) によってディレクトリを 作るだけでスプーリングディレクトリの復旧は完了します。

スプーリングディレクトリの名前は、これも慣例ですが、 次のようにプリンタの名前と同じにします。

```
# mkdir /var/spool/printer-name
```

しかしながら、ネットワーク上に使用可能なプリンタがたく さんあるならば、LPD で印字するための専用のディレクトリにスプーリングディレクトリを置きたくなるかもしれません。例に出てきたプリンタ rattan と bamboo について、この方式を採用すると、 次のようになります。

```
# mkdir /var/spool/lpd
# mkdir /var/spool/lpd/rattan
# mkdir /var/spool/lpd/bamboo
```

各ユーザが印字するジョブのプライバシを守りたいと考えているならば、スプーリングディレクトリを保護して、これを誰からでもアクセスできないようにしたいと思うかもしれません。スプーリングディレクトリは、 daemon ユーザと daemon グループに所有され、 読み込み、書き込み、検索可能であり、他からはアクセスできないようにするべきです。例題のプリンタに対して、次のようにすることにしましょ う。

```
# chown daemon:daemon /var/spool/lpd/rattan
# chown daemon:daemon /var/spool/lpd/bamboo
# chmod 770 /var/spool/lpd/rattan
# chmod 770 /var/spool/lpd/bamboo
```

最後に、/etc/printcap ファイルで、 これらのディレクトリの位置を LPD に伝える必要があります。スプーリングディレクトリのパス名は sd 項目で指定します。

```
#
#  /etc/printcap for host rose - added spooling directories
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :sh:sd=/var/spool/lpd/rattan:
```

```
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:sd=/var/spool/lpd/bamboo:
```

プリンタ名が最初のカラムから始まっており、
そのプリンタに関して記述される他の項目は字下げされていること、
各行がバックスラッシュで終わっていることに注意してください。

sd によりスプーリングディレクトリが指定されていない場合、 スプーリングシステムは /var/spool/lpd をデフォルト値として使用します。

プリンタデバイスの特定

プリンタ機器の設定 の節では、FreeBSD でプリンタとの通信に使用されるポートおよび /dev ディレクトリ内のエントリを特定します。 そして、LPD にその情報を伝えます。 印字するジョブを受け取ると、スプーリングシステムは、 (プリンタにデータを渡す義務がある) フィルタプログラムに代わって指定されたデバイスをオープンします。

/etc/printcap ファイルで lp 項目を使って /dev エントリを記入します。

ここでの例では、rattan は 1 番目のパラレルポートに、bamboo は 6 番目のシリアルポートに接続されていることにしましょう。 このとき、/etc/printcap には 次のようになります。

```
#
#  /etc/printcap for host rose - identified what devices to use
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :sh:sd=/var/spool/lpd/rattan:\
        :lp=/dev/lpt0:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:sd=/var/spool/lpd/bamboo:\
        :lp=/dev/ttyu5:
```

/etc/printcap でプリンタの lp 項目が指定されていない場合は、 LPD はデフォルトとして /dev/lp を使用します。/dev/lp は、現在の FreeBSD には存在していません。

設置したプリンタがパラレルポートに 接続されている場合は、 「テキストフィルタのインストール」 まで読み飛ばしてください。 そうでない場合は、次節の説明に続いてください。

スプーラのための通信パラメータの設定

シリアルポートにプリンタを接続した場合、 LPD は、プリンタにデータを送信するフィルタプログラムに代わり、 通信速度やパリティ、 その他のシリアル通信パラメータを設定することができます。 このことによる利点は、

- /etc/printcap を編集するだけで、 様々な通信パラメータを試してみることができます。 フィルタプログラムを再コンパイルする必要はありません。
- スプーリングシステムで、 シリアル通信の設定が異なっているかもしれない複数のプリンタに

同じフィルタプログラムを使うことが可能になります。

次の /etc/printcap の項目で、 lp で指定された デバイスのシリアル通信パラメータを制御できます。

br#bps-rate

デバイスの通信速度を *bps-rate* に設定します。 ここで、*bps-rate* は 50, 75, 110, 134, 150, 200, 300, 600, 1200, 1800, 2400, 4800, 9600, 19200, 38400, 57600, 115200 [bit/秒] のいずれかです。

ms#stty-mode

デバイスをオープンした後にターミナルデバイスのオプションを設定します。
利用できるオプションについては stty(1) を参照してください。

lp で指定されたデバイスをオープンするとき、 LPD は ms# で指定されたデバイスの特性を設定します。 特に関係があるのは、parenb, parodd, cs5, cs6, cs7, cs8, cstopb, crtscts, ixon モードです。 これらは stty(1) のマニュアルページで説明されています。

例題のプリンタで6番目のシリアルポートに接続された プリンタの設定を追加してみましょう。 通信速度は 38400bps に設定します。 モードとして、-parenb でパリティ無し、 cs8 で 8 ビットキャラクタ、 clocal でモデム制御無し、 そして crtscts でハードウェアフロー制御を設定します。

```
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:sd=/var/spool/lpd/bamboo:\
        :lp=/dev/ttyu5:ms#-parenb cs8 clocal crtscts:
```

テキストフィルタのインストール

ここまでで、 プリンタにジョブを送るために使うテキストフィルタを LPD に設定する準備が整いました。 テキストフィルタとは、 入力フィルタとしても知られていますが、 印字するジョブがあるときに LPD が起動するプログラムです。 LPD がプリンタのためにテキストフィルタを起動するとき、 LPD はフィルタの標準入力からプリントするジョブを入力し、 フィルタの標準出力に項目 lp で指定されたプリンタデバイスを接続します。フィルタは、 標準入力からジョブを読み込み、 プリンタのための必要な変換をおこなった後、 その結果を標準出力に出力する、 これにより印字がなされることを期待されています。
テキストフィルタについての更に詳しい情報については、「フィルタはどのように機能しているか」をご覧ください。

ここでの簡単なプリンタ設定では、 プリンタにジョブを送るため、/bin/cat を実行するだけの簡単なシェルスクリプトで間に合います。 FreeBSD に標準で付属している lpf というフィルタでは、バックスペース文字を使った 下線引きの動作をおこなう文字ストリームをうまく扱うことができない プリンタのための代替処理をおこなってくれます。 もちろん、 他のどんなフィルタプログラムを使っても構いません。 フィルタ lpf については、「テキストフィルタ lpf」で詳しく説明します。

最初に、簡単なテキストフィルタであるシェルスクリプト /usr/local/libexec/if-simple を作ってみましょう。 次のテキストをお好みのテキストエディタでファイルに 書き込んでください。

```
#!/bin/sh
#
# if-simple - Simple text input filter for lpd
# Installed in /usr/local/libexec/if-simple
#
# Simply copies stdin to stdout.  Ignores all filter arguments.

/bin/cat && exit 0
exit 2
```

そして、このファイルを実行可能にします。

```
# chmod 555 /usr/local/libexec/if-simple
```

LPD にこのテキストフィルタを使うことを設定するためには、/etc/printcap に if 項目を使って指定します。これまでの /etc/printcap の例のプリンタ 2 台に、このフィルタを加えてみましょう。

```
#
#  /etc/printcap for host rose - added text filter
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :sh:sd=/var/spool/lpd/rattan:\ :lp=/dev/lpt0:\
        :if=/usr/local/libexec/if-simple:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:sd=/var/spool/lpd/bamboo:\
        :lp=/dev/ttyu5:ms#-parenb cs8 clocal crtscts:\
        :if=/usr/local/libexec/if-simple:
```

if-simple スクリプトのコピーが /usr/shared/examples/printing ディレクトリにあります。

LPD の起動

lpd(8) は lpd_enable 変数に従って /etc/rc から実行されます。この変数の デフォルト値は NO です。まだそうしていなかったならば

```
lpd_enable="YES"
```

の行を /etc/rc.conf に追加して 計算機を再起動するか、そのまま lpd(8) を 起動してください。

```
# lpd
```

簡単な LPD 設定も終わりにたどり着きました。
残念ながら、設定はこれでおしまいというわけではありません。 なぜなら、さらに、設定をテストし、すべての問題点を解決しなくてはならないからです。 設定をテストするために、何かを印字してみましょう。 LPD システムで印字をするためには、 lpr(1) コマンドを使います。このコマンドは、 印字するためのジョブを投入する働きをします。

lpr(1) コマンドを 「プリンタとの通信状況を調べる」で紹介した、あるテスト用のテキストを生成してくれる lptest(1) プログラムと一緒に使うこともできます。

簡単な *LPD* 設定のテスト

次のように入力してください。

```
# lptest 20 5 | lpr -Pprinter-name
```

ここで、*printer-name* は /etc/printcap で指定したプリンタ名 (もしくはその別名) です。デフォルトのプリンタを使用する場合は、 -P 引数を付けないで lpr(1) を打ち込んでください。もう一度述べますが、 PostScript® を期待しているプリンタをテストするならば、 lptest(1) を使う代わりに PostScript® で書かれたプログラムをプリンタに送ってください。
プログラムを送るためには、プログラムをファイルに格納して、 `lpr file` と打ち込みます。

PostScript® プリンタの場合、 送信したプログラムによる結果が得られるでしょう。 lptest(1) を使った場合は、 以下のような結果が見られるでしょう。

```
!"#$%&'()*+,-./01234
"#$%&'()*+,-./012345
#$%&'()*+,-./0123456
$%&'()*+,-./01234567
%&'()*+,-./012345678
```

更にプリンタをテストしたい場合は、 (言語ベースのプリンタのための) もっと大きなプログラムを送信するか、 引数を変えて lptest(1) を実行します。たとえば、`lptest 80 60` で、それぞれ 80 文字の行を 60 行生成します。

プリンタがうまく動かなかった場合は、次の節、「トラブルシューティング」をご覧ください。

9.4. プリンタ設定上級編

この節では、特殊な形式のファイルを印字するためのフィルタ、
ヘッダページ、ネットワーク越しのプリンタへの印字、そして、
プリンタ使用の制限や課金について説明しています。

9.4.1. フィルタ

LPD は、ネットワークプロトコル、キュー、アクセス制御などの印刷にかかわるさまざまな点を扱いますが、 実際の作業のほとんどは

フィルタによっておこなわれています。 フィルタは、プリンタと通信し、
プリンタのデバイス依存性や特殊な要求を扱うプログラムです。 簡単なプリンタ設定では、
プレインテキストのためのフィルタをインストールしました。 このプレインテキストフィルタは、
ほとんどのプリンタで機能する極めて単純なものでした (「テキストフィルタのインストール」を参照)。

しかしながら、形式変換やプリンタ課金、特定のプリンタの癖、 など をうまく利用するためには、
フィルタがどのように機能するかという ことを理解しておくべきです。これらの側面を扱うことは、
最終的には、フィルタの責任であるからです。
そして、これは悪い情報ですが、ほとんどの場合において、 あなた自身が
フィルタを供給する必要があるということです。また都合のよいことには、
たくさんのフィルタが一般的に利用できるということです。 もしフィルタがなかったとしても、
普通はフィルタを作るのは簡単です。

FreeBSD にも、プレインテキストを印字させることができる /usr/libexec/lpr/lpf というフィルタが 1 つ付いています (このフィルタはファイルに含まれるバックスペースやタブを扱います。
また、課金をすることもできますが、 できることはこれだけしかありません)。
いくつかのフィルタとフィルタの構成要素は FreeBSD Ports Collection にもあります。

この節で述べることは次の通りです。

- 「フィルタはどのように機能しているか」では、 印字の過程におけるフィルタの役割を概説します。
  この節を読むことで、LPD がフィルタを使うときに、"見えないところで"
  何が起こっているかが理解できるでしょう。このことを知っておくと、
  プリンタそれぞれに様々なフィルタをインストールしたときに
  遭遇するかもしれない問題を予期したり、 デバッグするときに役立つでしょう。
- LPD は、すべてのプリンタがデフォルトでプレインテキストを印字できることを期待しています。
  これは、プレインテキストを直接印字できない PostScript® (または他の言語対応の)
  プリンタで問題になります。「プレインテキストのジョブを PostScript® プリンタで印字する」 で、
  この問題を克服する方法について述べます。 PostScript® プリンタをお持ちの方は、
  この節をお読みになることをおすすめします。
- PostScript® は様々なプログラムのための有名な出力形式です。 PostScript®
  のコードを直接書いてしまう人すらいます。 残念ながら、PostScript® プリンタは高価です。「非 PostScript® プリンタによる PostScript® のシミュレート」節では、PostScript® データを非 *PostScript*® プリンタに受けつけさせ、印字させるために、
  どのようにしてプリンタ用のテキストフィルタをさらに変更すればよいのか、
  ということについて説明しています。PostScript® プリンタを持っていない方は、
  この節をお読みになることをおすすめします。
- 「変換フィルタ」では、 図形や組版データといった特定のファイル形式を、
  プリンタが理解できる形式へ変換する作業を自動的におこなわせる方法について述べます。
  この節を読むと、troff のデータを印字するには `lpr -t`, または、TeX DVI を印字するには `lpr -d`、
  ラスタイメージデータを印字するには `lpr -v`、
  などといったようにユーザが入力することができるように
  プリンタの設定をおこなうことができます。 この節もお読みになることをお薦めします。
- 「出力フィルタ」では、あまり使われない LPD の機能のすべて、すなわち、
  出力フィルタに関することが記述されています。ヘッダページ (「ヘッダページ」参照)
  を印字させていない場合は、 多分、この節は飛ばしても構わないでしょう。
- 「テキストフィルタ lpf」では、`lpf` についての説明が、ほぼ完全におこなわれています。これは

FreeBSD に付属するラ インプリンタ (または、 ラインプリンタのように動作するレーザプリンタ)のための、 単純なテキストフィルタです。
プレインテキストを印字したことに対して課金をおこなう方法が
至急必要な場合、もしくは、バックスペース文字を印字しようと
すると煙を発するプリンタを持っている場合は、絶対に `lpf` を検討するべきです。

以下で述べられているさまざまなスクリプトは、/usr/shared/examples/printing ディレクトリにあります。

フィルタはどのように機能しているか

既に言及したように、フィルタとは、プリンタにデータを送る際に、
デバイスに依存した部分を取り扱うために LPD によって起動される実行プログラムです。

LPD がジョブ中のファイルを印字しようとするとき、 LPD はフィルタプログラムを起動します。このとき、 フィルタの標準入力を印字するファイルに、標準出力をプリンタに、そして、標準エラー出力を エラーログファイル (/etc/printcap 内の `lf` 項目で指定されたファイル、または、 指定されていない場合は、デフォルトとして /dev/console) にセットします。

LPD が起動するフィルタと、その引数が何であるかは、 /etc/printcap ファイルの内容と、ジョブの起動時にユーザが指定した lpr(1) コマンドの引数に依存しています。たとえば、ユーザが `lpr -t` と入力した場合は、 LPD は出力先のプリンタ用の `tf` 項目で指定されている troff 用のフィルタを起動させるでしょう。 ユーザがプレインテキストの印字を指示したときは、 `if` で指定されたフィルタが起動されるでしょう (このことはほとんどの場合にあてはまります。詳細については、「出力フィルタ」をご覧ください)。

/etc/printcap で指定可能なフィルタは次の3種類があります。

- テキストフィルタ (LPD のドキュメントでは紛らわしいことに 入力フィルタと呼んでいますが)は一般のテキストの印字を扱います。これはデフォルトのフィルタと 考えてください。LPD では、すべてのプリンタに対して、
  デフォルトでプレインテキストが印字できることを期待しています。
  さらに、バックスペースやタブを正しく扱い、また、
  他の特殊な文字が入力されてもプリンタに混乱を来さないように
  するのはテキストフィルタの仕事であると考えています。
  プリンタの使用に対して課金をしなくてはならない環境にあ
  るときは、テキストフィルタが印字したページ数を数える作
  業もしなくてはなりません。この作業は、通常、印字した行 数を数え、これをプリンタが 1 ページ当たりに印字できる行 数と比較することでおこなわれます。
  テキストフィルタは、次のような引数を付けて起動されます。

  ```
  filter-name [ -c ] -w width -l length -i indent -n login -h host acct-file
  ```

  ここで、

  `-c`
  : `lpr -l` によってジョブが入力されたときに与えられます。

width

/etc/printcap で指定された `pw` (page width) 項目の値が与えられます。デフォルトは、 132 です。

length

`pl` (page length) 項目で指定された値が与えられます。 デフォルトは 66 です。

indent

`lpr -i` によって与えられた字下げの量で、 デフォルトは 0 です。

login

ファイルを印字したユーザのアカウント名が 与えられます。

host

ジョブが入力されたホスト名が 与えられます。

acct-file

`af` 項目で指定されている課金データファイル の名前が与えられます。

- 変換フィルタは、 特定のファイル形式をプリンタが紙に印字できるようなものに変換します。たとえば、 プリンタで ditroff 組版データを直接印字することはできません。 しかし、ditroff データをプリンタが消化し、印字することができる形式へ変換するために、ditroff ファイル用フィルタをインストールすることができます。 「変換フィルタ」で、これらに関するすべてについて説明します。 プリンタの課金をする必要がある場合は、変換フィルタでも印字ページを数える作業が必要となります。
  変換フィルタは次の引数をとって起動されます。

  ```
  filter-name -x pixel-width -y pixel-height -n login -h host acct-file
  ```

  ここで、*pixel-width* は、 `px` 項目で指定された値 (デフォルトは 0)、 *pixel-height* は、 `py` 項目で指定された値 (デフォルトは 0) です。

- 出力フィルタは、 テキストフィルタが指定されて おらず、かつ、ヘッダページの出力が許可されている場合にのみ使われます。 「出力フィルタ」で、これらのことについて説明します。 出力フィルタに対する引数は次の 2 つだけです。

  ```
  filter-name -w width -l length
  ```

  ここで、`-w` と `-l` は、 テキストフィルタの場合と同じです。

フィルタは、次に示す終了状態をもってプログラムを *exit* するべきです。

exit 0

フィルタがファイルを正常に印字した場合。

exit 1

フィルタはファイルの印字に失敗したが、 LPD に再度ファイルの印字を試みて欲しい場合。この終了状態で終了した場合、LPD はフィルタを再スタートします。

exit 2

フィルタはファイルの印字に失敗し、かつ、LPD に再出力を試みて欲しくない場合。この場合、LPD はそのファイルを放棄します。

FreeBSD に付属するテキストフィルタ /usr/libexec/lpr/lpf は、FORM FEED 文字が送られたときやプリンタ使用に対する課金をどのようにするかを決定するために、
ページ幅やページ長の引数を利用します。また、
課金用のエントリを作成するため、ログイン名、ホスト名、 課金ファイル名の引数を利用します。

もし、フィルタの購入を検討しているならば、LPD
と互換性があるかどうかを確認してください。もしそうならば、
上述の引数リストをサポートしていなければなりません。
一般向けの使用のためにフィルタを作成する計画をしている場合は、
同じ引数リストと終了コードをサポートしてください。

プレインテキストのジョブを **PostScript®** プリンタで印字する

コンピュータと PostScript® (または、他の言語に対応した)
プリンタをあなたしか使用しない場合は、プリンタにプレ インテキストを絶対に送らない、そして、
プリンタにプレインテキストを送りたがっている
様々なプログラムの機能を決して使わないことにしてください。そうすれば、
この節に書かれたことに心を煩わせる必要はまったくなくなります。

しかし、PostScript® とプレインテキストの両方のジョブをプリンタへ送りたいと思っている場合は、
プリンタ設定についての要求が増えるでしょう。 両者をプリンタへ送信するためには、
到着したジョブがプレインテキストであるか PostScript®
であるかを検出するテキストフィルタが必要です。 PostScript® のジョブはすべて `%!`
で始まらなければならないことになっています (他のプリンタ言語に関しては、
プリンタのドキュメントをご覧ください)。 ジョブの最初の 2 文字がこれならば、PostScript®
であることが分かります。 したがって、 ジョブのそれ以降の部分をプリンタに直接送ることができます
(訳注: PostScript® では、% 以降はコメントとして扱われるので、最初の *%!*
の行を読み捨てても問題はない)。 最初の2文字が *%!* でない場合は、 フィルタはテキストを PostScript®
に変換し、 その結果を使って印字をおこないます。

この作業をどうやってやればよいのでしょうか。

シリアルポートにプリンタを接続した場合は、 `lprps` をインストールすることをお勧めします。 `lprps` は
PostScript® 用のフィルタで、 プリンタとの双方向通信をおこないます。
このフィルタでは、プリンタからの冗長な情報を得ることで、
プリンタの状況を示すファイルが更新されていきます。 したがって、ユーザや管理者は (トナー残量少や紙詰まりといった) プリンタの状況を正確に知ることができます。しかし、 もっと重要なことは、`psif`
と呼ばれるプログラムが含まれているということです。 このプログラムは、
入力されたジョブがプレインテキストかどうかを検出し、 これを PostScript® に変換するために、`textps`
(`lprps` に付属する別のプログラム) を呼び出します。そして、このジョブをプリンタに送るために、 `lprps`
が使われます。

`lprps` は FreeBSD Ports Collection に含まれています (Ports Collection を参照してください)。
紙のサイズに合わせて print/lprps-a4 または print/lprps-letter port をインストールしてください。`lprps`
をインストールした後は、`lprps` の一部である `psif` プログラムのパス名を指定するだけです。Ports
Collection から `lprps` をインストールしたときは、 /etc/printcap の中のシリアル接続した PostScript®

プリンタのエントリに対して、次を使ってください。

```
:if=/usr/local/libexec/psif:
```

LPD にプリンタをリード・ライトモードでオープンさせるために、 `rw` 項目も指定すべきです。

パラレルポート接続の PostScript® プリンタの場合 (すなわち、 `lprps` が必要としているプリンタとの双方向通信ができない)、
テキストフィルタとして次のシェルスクリプトを使うことができます。

```
#!/bin/sh
#
#  psif - Print PostScript or plain text on a PostScript printer
#  Script version; NOT the version that comes with lprps
#  Installed in /usr/local/libexec/psif
#

IFS="" read -r first_line
first_two_chars=`expr "$first_line" : '\(..\)'`

if [ "$first_two_chars" = "%!" ]; then
   #
   #  PostScript job, print it.
   #
   echo "$first_line" && cat && printf "\004" && exit 0
   exit 2
else
   #
   #  Plain text, convert it, then print it.
   #
   ( echo "$first_line"; cat ) | /usr/local/bin/textps && printf "\004" && exit 0
   exit 2
fi
```

上記のスクリプトにおいて、`textps` はプレインテキストから PostScript® へ変換するために別にインストールしたプログラムです。 テキストから PostScript® へ変換するのには、お好みのどんなプログラムでも使うことができます。FreeBSD Ports Collection (Ports Collection を参照してください) には、`a2ps` と呼ばれるテキストから PostScript® に変換するプログラムが入っています。

非 PostScript® プリンタによる PostScript® のシミュレート

PostScript® は質の高い組版と印字をおこなうための 事実上の標準です。しかしながら、PostScript® は、高価な標準です。ありがたいことに、 Aladdin Enterprises から Ghostscript と呼ばれる、PostScript® 互換の動作をするフリーのプログラムが出されていて、 FreeBSD で動きます。 Ghostscript はほとんどの PostScript® ファイルを読むことができ、 これらの各ページを多くのブランドの非 PostScript® プリンタを含む 様々なデバイス用に変換することができます。 Ghostscript をインストールし、 プリンタ用の特別なテキストフィルタを使うことによって、 非 PostScript® プリンタをあたかも本物の PostScript® プリンタであるかのように動作させることができます。

Ghostscript は FreeBSD Ports Collection に入っています。複数のバージョンがありますが、最も良く使われているバージョンは print/ghostscript-gpl です。

PostScript® プリンタをシミュレートさせる場合は、テキストフィルタに PostScript® ファイルを印字しようとしているかどうかを検出させます。PostScript® ファイルでない場合は、フィルタはそのファイルを直接プリンタに送ります (訳注: テキストファイルを直接印字できない場合は、もちろん、変換フィルタを通す必要があります)。PostScript® の場合は、まず、Ghostscript を使い、ファイルをそのプリンタが理解できる形式へ変換します。

次の例のスクリプトは、Hewlett Packard DeskJet 500 プリンタ用のテキストフィルタです。他のプリンタで用いるときは、`-sDEVICE` 引数を `gs` (Ghostscript) コマンドに変えてください (`gs -h` と入力すると、現在インストールされている Ghostscript でサポートされているデバイスのリストが得られます)。

```
#!/bin/sh
#
#  ifhp - Print Ghostscript-simulated PostScript on a DeskJet 500
#  Installed in /usr/local/libexec/ifhp

#
#  Treat LF as CR+LF (to avoid the "staircase effect" on HP/PCL
#  printers):
#
printf "\033&k2G" || exit 2

#
#  Read first two characters of the file
#
IFS="" read -r first_line
first_two_chars=`expr "$first_line" : '\(..\)'`

if [ "$first_two_chars" = "%!" ]; then
    #
    #  It is PostScript; use Ghostscript to scan-convert and print it.
    #
    /usr/local/bin/gs -dSAFER -dNOPAUSE -q -sDEVICE=djet500 \
      -sOutputFile=- - && exit 0

else
    #
    #  Plain text or HP/PCL, so just print it directly; print a form feed
    #  at the end to eject the last page.
    #
    echo "$first_line" && cat && printf "\033&l0H" &&
    exit 0
fi

exit 2
```

最後に、if 項目を通して、LPD にこのフィルタを教えてやる必要があります。

```
:if=/usr/local/libexec/ifhp:
```

これでおしまいです。lpr plain.text とか lpr whatever.ps と入力してみましょう。どちらも正常に印字されるはずです。

日本語を印字する場合は、 日本語対応の Ghostscript が必要です。日本語対応版の Ghostscript も Ports Collection に入っています。

変換フィルタ

「プリンタ設定導入編」 に書かれた簡単な設定が完了したら、最初に、やってみたいと思うことは、多分 (プレイン ASCII テキストに加えて) 好みのファイル形式のための変換フィルタをインストールすることでしょう。

なぜ、変換フィルタをインストールするのか?

変換フィルタによって、 様々な種類のファイルを印字することが簡単になります。たとえば、TeX 組版システムでたくさんの仕事をしたと仮定しましょう。 そして、PostScript® プリンタが接続されているとします。 すると、TeX で DVI ファイルを作成する度に、DVI ファイルを印字するために、これを PostScript® ファイルに変換する必要があります。 このコマンドは次のようになるでしょう。

```
% dvips seaweed-analysis.dvi
% lpr seaweed-analysis.ps
```

DVI ファイル用の変換フィルタがインストールしてあると、 LPD に変換を肩代わりさせることで毎回毎回おこなわなければならなかった面倒な変換作業を省くことができます。 つまり、DVI を生成したら、次のようなコマンドを入力するだけで、これが印字されます。

```
% lpr -d seaweed-analysis.dvi
```

LPD に DVI ファイルの変換をさせるためには、 -d オプション を指定します。変換オプションのリストは「整形と変換に関するオプション」 に載せてあります。

変化のオプションのそれぞれをプリンタに サポートさせるためには、 変換フィルタをインストールし、そのパス名を /etc/printcap の中で指定しなくてはなりません。変換フィルタは、プレインテキストを印字する代わりに、フィルタはファイルを
プリンタが理解できる形式に変換するところを除けば、
「プリンタの簡単な設定」で説明したテキストファイル (「テキストフィルタのインストール」を見て下さい) に似ています。

どの変換フィルタをインストールすべきか?

使いたいと思う変換フィルタをインストールすべきです。 DVI のデータを頻繁に印字するならば、DVI 変換フィルタ をインストールするのが適切でしょう。印字しなくてはなら ない troff を大量に抱えている場合は、多分、 troff フィルタが欲しくなるはずです。

次の表は、LPD で動作するフィルタと、 /etc/printcap ファイルでのエントリする項目、そして、 lpr コマンドで呼び出す方法をまとめたものです。

ファイル形式	**/etc/printcap**項目	**lpr** オプション
cifplot	cf	-c
DVI	df	-d
plot	gf	-g
ditroff	nf	-n
FORTRAN text	rf	-f
troff	tf	-f
raster	vf	-v
プレインテキスト	if	なし、-p、または -l

先の例のように、lpr -d を使うためには、出力先のプリンタの /etc/printcap 内のエントリで、 df 項目が必要であることが分かります。

反論はあるかも知れませんが、FORTRAN テキストや plot のような形式は、多分、廃れてていくでしょう。
あなたのサイトで、自前のフィルタをインストールするだけで、
プリントオプションのいくつか、あるいは、 全部に新しい意味を与えることができます。たとえば、Printerleaf ファイル (Interleaf デスクトップパブリッシングプログラムによるファイル) を直接印字したいとします。 そして、Printerleaf 用の変換フィルタを gf 項目で指定したパスにインストールすれば、lpr -g の意味は "Printerleaf ファイルを印字する" 意味だとユーザに教えることができます。

変換フィルタのインストール

変換フィルタは FreeBSD の基本システムのインストールとは別にインストールするプログラムなので、変換フィルタは、 /usr/local ディレクトリの下に置くべきでしょう。 フィルタは LPD だけが実行する特別なプログラム、 すなわち、一般ユーザが実行する必要すらないプログラムなので、/usr/local/libexec ディレクトリに置くのが普通です。

変換フィルタを使用可能にするためには、 /etc/printcap の目的のプリンタの適切な項目にフィルタがあるパス名を指定します。

DVI 変換フィルタをプリンタ bamboo のエントリに加えてみましょう。プリンタ bamboo の df 項目を新たに加えた /etc/printcap ファイルの例を以下に再掲します。

```
#
#  /etc/printcap for host rose - added df filter for bamboo
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :sh:sd=/var/spool/lpd/rattan:\
        :lp=/dev/lpt0:\
        :if=/usr/local/libexec/if-simple:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
```

```
        :sh:sd=/var/spool/lpd/bamboo:\
        :lp=/dev/ttyu5:ms#-parenb cs8 clocal crtscts:rw:\
        :if=/usr/local/libexec/psif:\
        :df=/usr/local/libexec/psdf:
```

DVI フィルタは /usr/local/libexec/psdf という名前のシェルスクリプトです。このスクリプトは次のようになっています。

```
#!/bin/sh
#
#  psdf - DVI to PostScript printer filter
#  Installed in /usr/local/libexec/psdf
#
#  Invoked by lpd when user runs lpr -d
#
exec /usr/local/bin/dvips -f | /usr/local/libexec/lprps "$@"
```

このスクリプトでは、dvips をフィルタモード (引数 -f) で、標準入力上で起動しています。標準入力は印字するジョブです。それから、PostScript® プリンタ用フィルタ lprps (これについては「プレインテキストのジョブを PostScript® プリンタで印字する」 を参照してください) を LPD に与えられた引数を付けて起動します。 lprps はこれらの引数を印字されたページ分の課金をおこなうために使われます。

変換フィルタのその他の例

変換フィルタのインストールには決まったステップがないので、
この節では、例をもっと挙げることにします。 これを自分でフィルタを作る際のガイドにしてください。
適当な例があったら、それをそのまま使ってください。

次のスクリプト例は、Hewlett Packard LaserJet III-Si のための、raster (ええと・・実は、GIF ファイル) 用の変換フィルタです。

```
#!/bin/sh
#
#  hpvf - Convert GIF files into HP/PCL, then print
#  Installed in /usr/local/libexec/hpvf

PATH=/usr/X11R6/bin:$PATH; export PATH

giftopnm | ppmtopgm | pgmtopbm | pbmtolj -resolution 300 \
    && exit 0 \
    || exit 2
```

ここでは、GIF ファイルから PNM (portable anymap) 形式に変換し、次に PGM (portable graymap) 形式に変換してから、 LaserJet/PCL-互換データに変換しています。

上記のフィルタを使うプリンタのためのエントリを付け加えた /etc/printcap ファイルは次のようになります。

```
#
#  /etc/printcap for host orchid
#
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
        :lp=/dev/lpt0:sh:sd=/var/spool/lpd/teak:mx#0:\
        :if=/usr/local/libexec/hpif:\
        :vf=/usr/local/libexec/hpvf:
```

次のスクリプトは、PostScript® プリンタ bamboo のための groff 組版システムの troff データのための変換フィルタです。

```
#!/bin/sh
#
#  pstf - Convert groff's troff data into PS, then print.
#  Installed in /usr/local/libexec/pstf
#
exec grops | /usr/local/libexec/lprps "$@"
```

上記のスクリプトではプリンタとの通信をおこなうため、 lprps をまた利用しています。プリンタがパラレルポートに接続されている場合は、代わりに、 次のスクリプトを使うかもしれません。

```
#!/bin/sh
#
#  pstf - Convert groff's troff data into PS, then print.
#  Installed in /usr/local/libexec/pstf
#
exec grops
```

これで完成しました。次に、フィルタを使用可能にするため に /etc/printcap に加える必要があるエントリを示します。

```
:tf=/usr/local/libexec/pstf:
```

次の例をみたら、FORTRAN のベテランは赤面するかもしれません。 この FORTRAN テキストフィルタは、 プレインテキストを直接印字できるすべてのプリンタで利用できます。このフィルタをプリンタ teak にインストールすることにしましょう。

```
#!/bin/sh
#
# hprf - FORTRAN text filter for LaserJet 3si:
# Installed in /usr/local/libexec/hprf
#

printf "\033&k2G" && fpr && printf "\033&l0H" && exit 0
exit 2
```

そして、このフィルタを使用可能にするため、以下の行を /etc/printcap のプリンタ teak のエントリに加えます。

```
:rf=/usr/local/libexec/hprf:
```

これが最後の、そして、若干複雑な例です。前に紹介した LaserJet プリンタ teak に、DVI フィルタを加える ことにしましょう。最初に、 簡単な部分をおこないます。すなわち、DVI フィルタの位置を /etc/printcap に書き加えます。

```
:df=/usr/local/libexec/hpdf:
```

さて、難しい部分であるフィルタの作成をおこないます。 このために、DVI から LaserJet/PCL への変換プログラムが必要です。FreeBSD の Ports Collection (Ports Collection を参照してください) には、それがあります。 dvi2xx というのがその port の名前です。 これをインストールすると、必要なプログラム dvilj2p が使えます。このプログラムは DVI を LaserJet IIp、LaserJet III、そして LaserJet 2000 の互換コードへ変換してくれます。

dvilj2p はフィルタ hpdf を極めて複雑にしています。 なぜなら、dvilj2p は標準入力からデータを読み込むことができないからです。
このプログラムを働かせるためには、ファイル名が必要です。 もっと悪いことに、ファイル名は .dvi で終わっている必要があり、標準入力の代わりに、 /dev/fd/0 を使うのは問題があります。 この問題は、(.dvi で終わる) 一時的なファイル名から/dev/fd/0 に (シンボリックな) リンクを張ることで回避することができます。これで、 dvilj2p に強制的に標準入力からデータを読み込ませることができます。

もう1つの問題は、一時的なリンクを張るために /tmp ディレクトリを使うことができないという事実です。 シンボリックリンクはユーザ、グループが bin であるユーザに所有されています。フィルタはユーザ daemon として起動します。そして、 /tmp ディレクトリはスティッキービットが立っています。 フィルタはリンクを作ることができます。しかし、リンクは別のユーザに所有されているため、
作業が終了したとき、このリンクを削除することができません。

その代わりに、シンボリックリンクは現在の作業ディレクトリ、 すなわち、スプーリングディレクトリ (/etc/printcap の sd 項目で指定する) に作ることにします。 フィルタが作業するにはここの場所は完璧な場所で、なぜなら、 特に、スプーリングディレクトリのディスクの空き容量は (ときどき) /tmp ディレクトリよりもたくさんあるからです。

以下に示すのが最後のフィルタです。

```
#!/bin/sh
#
#  hpdf - Print DVI data on HP/PCL printer
#  Installed in /usr/local/libexec/hpdf

PATH=/usr/local/bin:$PATH; export PATH

#
#  Define a function to clean up our temporary files.  These exist
```

```
#  in the current directory, which will be the spooling directory
#  for the printer.
#
cleanup() {
   rm -f hpdf$$.dvi
}

#
#  Define a function to handle fatal errors: print the given message
#  and exit 2.  Exiting with 2 tells LPD to do not try to reprint the
#  job.
#
fatal() {
    echo "$@" 1>&2
    cleanup
    exit 2
}

#
#  If user removes the job, LPD will send SIGINT, so trap SIGINT
#  (and a few other signals) to clean up after ourselves.
#
trap cleanup 1 2 15

#
#  Make sure we are not colliding with any existing files.
#
cleanup

#
#  Link the DVI input file to standard input (the file to print).
#
ln -s /dev/fd/0 hpdf$$.dvi || fatal "Cannot symlink /dev/fd/0"

#
#  Make LF = CR+LF
#
printf "\033&k2G" || fatal "Cannot initialize printer"

#
#  Convert and print.  Return value from dvilj2p does not seem to be
#  reliable, so we ignore it.

#
dvilj2p -M1 -q -e- dfhp$$.dvi

#
#  Clean up and exit
#
cleanup
```

```
exit 0
```

自動変換: その他の変換フィルタ

ここまでに述べてきたフィルタによって、 印字環境の能率が上がったことと思います。しかし、
これはどのフィルタを使うかを (lpr(1) のコマンドライン上で)
ユーザが指定しなくてはならないという代価を支払って実現されています。
コンピュータの事情にあまり詳しくないユーザにとって、
フィルタのオプションを指定させられるということは
いらいらさせられるものになるでしょう。更に悪いことに、
間違ったフィルタオプションを指定されると、
間違った形式のファイルがそのフィルタに適用されることになり、
その結果、何百枚もの紙を吐き出すことになるかもしれません。

そのような結果になるならば、 変換フィルタをインストールするよりもむしろ、 テキストフィルタ
(これがデフォルトフィルタなので) に印字するよう要求されたファイルの形式を検出させ、自動的に、
適切な変換フィルタを起動するようにしたいと思うかもしれません。 ここでは file
コマンドのようなツールを役立たせることができます。 もちろん、いくつかの
ファイル形式の違いを見分けることは難しいことでしょう。
そして、もちろん、それらのファイルに対しては、
変換フィルタを提供するだけで済ますこともできるのです。

FreeBSD Ports Collection には、apsfilter (print/apsfilter)
と呼ばれる自動変換をおこなうテキストフィルタがあります。 このフィルタは
プレインテキスト、PostScript®, DVI
など、ほとんどすべてのファイル形式を検出し、適当な変換をおこなった後、
データを印字することができます。

出力フィルタ

LPD スプーリングシステムでは、 ここまでにまだ取り上げていないフィルタ形式、
出力フィルタをサポートしています。出力フィルタは、 テキストフィルタのように、
プレインテキストのみを印字するために意図されたものですが、
非常に簡単化されています。テキストフィルタを用いずに、
出力フィルタを使っている場合は、次のようになります。

- LPD はジョブ中の各ファイルに一度ではなく、
  ジョブ全体に対して一度だけ出力フィルタを起動します。
- LPD は出力フィルタに対し、 ジョブ中のファイルの先頭や末尾を特定するための対策を
  一切おこなっていません。
- LPD はユーザのログイン名やホスト名をフィルタに渡しません。
  したがって、課金の処理をおこなうことは考えていません。 実際、出力フィルタには、以下
  2つの引数しか与えられません。

  ```
  filter-name -wwidth -llength
  ```

  ここで、*width* は対象となるプリンタの pw 項目、 *length* は pl 項目に指定された数です。

出力フィルタの簡便さに誘惑されてはいけません。もし、
ジョブ中のそれぞれのファイルに別のページ番号を付加しようとしても、

出力フィルタはうまく動作しないでしょう。そのような動作を期待しているならば、(入力フィルタとしても知られている)テキストフィルタを使ってください。詳しくは、「テキストフィルタのインストール」をご覧ください。
さらに、出力フィルタは、実のところ、もっと複雑になっています。まず、特殊なフラグ文字を検出するために、フィルタに送られてくるバイトストリームを検査する必要があります。また、LPD に代わって、自分自身にシグナルを送らなければなりません。

しかしながら、ヘッダページの印字をおこないたくて、エスケープシーケンスやヘッダページを印字できるようにするその他の初期化文字列を送信する必要がある場合は、出力ファイルが必要です。(しかし、ヘッダページを要求したユーザに対して課金しようとするのもまた無駄なことです。LPD は出力フィルタにユーザやホストの情報を渡しません)。

1 台のプリンタに対し、LPD では出力フィルタとテキストやその他のフィルタを両方使うことができます。このような場合、LPD はヘッダページ(「ヘッダページ」を参照してください)だけを印字させるために、出力フィルタを起動させます。それから LPD では、出力フィルタに 2 バイトの文字(ASCII 031 の次に ASCII 001)を送ることで、出力フィルタが自分自身を停止することを期待しています。2 バイト (031, 001) が出力フィルタに送られたとき、出力フィルタは自分自身にシグナル SIGSTOP を送ることによって停止するはずです。LPD がその他のフィルタを動かし終わると、出力フィルタにシグナル SIGCONT を送って、出力フィルタを再起動します。

出力フィルタがあり、テキストフィルタがない場合、LPD はプレインテキストジョブを扱う場合に、出力フィルタを使います。前述したように、出力フィルタでは、ジョブ中の各ファイルの間に FORM FEED 文字や紙を送る他の文字を入れることはしません。この動作は多分、あなたが求めているものとは異なっているでしょう。
ほとんどの場合において、テキストフィルタが必要なはずです。

プログラム lpf は、テキストフィルタの項で既に紹介しましたが、出力フィルタとしても動作させることができます。もし、簡便で極悪な出力フィルタが必要で、かつ、バイトストリームを検査したりシグナルを送るコードを書きたくないときには、lpf をお試しください。あるいは、プリントが要求する初期化コードを送るために、lpf をシェルスクリプトに包んで使うこともできます。

テキストフィルタ lpf

プログラム /usr/libexec/lpr/lpf は、FreeBSD の バイナリ配布に付属しているテキストフィルタ(入力フィルタ)で、出力を字下げしたり(lpr -i でジョブが入力されたとき)、文字を未処理のままプリンタに送ったり(lpr -l でジョブが入力されたとき)、ジョブ中のバックスペースやタブの印字位置を調節したり、印字したページに対して課金したりすることができます。また、このフィルタは出力フィルタとしても動作させることができます。

lpf フィルタは多くの印字環境において使用することに適しています。このフィルタには、プリンタに初期化文字列を送る機能はありませんが、必要とされる初期化をおこない、それから lpf を実行させるためのシェルスクリプトを作成するのはたやすいことです。

lpf に対して、印字ページへの課金を正確におこなわせるためには、/etc/printcap ファイルの中の pw と

pl の項目に正確な値を入れておく必要があります。これらの値は、
どのくらいの量のテキストがページにフィットするか、また、
ユーザのジョブが何ページあるのかを調べるために使われます。
プリンタの課金についての詳しい情報については、「プリンタの利用に対する課金」をご覧ください。

9.4.2. ヘッダページ

あなたが管理するシステムのユーザが たくさんおり、
ユーザ全員が様々なプリンタを使用する場合、多分、 必要悪であるヘッダページを
印字させることを検討したいと思うかもしれません。

ヘッダページは、バナー とか バーストページ としても知られていますが、
出力されたジョブが誰によるものなのかを特定させる働きがあります。 印字結果の山の中において、
ユーザのジョブによって印字された本物のドキュメント部分よりも際立たせるために、
ヘッダページは、通常、多分、縁が装飾されている大きな太文字で印字されます。 ヘッダページにより、
ユーザは自分が出したジョブがどこにあるのかをすばやく見つけることができます。
ヘッダページの欠点は、明らかに、すべてのジョブに対して、 紙が 1
枚余分に印字されるということです。 この紙の有効期間は短く、2 ～ 3 分も続きません。最終的に、
これらの紙は再利用紙入れの中かくずの山に入れられることでしょう
(ヘッダページはジョブ中の各ファイル毎に印字されるのではなく、
ジョブ毎に印字されるということに注意してください。したがって、
紙の消費はそれほどひどくはないかもしれません)。

もし、 プリンタがプレインテキストを直接印字できるならば、LPD
システムは印字物に対して自動的にヘッダページを付けることができます。 PostScript®
プリンタを使っている場合は、
ヘッダページを生成する外部プログラムが必要になります。これについては、 「PostScript®
プリンタでのヘッダページ」をご覧ください。

ヘッダページの印字を許可する

「プリンタ設定導入編 」節では、/etc/printcap ファイルの sh ("suppress header" : "
ヘッダを供給しない" という意味) を指定して、 ヘッダページの印字を止めていました。
プリンタでのヘッダページの印字を許可するには、 sh 項目を取り除くだけでよいのです。

とても簡単そうに見えるけど、本当かな?

それは本当です。 プリンタに初期化文字列を送るための
出力フィルタを用意しなくてはならないかもしれません。次に、Hewlett Packard PCL
互換プリンタの例を挙げます。

```
#!/bin/sh
#
#  hpof - Output filter for Hewlett Packard PCL-compatible printers
#  Installed in /usr/local/libexec/hpof

printf "\033&k2G" || exit 2
exec /usr/libexec/lpr/lpf
```

of 項目に出力フィルタのパス名を指定してください。 詳細については、「出力フィルタ」節

をご覧ください。

次に、以前紹介したプリンタ teak のための /etc/printcap ファイルの例を示します。ここでは、ヘッダページの印字を許可し、上記の出力フィルタを追加しました。

```
#
#  /etc/printcap for host orchid
#
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
        :lp=/dev/lpt0:sd=/var/spool/lpd/teak:mx#0:\
        :if=/usr/local/libexec/hpif:\
        :vf=/usr/local/libexec/hpvf:\
        :of=/usr/local/libexec/hpof:
```

さて、ユーザが teak からジョブを印字させたとき、それぞれのジョブ毎にヘッダページが印字されます。
もし、ユーザが印字物を探すのに時間を費やしたいと思うなら、 lpr -h によってジョブを入力することで、 ヘッダページの印字を止めることができます。 これ以外の lpr(1) のオプションについては、 「ヘッダページ用オプション」節をご覧ください。

> LPD では、ヘッダページの最後に、 FORM FEED 文字が印字されます。
> プリンタに紙排出をさせるために、別な文字、
> もしくは、別な文字列が利用されている場合は、 /etc/printcap 中の ff 項目で指定することができます。

ヘッダページを制御する

ヘッダページの印字が許可されていると、LPD は 長いヘッダを作ります。これには、紙全面に大きな文字でユーザ名、ホスト名、
ジョブ名が書かれています。次に、このヘッダページの例を示 します (kelly がジョブ名 "outline" を rose というホストから印字 された場合)。

```
      k                  ll       ll
      k                   l        l
      k                   l        l
      k   k     eeee      l        l     y    y
      k  k     e    e     l        l     y    y
      k k      eeeeee     l        l     y    y
      kk k     e          l        l     y    y
      k   k    e    e     l        l     y   yy
      k    k    eeee     lll      lll     yyy y
                                            y
                                        y    y
                                         yyyy

                                 ll
                        t         l        i
                        t         l
       oooo    u    u   ttttt      l       ii     n nnn     eeee
```

```
      o    o   u    u     t         l        i     nn   n   e    e
      o    o   u    u     t         l        i     n    n   eeeeee
      o    o   u    u     t         l        i     n    n   e
      o    o   u   uu     t  t      l        i     n    n   e    e
       oooo     uuu u      tt      lll      iii    n    n    eeee

      r rrr      oooo     ssss     eeee
      rr   r    o    o   s    s   e    e
      r         o    o    ss      eeeeee
      r         o    o      ss    e
      r         o    o   s    s   e    e
      r          oooo     ssss     eeee

                                          Job:  outline
                                          Date: Sun Sep 17 11:04:58 1995
```

LPD はこのテキストの終わりに FORM FEED 文字を加えますので、ジョブは新しいページから開始されます (ただし、 /etc/printcap で出力先のプリンタのエントリに sf (suppress form feeds) が指定されているときはこ の限りではありません)。

お望みならば、LPD に短いヘッダページを出力させることもできます。 この場合は、 /etc/printcap ファイルの中で sb (short banner) を指定してください。 ヘッダページは次のようになります。

```
rose:kelly  Job: outline  Date: Sun Sep 17 11:07:51 1995
```

デフォルトでは、LPD はヘッダページを最初に印字し、次にジョブの印字をおこないます。この順番を逆にするときは、 /etc/printcap で hl (header last) を指定してください。

ヘッダページに対する課金

LPD に備わっているヘッダページ出力機能を使うと、入力されたジョブに対して課金をおこなうことができても、 ヘッダページは無料で提供しなくてはならない、 という特有のやり方を強要されます。

なぜでしょうか。

出力フィルタは単なる外部プログラムなので、 課金をするための制御をおこなうとすれば、それはヘッダページを印字するときですが、出力フィルタには、 ユーザ名とホスト名の情報や課金情報を格納するファイルがどれな
のかということが知らされません。それゆえ、出力ファイルには、
誰にプリンタ利用の課金をおこなえばよいのかが分からないのです。
テキストフィルタやその他の変換フィルタ (これらのフィルタはユーザやホストの情報が知らされます) が出力ページの枚数に "1 ページ分水増しする" だけでは十分ではありません。 なぜなら、ユーザは lpr -h に よってヘッダページの出力を止めることができるからです。 やみくもに 1 ページを水増しすると、印字されてもいないヘッダページに対する 料金をとることになります。基本的に、lpr -h は環境に優しい心を持つユーザに好まれるオプションですが、
これを使うように奨励することもできません。

各々のフィルタに独自のヘッダページを生成させる (その結果、ヘッダページに課金することができる)

という方法でも十分であるとはいえません。この場合、LPD はフィルタに -h の情報を送りませんので、lpr -h によってヘッダページを印字しないオプションを選択したとしても、依然としてヘッダページは印字され、その分の課金がおこなわれてしまいます。

では、どのような選択肢があるのでしょうか。

ヘッダページへの課金に関しては、次のことができます。

- LPD のやり方を受け入れ、ヘッダページは無料とする。
- LPRng などの LPD の代替品をインストールする。LPD と入れ替えが可能な他のスプーリングソフトウェアに関しては、標準スプーラの代替品をご覧ください。
- スマートな出力フィルタを作成する。通常、出力フィルタはプリンタを初期化するか、単純な文字列変換をする程度の働きしかしません。(テキスト (入力) フィルタがない場合) 出力フィルタはヘッダページとプレインテキストの印字をおこなうのに適しています。プレインテキストを印字するためのテキストフィルタがない場合、LPD はヘッダページを印字するためだけの目的で出力フィルタを起動します。そして、LPD が生成するヘッダページのテキストを解析することにより、出力フィルタはヘッダページに課金するために必要なユーザ名とホスト名を取得することができます。この方式の唯一の問題点は、出力フィルタは課金情報を格納するデータファイルの名前を知ることができないということです (af 項目で指定されたファイル名は出力ファイルに渡されません)。しかし、既知の名前の課金データファイルを使うのならば、その名前を出力フィルタのプログラム中に埋め込むことができます。解析の手順を簡単にするためには、/etc/printcap で sh 項目 (短いヘッダを指定) を使うとよいでしょう。そしてまた、ここまでの方法は少なからぬトラブルを生じさせるかもしれません。そうなれば、もちろんユーザはヘッダページを無料で提供してくれる気前のよいシステム管理者に感謝することでしょう。

PostScript® プリンタでのヘッダページ

これまでに述べたように、LPD ではプレインテキストのヘッダページをたくさんのプリンタに合うように生成することができます。残念ながら、PostScript® プリンタは、プレインテキストを直接印字することができません。ですから、LPD のヘッダページ機能はまったく、あるいはほとんどの場合、役に立ちません。

ヘッダページを出力するための自明な方法の1つに、すべての変換フィルタとテキストフィルタにヘッダページを生成させる方法があります。フィルタは、適切なヘッダページを生成するために、ユーザ名とホスト名の引数を使うべきです。この方法の欠点は、いつでも、lpr -h によってジョブが入力された場合でさえも、ヘッダページが印字されるということです。

この方法で試してみましょう。次のスクリプトは、3 つの引数 (ユーザのログイン名、ホスト名、ジョブ名) をとり、簡単な PostScript® 用 のヘッダページを生成します。

```
#!/bin/sh
#
```

```
#  make-ps-header - make a PostScript header page on stdout
#  Installed in /usr/local/libexec/make-ps-header
#

#
#  These are PostScript units (72 to the inch).  Modify for A4 or
#  whatever size paper you are using:
#
page_width=612
page_height=792
border=72

#
#  Check arguments
#
if [ $# -ne 3 ]; then
    echo "Usage: `basename $0` <user> <host> <job>" 1>&2
    exit 1
fi

#
#  Save these, mostly for readability in the PostScript, below.
#
user=$1
host=$2
job=$3
date=`date`

#
#  Send the PostScript code to stdout.
#
exec cat <<EOF
%!PS

%
%  Make sure we do not interfere with user's job that will follow
%
save

%
%  Make a thick, unpleasant border around the edge of the paper.
%
$border $border moveto
$page_width $border 2 mul sub 0 rlineto
0 $page_height $border 2 mul sub rlineto
currentscreen 3 -1 roll pop 100 3 1 roll setscreen
$border 2 mul $page_width sub 0 rlineto closepath
0.8 setgray 10 setlinewidth stroke 0 setgray

%
%  Display user's login name, nice and large and prominent
```

```
%
/Helvetica-Bold findfont 64 scalefont setfont
$page_width ($user) stringwidth pop sub 2 div $page_height 200 sub moveto
($user) show

%
%  Now show the boring particulars
%
/Helvetica findfont 14 scalefont setfont
/y 200 def
[ (Job:) (Host:) (Date:) ] {
200 y moveto show /y y 18 sub def
} forall

/Helvetica-Bold findfont 14 scalefont setfont
/y 200 def
[ ($job) ($host) ($date) ] {
        270 y moveto show /y y 18 sub def
} forall

%
%  That is it
%
restore
showpage
EOF
```

そして、変換フィルタやテキストフィルタがそれぞれ、ヘッダページが出力され、それから、このドキュメントの始めのほうで紹介したヘッダページを印字するように変更したものを示します。

最初にこのスクリプトを起動することで、ユーザのジョブの印字をおこないます。次に、DVI 変換フィルタを、

```
#!/bin/sh
#
#  psdf - DVI to PostScript printer filter
#  Installed in /usr/local/libexec/psdf
#
#  Invoked by lpd when user runs lpr -d
#

orig_args="$@"

fail() {
    echo "$@" 1>&2
    exit 2
}

while getopts "x:y:n:h:" option; do
    case $option in
        x|y)  ;; # Ignore
```

```
        n)    login=$OPTARG ;;
        h)    host=$OPTARG ;;
        *)    echo "LPD started `basename $0` wrong." 1>&2
              exit 2
              ;;
    esac
done

[ "$login" ] || fail "No login name"
[ "$host" ] || fail "No host name"

( /usr/local/libexec/make-ps-header $login $host "DVI File"
  /usr/local/bin/dvips -f ) | eval /usr/local/libexec/lprps $orig_args
```

このフィルタがユーザ名やホスト名を決定するために
引数リストをどのように解析しなくてはならないかという点に注意してください。
この解析方法は他の変換フィルタに対しても同様です。 しかしながら、テキストフィルタについては、
引数の設定が少し異なっています (これについては、「フィルタはどのように機能しているか」
をご覧ください)。

前述の通り、上記の手法は、極めて単純なのにも関らず、 lpr で "ヘッダページを印字しない" オプション
(-h オプション) が使えなくなっています。 ユーザが森林資源を (あるいは、
ヘッダページが課金されているならば、その僅かな金額を)、 節約したいと望んでいる場合でも、
すべてのフィルタがすべてのジョブ毎にヘッダページを印字
することになっているので、節約することはできません。

ジョブ毎に印字されるヘッダページを
ユーザが抑制できるようにするためには、「ヘッダページに対する課金」で紹介したトリックを
使う必要があります。すなわち、LPD が生成するヘッダページの解析をおこない、PostScript®
版のヘッダページを出力させる出力フィルタを作るのです。 この場合、ユーザが lpr -h
でジョブを入力すると、 LPD はヘッダページを生成しなくなり、また、
出力フィルタも起動されません。そうでないならば、 作成した出力フィルタが LPD
からのテキストを読み込み、ヘッダページを印字する適当な PostScript®
のコードがプリンタに送られるでしょう。

PostScript® プリンタがシリアルポートに接続されている場合、 出力フィルタとして lprps を、
上記の動作をおこなうものとして psof を使うことができます。ただし、psof
はヘッダページに対して課金をおこないませんので注意してください。

9.4.3. リモートプリンタからの出力

FreeBSD では、ネットワーク越しの印字、すなわち、
ジョブをリモートプリンタに送ることをサポートしています。
リモートプリンタからの出力をするには、一般に、 次の 2 つを参照してください。

- リモートホストに接続されたプリンタにアクセスする方法。 プリンタがあるホストのシリアル、
  または、パラレルインタフェースに接続されている場合、
  ネットワーク上の他のホストからこのプリンタにアクセスできるように LPD
  を設定します。「リモートホストに 接続されたプリンタ」 でどのようにするかを説明します。
- ネットワークに直接接続されているプリンタにアクセスする方法。

プリンタに、旧来のシリアル、または、パラレルインタフェースに加えて
(もしくは、これらに代わって)ネットワーク用のインタフェースがある場合。
そのようなプリンタは次のように動作するでしょう。

- そのプリンタがLPDのプロトコルを理解でき、リモートホストからのジョブをキューに入れることさえできる場合。この場合、プリンタは、LPDが起動している一般のホストのように振る舞います。そのようなプリンタを設定するために、「リモートホストに接続されたプリンタ」と同様の手順をおこなってください。
- そのプリンタが、データストリームによるネットワーク接続をサポートしている場合。この場合、ネットワーク上の1つのホストとしてプリンタを"接続"します。このホストは、ジョブをスプーリングする責任を負い、
スプーリングされたジョブはプリンタに送られます。
そのようなプリンタをインストールするためのいくつかの提案が
「ネットワークにおけるデータストリームの インタフェースを持つプリンタ」にあります。

リモートホストに接続されたプリンタ

LPDスプーリングシステムではLPD(またはLPD互換のシステム)が起動している他のホストへジョブを送る機能が始めからサポートされています。この機能により、あるホストに接続されたプリンタへ、他のホストからアクセスできるようになります。また、LPDプロトコルを理解するネットワークインタフェースを持ったプリンタに対しても、この機能は働きます。

リモートプリンタへの出力を許可するためには、最初に、あるホスト(これを、プリンタホストと呼びます)にプリンタを接続します。そして、「プリンタ設定導入編」に書かれた簡単なプリンタの設定をおこなってください。必要ならば、「プリンタ設定上級編」にある、更に進んだ設定をおこなってください。そして、
そのプリンタをテストしてうまく動作することを確認し、LPD
に許可した機能がうまく働くかどうかを見てください。さらに ローカルホストが プリンタホストの LPDサービスの使用を許可されているか確認して下さい(「リモートホストからの利用を制限する 」参照)。

LPD互換のネットワークインタフェースを持つプリンタを使用している場合は、そのプリンタ自身が以下で説明するプリンタホストになります。そして、プリンタ名とは、そのプリンタに設定した名前のことを指します。これについては、プリンタ、および(または)、プリンタのネットワークインタフェースに付属するドキュメントを参照してください。

ヒューレット・パッカード社のLaserjetシリーズを使用している場合には、プリンタ名を `text` とすると、自動的に LF から CRLF への変換が行なわれます。そのため、hpif スクリプトは必要ありません。

次に、そのプリンタにアクセスしたいと思っている他ホストにおいて、そのホストの /etc/printcapファイルに次にあげるエントリを作ります。

1. 名前のエントリ。どんな名前でもよいのですが、簡単のため、多分、
プリンタホストで設定されたプリンタ名や別名と同じものを使いたいと思うでしょう。
2. `lp` 項目で指定されるデバイスは明示的に空にします(`:lp=:` とします)。
3. スプーリングディレクトリを作成し、`sd` 項目でその位置を指定します。LPDでは、プリンタホストにジョブを送信するまでの間、このディレクトリにジョブを格納します。
4. `rm` 項目でプリンタホストの名前を指定します。

5. rp 項目で プリンタホストに接続したプリンタ名を指定します。

これで終わりです。 変換フィルタやページの大きさやその他の事項を /etc/printcap に加える必要はありません。

次に、 リモートホストに接続されたプリンタで印字するための設定例を示します。 ホスト rose には 2 台のプリンタ bamboo と rattan が接続されています。これらのプリンタをホスト orchid のユーザが使えるようにしましょう。最初に orchid の /etc/printcap を示します (このファイルは、「ヘッダページの出力を許可する」 で参照することができます)。このファイルには、既に、プリンタ teak 用のエントリがありました。以下では、 これに、ホスト rose にある2台のプリンタ用のエントリが加えられています。

```
#
#  /etc/printcap for host orchid - added (remote) printers on rose
#

#
#  teak is local; it is connected directly to orchid:
#
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
        :lp=/dev/lpt0:sd=/var/spool/lpd/teak:mx#0:\
        :if=/usr/local/libexec/ifhp:\
        :vf=/usr/local/libexec/vfhp:\
        :of=/usr/local/libexec/ofhp:

#
#  rattan is connected to rose; send jobs for rattan to rose:
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :lp=:rm=rose:rp=rattan:sd=/var/spool/lpd/rattan:

#
#  bamboo is connected to rose as well:
#
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :lp=:rm=rose:rp=bamboo:sd=/var/spool/lpd/bamboo:
```

orchid で必要となる作業はスプーリングディレクトリを作ることだけです。

```
# mkdir -p /var/spool/lpd/rattan /var/spool/lpd/bamboo
# chmod 770 /var/spool/lpd/rattan /var/spool/lpd/bamboo
# chown daemon:daemon /var/spool/lpd/rattan /var/spool/lpd/bamboo
```

これで、orchid のユーザが rattan と bamboo で印字することができるようになりました。 たとえば、orchid のユーザが次のように入力したとします。

```
% lpr -P bamboo -d sushi-review.dvi
```

すると、orchid の LPD システムは、 ジョブをスプーリングディレクトリ /var/spool/lpd/bamboo にコピーし、これが DVI ファイルを印字するジョブであることを記録します。 ホスト rose の bamboo スプーリングディレクトリに十分な容量が確保でき次第、 両者の LPD は、ジョブのファイルを rose に転送します。 このファイルは、そのすべてが印字されるまで、rose のキューに留まります。 (bamboo は PostScript® プリンタなので) DVI から PostScript® への変換は rose でおこなわれます。

ネットワークにおけるデータストリームの インタフェースを持つプリンタ

プリンタのネットワークインタフェースカードは、 2 種類に分類することができます。 1 つはスプーラをエミュレートするもの (高価) で、もう 1 つはシリアルやパラレルポートを使うようにプリンタにデータを送ることができるだけのもの (安価) です。この節では、後者の使い方を説明します。前者のプリンタは、前節「リモートホストに接続されたプリンタ」の方法が適用できます。

/etc/printcap ファイルでは、 シリアルかパラレルのインタフェースのどちらを使うのか、 そして、(シリアルインタフェースを使う場合) そのボーレートはいくらであるか、フロー制御は使うのか、タブのための遅延を加えるのか、 改行文字を変換するかなどの指定をおこなうことができます。しかし、TCP/IP や他のネットワークポートからデータを受け取るプリンタを接続するための指定をおこなうことはでき ません。

ネットワーク接続されたプリンタにデータを送るためには、
テキストフィルタと変換フィルタから呼び出すことができる
通信プログラムを開発する必要があります。以下に、 そのようなプログラムの例を示します。スクリプト netprint では、 標準入力から印字データをすべて受け取り、ネットワーク接続されたプリンタにこれを送ります。 netprint の最初の引数でプリンタのホスト名を、 2 番目の引数で接続するポート番号を指定します。 このプログラムでは単方向通信 (FreeBSD からプリンタ) のみをサポートしていることに注意してください。ネットワークプリンタの多くは双方向通信をサポートしていますので、 その恩恵 (プリンタの状態を得たり、 課金をおこなうなど) にあずかりたいと思われるかもしれません。

```
#!/usr/bin/perl
#
#  netprint - Text filter for printer attached to network
#  Installed in /usr/local/libexec/netprint
#

$#ARGV eq 1 || die "Usage: $0 <printer-hostname> <port-number>";

$printer_host = $ARGV[0];
$printer_port = $ARGV[1];

require 'sys/socket.ph';

($ignore, $ignore, $protocol) = getprotobyname('tcp');
($ignore, $ignore, $ignore, $ignore, $address)
    = gethostbyname($printer_host);

$sockaddr = pack('S n a4 x8', &AF_INET, $printer_port, $address);

socket(PRINTER, &PF_INET, &SOCK_STREAM, $protocol)
```

```
    || die "Can't create TCP/IP stream socket: $!";
connect(PRINTER, $sockaddr) || die "Can't contact $printer_host: $!";
while (<STDIN>) { print PRINTER; }
exit 0;
```

このスクリプトは、様々なフィルタが利用することができます。仮に、Diablo 750-N ラインプリンタを持っており、これがネットワークに接続されているとしましょう。プリンタはポート番号 5100 にて印字するデータを受け取ります。 プリンタのホスト名は `scrivener` とします。このとき、このプリンタのテキストフィルタは次のようになります。

```
#!/bin/sh
#
#  diablo-if-net - Text filter for Diablo printer `scrivener' listening
#  on port 5100.  Installed in /usr/local/libexec/diablo-if-net
#

exec /usr/libexec/lpr/lpf "$@" | /usr/local/libexec/netprint scrivener 5100
```

9.4.4. プリンタの利用に制約を与える

本節では、プリンタの利用に制約を与えるための情報を記しています。 LPD システムでは、プリンタ(ローカル、リモートのいずれに接続されていても)にアクセスできる人を制限する機能、複数部のコピーの印字の可否を制御する機能、ジョブのサイズの最大値やプリンタキューに入るジョブの最大個数を制御する機能を提供しています。

複数部のコピーの印字を制限する

LPD システムではユーザが複数部のコピーの印字を簡単におこなう 機能を提供しています。ユーザが、(たとえば) `lpr -#5` コマンドを使ってジョブを印字すると、 ジョブのそれぞれのファイルのコピーを 5 部得ることができます。 これがよい機能であると思うかどうかは人それぞれでしょう。

複数部のコピーの印字によってプリンタが 必要以上に消耗してしまうと感じるならば、 /etc/printcap ファイルに `sc` 項目を加えてください。これにより、 lpr(1) の `-` オプションの使用が禁止されます。このオプションが指定されているにも関らず、`-` オプションを使うと、次のようなメッセージが表示され、 このオプションの利用できない旨を伝えます。

```
lpr: multiple copies are not allowed
```

リモートホストからプリンタをアクセスできる 設定にしている場合 (この設定については、「リモートホストに接続されたプリンタ」 をご覧ください)、そのリモートホストの /etc/printcap にも同じように `sc` 項目を追加する必要があることに注意してください。そうしないと、ユーザは別なホストから複数部のコピーの 印字をすることができてしまいます。

例を使って説明しましょう。次に示す /etc/printcap ファイルは、ホスト `rose` のものです。プリンタ `rattan` は極めて頑丈なので、 複数部のコピーの印字は許可されています。しかし、 レーザプリンタの `bamboo` はもう少しデリケートで、このプリンタから複数部のコピーを印字することを `sc` 項目を追加することで禁止しています。

```
#
#  /etc/printcap for host rose - restrict multiple copies on bamboo
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :sh:sd=/var/spool/lpd/rattan:\
        :lp=/dev/lpt0:\
        :if=/usr/local/libexec/if-simple:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:sd=/var/spool/lpd/bamboo:sc:\
        :lp=/dev/ttyu5:ms#-parenb cs8 clocal crtscts:rw:\
        :if=/usr/local/libexec/psif:\
        :df=/usr/local/libexec/psdf:
```

さらに、orchid の /etc/printcap にも

```
sc
```

項目を追加する必要があります (orchid でこの編集をおこなっているときに、ついでに、プリンタ teak でも複数部のコピーの印字を禁止することにしましょう)。

```
#
#  /etc/printcap for host orchid - no multiple copies for local
#  printer teak or remote printer bamboo

teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
        :lp=/dev/lpt0:sd=/var/spool/lpd/teak:mx#0:sc:\
        :if=/usr/local/libexec/ifhp:\
        :vf=/usr/local/libexec/vfhp:\
        :of=/usr/local/libexec/ofhp:

rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :lp=:rm=rose:rp=rattan:sd=/var/spool/lpd/rattan:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :lp=:rm=rose:rp=bamboo:sd=/var/spool/lpd/bamboo:sc:
```

sc 項目を指定することにより、 lpr -# の使用を防ぐことができます。しかし、この状態では lpr(1) を複数回起動したり、 1 回のジョブで次のように同じファイルを複数個指定することを防ぐまでには至っていません。

```
% lpr forsale.sign forsale.sign forsale.sign forsale.sign forsale.sign
```

このような悪用を防ぐ方法は (その指示を無視することも含めて) たくさんあります。各自で調べてみてください。

プリンタを使用できる人を限定する

それぞれのプリンタを使用できる人を限定するには、UNIX® のグループ権限のメカニズムを利用し、さらに、 /etc/printcap で `rg` 項目を指定することでおこないます。あるプリンタにアクセスさせてもよいと思うユーザすべてをグループのどれかに入れてください。そして、 そのグループ名を `rg` で指定します。

このとき、そのグループに含まれないユーザ (`root` も含みます) がプリントしようとすると、次のようなメッセージが表示されます。

```
lpr: Not a member of the restricted group
```

`sc` (suppress multiple copies : 複数部のコピーの印字を禁止する) を指定するときと同様に、`rg` が指定されたプリンタがリモートホストからもアクセスでき (この設定については、「リモートホストに接続されたプリンタ」をご覧ください)、かつ、そのホストでもプリンタを使用できる人を限定するのが妥当であると思う場合は、そのホストの /etc/printcap にも *rg* 指定をおこなう必要があります。

たとえば、プリンタ `rattan` は誰でも利用できるが、`bamboo` はグループ `artists` に属している人のみが利用できるようにしてみましょう。 以下に、もうお馴染みとなったホスト `rose` の /etc/printcap を示します。

```
#
#  /etc/printcap for host rose - restricted group for bamboo
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :sh:sd=/var/spool/lpd/rattan:\
        :lp=/dev/lpt0:\
        :if=/usr/local/libexec/if-simple:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:sd=/var/spool/lpd/bamboo:sc:rg=artists:\
        :lp=/dev/ttyu5:ms#-parenb cs8 clocal crtscts:rw:\
        :if=/usr/local/libexec/psif:\
        :df=/usr/local/libexec/psdf:
```

これ以外の /etc/printcap ファイル (ホスト `orchid` のもの) はそのままにしておくことにします。もちろん、`orchid` のユーザは全員 `bamboo` を利用することができます。これは、`orchid` には特定のユーザのみにしかアクセスさせておらず、そのユーザにはプリンタを利用させたいと思っているからなのかもしれませんし、そうでないかもしれません。

1台のプリンタを複数グループのユーザに利用させることはできません。

入力可能なジョブのサイズを制限する

たくさんのユーザからプリンタが利用される場合には、多分、ユーザが印字要求を出すことができるファイルのサイズに上限値を置く必要が生じるでしょう。結局のところ、スプーリングディレクトリ

が置かれているファイルシステムの空き容量がその 上限値になる訳ですが、
あるユーザがこれを独占的に使用すること避けるために、
他ユーザからのジョブ用の空き容量を確保する必要もあります。

LPD では、mx 項目を指定することにより、
ジョブ中の個々のファイルのサイズの上限値を制限する機能を提供しています。 指定される
ファイルサイズの単位は BUFSIZ ブロックで、1 BUFSIZ ブロックは 1024バイトを表わします。この mx
項目の値として 0 が指定されると、 ファイルサイズの制限はなくなります。 mx が指定されない場合は、
デフォルトの制限として 1000 ブロックが使われます。

この制限はジョブ中の各 ファイルに対して適用されるものであり、
ジョブ全体のサイズ を制限するものではありません。

ところで、 プリンタに設定された上限値を超えるファイルサイズの ファイルが入力された場合でも、LPD
はこれを拒否しません。その代わりに、このファイルは、
その先頭から上限値のファイルサイズまでしかキューに入れられません。
そして、その部分までが印字され、 残りの部分は捨てられます。
これが正しい動作といえるのかどうかは議論の余地があるところです。

それでは、設定例に登場しているプリンタ rattan と bamboo
の印字可能なファイルサイズに制限を加えてみましょう。 artists グループの人達が作る PostScript®
ファイルのサイズは 巨大になる傾向があるので、上限値を 5M バイトとします。 それから、
プレインテキスト用のラインプリンタは無制限とします。

```
#
#  /etc/printcap for host rose
#

#
#  No limit on job size:
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :sh:mx#0:sd=/var/spool/lpd/rattan:\
        :lp=/dev/lpt0:\
        :if=/usr/local/libexec/if-simple:

#
#  Limit of five megabytes:
#
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:sd=/var/spool/lpd/bamboo:sc:rg=artists:mx#5000:\
        :lp=/dev/ttyu5:ms#-parenb cs8 clocal crtscts:rw:\
        :if=/usr/local/libexec/psif:\
        :df=/usr/local/libexec/psdf:
```

この場合もそうですが、この制限はローカル (ホスト rose) のユーザのみに適用されます。
リモートホストからプリンタを利用できるように設定している場合は、
そのリモートホストのユーザはこの制限を受けません。
これらのユーザにも制限を加える場合は、リモートホストの /etc/printcap の mx

を指定する必要があります。 リモートホストから印字するための詳しい情報については、「リモートホストに接続されたプリンタ」 を参照してください。

リモートホストに接続されたプリンタへのジョブの
サイズを制限する特別な方法は他にもあります。これについては、
「リモートホストからの利用を制限する」 を参照してください。

リモートホストからの利用を制限する

LPD スプーリングシステムでは、
リモートホストから要求されたジョブの印字を制限するための方法がいくつか提供されています。

ホストの制限

ローカルの LPD が印字要求を受け付けるリモートホストは、ファイル /etc/hosts.equiv と /etc/hosts.lpd によって制御することができます。LPD では、あるホストから印字の要求がきたとき、このホストの名前がこれら 2 つのファイルのどちらかに含まれているかどうかを調べます。これが含まれていない場合は、LPD はこの要求を拒否します。

これらのファイルの形式は単純です。 各行にホストの名前を 1つずつ書いていきます。ファイル /etc/hosts.equiv の方は ruserok(3) プロトコルでも利用され、 rsh(1) や rcp(1) といったプログラムの動作に影響するので注意が必要です。 /etc/hosts.equiv の記述は慎重におこないましょう。

例として、以下にホスト rose の /etc/hosts.lpd を示します。

```
orchid
violet
madrigal.fishbaum.de
```

この例では、rose はホスト orchid, violet そして madrigal.fishbaum.de からの要求を受け付けることになります。 その他のホストが rose の LPD にアクセスしようとしても、 LPD はそのジョブを拒否します (訳注: 拒否されるのは、そのホストが /etc/hosts.equiv にも含まれていない場合です)。

サイズの制限

スプーリングディレクトリがある ファイルシステムに残しておく必要がある
空き容量の大きさを制御することができます。 ローカルプリンタ用のスプーリングディレクトリに minfree という名前のファイルを作成します。そして、
そのファイルの中にリモートホストからのジョブの
要求を受け付けるために必要な空き容量のディスクブロックサイズ (1 ディスクブロック = 512 バイト) を記します。

これで、 リモートホストのユーザにファイルシステムを満杯にされないことが保証されます。
この機能を使うと、 ローカルホストのユーザに対してある種の優先権を与えることもできます。
ローカルホストのユーザは、 minfree
ファイルで指定された値よりもディスクの空き容量が下回った後でもずっと、
ジョブをキューに入れることができるのです。

たとえば、プリンタ bamboo 用の minfree を作ってみましょう。
このプリンタのスプーリングディレクトリを調べるために、 /etc/printcap を調べてみましょう。

以下に、bamboo のエントリ部分を示します。

```
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:sd=/var/spool/lpd/bamboo:sc:rg=artists:mx#5000:\
        :lp=/dev/ttyu5:ms#-parenb cs8 clocal crtscts:rw:mx#5000:\
        :if=/usr/local/libexec/psif:\
        :df=/usr/local/libexec/psdf:
```

スプーリングディレクトリは sd 項目で指定されます。LPD がリモートホストからのジョブを受け付けるために必要な ファイルシステムの空き容量を 3M バイト (= 6144 ディスクブロック) にすることにしましょう。

```
# echo 6144 > /var/spool/lpd/bamboo/minfree
```

利用ユーザの制限

/etc/printcap の rs 項目を指定することで、ローカルプリンタを利用できるリモートホストのユーザを制限することができます。
ローカルホストに接続されたプリンタ用のエントリに rs 項目が指定されている場合、 LPD は、印字を要求したユーザのアカウントと同じログイン名が
ローカルホストに登録されている場合に限り、 そのジョブを受け付けます。それ以外のジョブを LPD は拒否します。

この機能は、(たとえば) 複数の部署がネットワークを共有しており、この内のあるユーザが部署の境界を越えて活動している場合には特に有用です。
そのようなユーザに対して、システムのアカウントを与えるだけで、
これらのユーザは自分が所属する部署のシステムから
そのシステムに接続されているプリンタを使用することができます。 これらのユーザにはむしろ、プリンタの使用だけを認め、 その他のコンピュータ資源を利用させたくないときは、
それらのユーザにはホームディレクトリを与えず、
ログインシェルはシェルとしては何の役にも立たない /usr/bin/false などを指定して、これらのユーザのアカウントはプリンタ用の "形式的な" ものとします。

9.4.5. プリンタの利用に対する課金

という訳で、印字するためには料金をとることが必要です。
取らない理由などありましょうか。紙やインクにはお金がかかります。
そして、プリンタの維持費もかかります。 プリンタには可動部分が搭載されており、これらの部分は壊れやすいという傾向があります。
プリンタや、その利用形態、維持費について調査をし、1 ページ (1 フィート、1 メートルなど) 当たりにかかるコストを調べておいてください。
これに基づき、プリンタの利用に対する課金を、実際に、 どのように始めればよいのでしょうか。

さて、残念ながら、この部分に関しては LPD スプーリングシステムはほとんど役に立ちません。課金は使用しているプリンタの種類、印字するもののファイルの形式、 プリンタの利用に対する課金でのあなた自身の要求に大きく左右されます。

課金システムを実現するためには、プリンタのテキストフィルタ

(プレインテキストのジョブに対して課金するため) と変換フィルタ
(その他のファイル形式に対して課金するため) を変更して、 印字したページを数えたり、
プリンタに印字したページ数を取得するための要求を送る必要があります。
ただし、出力フィルタのみを利用している場合は、
課金をおこなうことができません。フィルタに関しては、「フィルタ」をご覧ください。

一般に、課金方式には次の 2 つがあります。

- 定期的に課金する方法 はよく利用される方法です。この理由は、
  恐らく比較的簡単に実現できるからです。 誰かがジョブを印字する度に、フィルタはそのユーザ名、
  ホスト名、印字したページ数を課金データファイルに記録します。
  毎月、毎学期、毎年、その他お好みの時期に、 各プリンタの課金用ファイルを集め、
  それぞれのユーザが印字したページ数を合計して その分の課金をおこないます。
  次回の課金期間をデータを 0 にして課金を再開するために、 すべてのログファイルを削除します。
- 利用毎に課金する方法 はあまり利用されていません。これは、
  実現するのが比較的難しいからです。この方式では、 プリンタを使用したらすぐに、
  フィルタがユーザにその利用に対する課金をおこないます。
  ディスククォータのように、課金作業は瞬時におこなわれます。
  この方式では、ユーザのアカウントが赤字になる場合に、
  ユーザが印字をおこなうことを拒否することができます。 また、ユーザに "プリンタ版 quota"
  を調べたり、 調整したりする方法を提供したいと思うかもしれ ません。
  これを実現するためには、ユーザとその quota を追跡するために、
  あるデータベース用のコードが必要となります。

LPD スプーリングシステムでは、 どちらの方式にも簡単に対応できます。(ほとんどの場合は)
フィルタを用意しなければならないので、 課金作業のためのコードも用意しなければなりません。
しかし、明るい面もあります。
それは、課金方式に関して、非常に大きな柔軟性が与えられたということです。
たとえば、「定期的に課金する方法」か、 「利用毎に課金する方法」のどちらかを選びまず、そして、
どんな情報 (ユーザ名、ホスト名、ジョブのタイプ、印字された頁数、
使用した紙の大きさ、印字をするために要した時間など) をログに記録するかを決めます。
以上のことをおこなうには、上記の情報を保持するために、 フィルタを変更しなくてはなりません。

手軽なプリンタ課金方法

FreeBSD には、「定期的に課金する方法」による課金を すぐに設定できるように、2
個のプログラムを添付しています。 その内の1つはテキストフィルタ lpf で、
これについては、「テキストフィルタ lpf」をご覧ください。もう1つは、 pac(8) で、
これはプリンタの課金データファイルからのエントリを集め、 これを合計するプログラムです。

「フィルタはどのように機能しているか」で述べたように、 LPD
ではテキストフィルタや変換フィルタを起動しますが、
そのコマンドラインで使用している課金データファイルの名前が指定されます。
両フィルタはこの引数を使って、
どの課金データファイルのエントリに書き込めばよいのかを知ることができます。 このファイルの名前は
/etc/printcap 中の af 項目によって指定されます。 このファイルが絶対パ スで指定されない場合は、
スプーリングディレクトリからの相対パスとして扱われます。

LPD は、紙のページの幅と行数 (pw と pl 項目で 指定される) を引数として lpf を起動します。lpf

フィルタでは、何ページ印字したかを決定するためにこれらの引数を使用します。
ファイルをプリンタに送った後、課金情報を課金データファイルに書き込みます。
このファイルは次のようになります。

```
2.00 rose:andy
3.00 rose:kelly
3.00 orchid:mary
5.00 orchid:mary
2.00 orchid:zhang
```

課金データファイルはプリンタ毎に分けて作るべきです。これは、lpf にはファイルをロックする機構が組み込まれていないためです。したがって、lpf が 2 つ起動されたとき、同じファイルに同時に書き込みをおこなった場合、お互いのエントリを破壊してしまうかもしれません。
課金用ファイルを各プリンタ毎に確実に分けるには、/etc/printcap 中の af=acct 項目を使います。そうすれば、それぞれの課金用ファイルがプリンタのスプーリングディレクトリに、acct という名称で作成されます。

プリンタの利用に対してユーザに課金する準備ができたら、pac(8) プログラムを実行してください (課金したいプリンタのスプーリングディレクトリに移動した後、pac と入力してください)。次のような、ドル中心主義の課金リストが表示されます (訳注: ドル中心主義という表現は、表示がドルで出ることへの著者の皮肉でしょう。セントがあるので小数点以下が表示されますが、この機能も日本では邪魔ですね)。

```
  Login               pages/feet   runs    price
orchid:kelly               5.00     1   $  0.10
orchid:mary               31.00     3   $  0.62
orchid:zhang               9.00     1   $  0.18
rose:andy                  2.00     1   $  0.04
rose:kelly               177.00   104   $  3.54
rose:mary                 87.00    32   $  1.74
rose:root                 26.00    12   $  0.52

total                    337.00   154   $  6.74
```

pac(8) が受け付ける引数には次のようなものがあります。

-P printer

プリンタ *printer* の利用に対する課金リストを作成します。このオプションは、/etc/printcap の af が絶対パスで指定されていた場合に限り、動作します。

-c

ユーザ名のアルファベット順ではなく、課金額の低い順にリストを並べます。

-m

課金データファイルにあるホスト名を無視します。このオプションを使用すると、ホスト alpha のユーザ smith とホスト gamma のユーザ smith は同一人物として扱われます。このオプションが指定されない場合は、両者は別なユーザとして扱います。

-p price

/etc/printcap の pc 項目で指定された値、または、 デフォルトの値 (2 セント) に代わり、紙1ページ、または、 1フィート当たりの価格を指定します。 *price* として、浮動小数点数を指定することができます。

-r

リストの並べる順番を逆順にします。

-s

課金リストを作成し、 課金データファイルを削除します。

name...

ユーザ *names* に対する課金情報のみを表示します。

pac(8) が生成するデフォルトのリストには、 各ホストのユーザ別に印字ページ数が表示されます。(ユーザがサイト内のすべてのホストを使用できるため) ホスト名の情報が意味を持たない場合、 pac -m を実行してください。次のようなリストが得られます。

```
  Login               pages/feet   runs    price
andy                        2.00     1   $  0.04
kelly                     182.00   105   $  3.64
mary                      118.00    35   $  2.36
root                       26.00    12   $  0.52
zhang                       9.00     1   $  0.18

total                     337.00   154   $  6.74
```

課金額を決めるために、 pac(8) は /etc/printcap ファイルの pc 項目で指定された値 (デフォルト値は 200、すなわち 1 ページ当たり 2 セント) を使います。この項目で、印字物に課金したい ファと思う 1 ページ当たり、 または、1 フィート当たりの価格を 100 分の 1 セント単位で指定します。 pac(8) を -p オプション付きで起動すると、 この値を置き換えることができます。 この -p オプションで指定する額の単位は、 100 分の 1 セント単位ではなく、ドル単位です。たとえば、次の指定では、 1 ページ当たりの単価が 1 ドル 50 セントになります。

```
# pac -p1.50
```

このオプションを使うと、 実際の課金額を集計することができます。

最後に、pac -s を起動すると、課金情報は課金データ累計ファイルに保存されます。このファイルの名前は、プリンタの課金データファイルの後ろに _sum を付けたものとなります。そして、 課金データファイルは削除されます。次に pac(8) が起動されると、その時点までの累計金額を得るために、 課金データ累計ファイルが読み込まれ、通常の課金データファイルからの情報に加算されます。

印字されたページ数をどのように数えるか?

課金を、リモートホストからの印字でさえも、 正確におこなうためには、

ジョブで使用された紙が何ページであるかを特定できる必要があります。
このことは、プリンタ利用に対する課金をおこなう上の根本的な問題です。

プレインテキストのジョブの場合、 問題を解決するのはさほど難しくはありません。
ジョブが何行であったかを数え、プリンタがサポートしている紙 1
ページに印字できる最大の行数と比較すればよいのです。
重ね打ちするために利用されるファイル中のバックスペース文字や、
物理的に複数の行に渡る長い論理行に対する取り扱いを忘れずにおこなってください。

(「テキストフィルタ lpf」で紹介した) テキストフィルタ lpf では、課金をおこなうときに、
これらの取り扱いをおこなってくれます。
課金をおこなうために必要なテキストフィルタを作成している方は、 lpf
のソースコードが参考になるでしょう。

これに対して、他のファイル形式の処理はどのようにすれば よいのでしょうか。

まず、DVI から LaserJet, または、DVI から PostScript® への変換の場合、フィルタが dvilj や dvips の
出力メッセージを解析することで、 何ページ分の変換がおこなわれたかを知ることができます。
他のファイル形式とその変換プログラムに関しても、 同様のことができるかもしれません。

しかし、この方式には問題点があります。それは、
変換されたページがすべて印字されるとは限らないということです。
たとえば、プリンタが紙詰まりを起こしたり、トナー切れになったり、
はたまた、爆発したりするかもしれません。
そのような状況により印字が途中で中止されたとしても、この方式では、
ユーザは全ページ分の料金を課されてしまうのです。

それでは、どのような対策をたてることができるのでしょうか。

正確な 課金をおこなうための唯一の確実な方法は、
何ページ印字したのかを知らせることができるプリンタを入手し、
これをシリアルポートかネットワークに接続することです。 ほとんどすべての PostScript®
プリンタではこの概念がサポートされています。 他のプリンタも同様です (Imagen
レーザプリンタをネットワーク接続するなど)。 それぞれのプリンタのフィルタを、
ジョブを印字した後で印字ページ数を得るように変更してください。
そして、課金情報はここで得られた値のみに 基づいて記録してください。行数を数えたり、
エラーが生じやすいファイルの調査は必要とされません。

もちろん、 気前よく印字料金をすべて無料にすることもできます。

9.5. プリンタを使う

この節では、FreeBSD で設定したプリンタを使う方法について説明します。
ここでは、ユーザレベルでのコマンドを概説します。

lpr(1)

印字をおこないます。

lpq(1)

プリンタキューを調べます。

lprm(1)
: プリンタキューにあるジョブを削除します。

また、「プリンタの管理」節で説明されている管理者用コマンド lpc(8) もあり、プリンタやそのキューの制御のために用いられています。

lpr(1)、lprm(1)、そして lpq(1) の 3 コマンドは、`-P printer-name` オプションをとり、これによって、/etc/printcap のように操作の対象となるプリンタやキューを指定します。これによって、様々なプリンタに対してジョブを送る、取り消す、調査することができます。`-P` が使われなかった場合は、これらのコマンドは `PRINTER` 環境変数で指定されたプリンタを使用します。そして、`PRINTER` 環境変数がなかった場合は、これらのコマンドはデフォルトのプリンタ `lp` を使います。

以下では、デフォルトプリンタという用語が意味するプリンタは、`PRINTER` 環境変数で指定されたプリンタ、もしくは、`PRINTER` 環境変数がない場合は、`lp` という名前のプリンタです。

9.5.1. 印字する

ファイルを印字するためには、 次のように入力してください。

```
% lpr filename ...
```

これにより、入力されたファイルのそれぞれをデフォルトのプリンタから印字します。ファイル名が与えられなかった場合、lpr(1) は標準入力から印字するデータを読み込みます。たとえば、
次のコマンドにより、ある重要なシステムファイルが印字されます。

```
% lpr /etc/host.conf /etc/hosts.equiv
```

印字させるプリンタを選択するためには、 次のように入力します。

```
% lpr -P printer-name filename ...
```

次の例では、プリンタ `rattan` に、
カレントディレクトリにあるファイルの詳細なリストを印字しています。

```
% ls -l | lpr -P rattan
```

上記の lpr(1) コマンドではファイル名の指定がないので、`lpr` は標準入力から印字するデータ、この場合、`ls -l` コマンドの出力、を読み込みます。

lpr(1) コマンドでは、出力の整形を制御したり、ファイル変換を適用したり、複数部数のコピーを作成したり、などといた様々な幅広いオプションを受け付けることもできます。詳細については、「その他の印字オプション」をご覧ください。

9.5.2. ジョブの処理状況を調べる

lpr(1) コマンドを使って印字をする場合、プリントしようとするデータは "プリントジョブ" と呼ばれる箱に一緒に置かれ、これが LPD スプーリングシステムに送られます。
プリンタにはそれぞれジョブ用のキューがあり、
送られてきたジョブはあなたや他のユーザからの別のジョブと一緒にそのキューで並んで、
処理される順番を待ちます。 プリンタは到着順にこれらのジョブの印字をおこないます。

デフォルトプリンタのキューの状態を表示するには、 lpq(1) と入力します。プリンタを指定するときは、 -P オプションを使います。たとえば、次のコマンド

```
% lpq -P bamboo
```

は、プリンタ bamboo のキューの状態を表示します。この lpq コマンドの出力結果の例を次に示します。

```
bamboo is ready and printing
Rank    Owner    Job  Files                              Total Size
active kelly    9    /etc/host.conf, /etc/hosts.equiv   88 bytes
2nd    kelly    10   (standard input)                   1635 bytes
3rd    mary     11   ...                                78519 bytes
```

この例では、bamboo のキューに 3 つのジョブがあることが分かります。 最初のジョブはユーザ kelly からのものであり、"ジョブ番号" 9 が割り当てられています。
プリンタのすべてのジョブには一意なジョブ番号が付けられています。
ほとんどの場合、このジョブ番号は無視することができますが、
ジョブをキャンセルするときにはこの番号が必要になります。
このことの詳細については、「ジョブの削除 」をご覧ください。

ジョブ番号 9 のジョブは 2 つのファイルを処理します。すなわち、 lpr(1) のコマンドラインに複数のファイル名が与えられたときは、
1つのジョブとして扱われるのです。このジョブは、現在、 アクティブジョブ ("Rank" の欄の active という後に注目) になっています。
これは、プリンタからそのジョブが現在印字されているはずであることを意味しています。 2 番目のジョブでは、 lpr(1) コマンドに標準入力からデータが与えられています。 3番目のジョブはユーザ mary から与えられました。 このジョブのサイズはとても大きくなっています。
彼女がプリントしようとしたファイルのパス名はここで表示させるには長すぎるため、 lpq(1) コマンドはドットを 3 つだけ表示しています。

lpq(1) からの出力で一番最初の行もまた有益な情報を与えています。
この行から、プリンタが現在何をしているか (あるいは、少なくとも LPD がプリンタがしていると思っていること) が分かります。

lpq(1) コマンドは -l オプションもサポートしています。 これにより、 詳しい情報が表示されます。 lpq -l の実行例を次に示します。

```
waiting for bamboo to become ready (offline ?)
kelly: 1st                              [job 009rose]
```

```
       /etc/host.conf                    73 bytes
       /etc/hosts.equiv                  15 bytes

kelly: 2nd                               [job 010rose]
       (standard input)                  1635 bytes

mary: 3rd                                [job 011rose]
      /home/orchid/mary/research/venus/alpha-regio/mapping 78519 bytes
```

9.5.3. ジョブの削除

印字するようジョブを　　送った後で印字を中断したくなったときは、　　lprm(1)　　コマンドで、キューの中からそのジョブを削除することができます。 大抵の場合、アクティブジョブでさえも lprm(1) を使って削除することができますが、
そのジョブの一部またはすべてが印字されてしまうかもしれません。

デフォルトプリンタへのジョブを削除するためには、最初に、　　lpq(1)
を使ってそのジョブ番号を調べます。　　すなわち、それから、
次のように入力して、ジョブを削除します。

```
% lprm job-number
```

特定のプリンタへのジョブを削除するときは、　　-P　　オプションを使ってそのプリンタを指定します。たとえば、プリンタ bamboo のキューからジョブ番号 10 のジョブを削除するには次のようにします。

```
% lprm -P bamboo 10
```

lprm(1) コマンドには略記法がいくつかあります。

lprm -
: あなたが (デフォルトプリンタへ) 送ったジョブをすべて削除します。

lprm *user*
: ユーザ　　*user*　　が　　(デフォルトプリンタへ)　　送ったジョブをすべて削除します。
他のユーザのジョブを削除できるのはスーパユーザだけです。
あなたは、あなた自身のジョブしか削除することはできません。

lprm
: ジョブ番号もユーザ名もシンボル -も指定されないときは、 lprm(1) は現在のアクティブジョブを、
そのジョブを送ったのがあなた自身であるときに限り、
デフォルトプリンタから削除します。ただし、
スーパユーザは任意のアクティブジョブを削除することができます。

上記の略記法をデフォルトプリンタではなく　　特定のプリンタに対しておこなうときは、-P オプションでそのプリンタを指定するだけよいのです。たとえば、　　プリンタ　　rattan のキューへあなたが送ったジョブを すべて削除するためには次のようにします。

```
% lprm -P rattan -
```

ネットワーク環境で作業をしている場合、
あるホストから送られたプリンタジョブは、これを送ったホストで lprm(1)
を使った場合に限って、 これを削除することができます。
他のホストで同じプリンタを使えたとしても、
このジョブを削除することはできません。
次の例では、他ホストからジョブを削除することを試みています。

```
% lpr -P rattan myfile
% rlogin orchid
% lpq -P rattan
Rank   Owner      Job  Files                          Total Size
active seeyan     12    ...                            49123 bytes
2nd    kelly      13   myfile                          12 bytes
% lprm -P rattan 13
rose: Permission denied
% logout
% lprm -P rattan 13
dfA013rose dequeued
cfA013rose dequeued
```

9.5.4. その他の印字オプション

lpr(1) コマンドには、テキストの整形や、 図や他のファイル形式の変換、複数部コピーの生成、
ジョブの扱いなどを制御することができます。
この節では、これに関するオプションについて記しています。

整形と変換に関するオプション

以下の lpr(1) 用のオプションはジョブにおける ファイルの整形の制御に関するものです。
このオプションは、ジョブにプレインテキストが含まれない場合や pr(1)
ユーティリティを使ってプレインテキストを整形する場合に用いてください。

次の例では、プリンタ bamboo に (TeX 組版システムによる) DVI ファイル fish-report.dvi
を印字しています。

```
% lpr -P bamboo -d fish-report.dvi
```

このオプションは、 ジョブに含まれるすべてのファイルに対して適用されます。 したがって、1
つのジョブに (たとえば) DVI ファイルと ditroff
ファイルを混在させることはできません。その代わりに、 ファイルを形式毎に別々のジョブに分け、
それぞれのジョブでその形式用の変換オプションを使って印字してください。

-p と -T を除くすべてのオプションを使用 するためには、
出力先プリンタ用の変換フィルタが必要です。たとえば、 -d
オプションを使用するには、DVI 用の変換フィルタが必要

です。詳細については、「変換フィルタ」で説明しています。

`-c`

cifplot ファイルを印字します。

`-d`

DVI ファイルを印字します。

`-f`

FORTRAN プログラムを印字します。

`-g`

plot のデータを印字します。

`-i number`

出力に対して、*number* カラム分の字下げをおこないます。 *number* が省略されると、 8 カラム分字下げされます。
このオプションはある変換フィルタと一緒の指定されたときのみに機能します。

`-i` と数字の間に空白を入れてはいけません。

`-l`

制御文字を含む文字通りのテキストデータを印字します。

`-n`

ditroff (device independent troff) データを印字します。

-p

印字する前に pr(1) によってプレインテキストを整形します。 詳細については pr(1) をご覧ください。

`-T title`

pr(1) コマンドにより生成されるヘッダを、 ファイル名の代わりに *title* とする。 このオプションは、 `-p` と一緒に使ったときのみ機能する。

`-t`

troff データを印字します。

`-v`

ラスタのデータを印字します。

次の例では、ls(1) のマニュアルを美しく整形したものをデフォルトプリンタで印字しています。

```
% zcat /usr/shared/man/man1/ls.1.gz | troff -t -man | lpr -t
```

zcat(1) コマンドで ls(1) のマニュアルのソースファイルの圧縮を復元し、これを troff(1) コマンドに渡しています。 これによりソースファイルが整形され GNU troff の形式となります。 その結果は lpr(1) に渡され、 LPD スプーラへジョブの要求が発せられます。 lpr(1) には `-t`

オプションが使われているため、 スプーラでジョブを印字したときに GNU troff の形式から、デフォルトプリンタが解釈できる形式へと変換されます。

ジョブに関するオプション

以下のオプションは、lpr(1) によって、 そのジョブを特殊な扱いにするよう LPD に指示するためのものです。

-# *copies*

ジョブに含まれるファイルのそれぞれを 1 部だけ印字するのではなく、 *copies* 部のコピーを生成させるものです。管理者によっては、
プリンタの消耗を避け、コピー機による複製を奨励するために
このオプションの使用が禁止されているかもしれません。
これに関しては、「複数部のコピーの印字を制限する 」をご覧ください。

次の例では、デフォルトプリンタで parser.c を 3 部コピーし、次に、 parser.h を 3 部コピーしています。

```
% lpr -#3 parser.c parser.h
```

-m

印字ジョブが完了した後で、メールを送ります。 このオプションを付けると、LPD システムはジョブの処理が終了したときに、 あなたのアカウントにメールを送ります。 メールのメッセージには、ジョブが正常終了したのか、あるいは、 何か異常があり、(しばしば) その異常が何であったのかが書かれています。

-s

印字ファイルをスプールディレクトリにコピーせず、 代わりに、 シンボリックリンクを作成するよう指示します。

印字させるジョブのサイズが大きいとき、
このオプションを使うと便利かもしれません。このオプションにより、 スプールディレクトリの容量が節約されます (それに、巨大なジョブのお陰でスプールディレクトリのあるファイルシステムの空き容量がなくなってしまうかもしれません)。 さらに、LPD がいちいちすべてのデータをコピーする必要がなくなりますので、時間の節約にもなります。

ただし、欠点もあります。LPD はオリジナルのファイルを直接参照するので、印字が終了するまでそのファイルを変更したり削除することができません。

リモートのプリンタで印字している場合、 LPD は、結局のところ、ローカルホストからリモートホストにファイルをコピーする必要があります。
したがって、`-s`
オプションはローカルのスプーリングディレクトリの空き容量を節約するだけで、リモート側では節約されません。 それでも、このオプションは有用です。

-r

ジョブに含まれるファイルを、 スプーリングディレクトリにファイルをコピーした後に削除します。もしくは、 `-s` オプションと一緒に使われた場合は、

印字終了後に削除されます。 このオプションの使用には十分注意して下さい。

ヘッダページ用オプション

以下のオプションにより、 ジョブのヘッダページに通常印字さ れるテキストを lpr(1) に調整させることができます。 対象のプリンタからヘッダページが出力されない場合は、これらのオプションは何の効力も持ちません。 ヘッダページの設定に関する情報については、「ヘッダページ」を参照してください。

-C *text*

ヘッダページに印字されるホスト名を *text* に置き換えます。なお、 ホスト名の場所には、通常、ジョブの要求があったホストの名前が印字されます。

-J *text*

ヘッダページに印字されるジョブ名を *text* に置き換えます。ジョブ名の場所には、通常、ジョブの最初のファイル名、または、標準入力からデータが印字されたときは stdin が印字されます。

-h

ヘッダページの出力を禁止します。

サイトによっては、 そのヘッダページの生成方法により、このオプションの効果が現れないかもしれません。 詳細は、「ヘッダページ」をご覧ください。

9.5.5. プリンタの管理

プリンタの管理者として、プリンタの設置、設定、そして、それらのテストをおこなう必要がありました。 lpc(8) コマンドにより、これまでとは別な管理方法がプリンタと対話的におこなわれます。 lpc(8) により、次のことが可能となります。

- プリンタの起動、停止をおこなう。
- キューへの入力の許可、禁止をおこなう。
- それぞれのキューにあるジョブの順番を変更する。

最初に用語に関する注意をしておきます。 プリンタが停止しているとは、キューの中にあるどのジョブも印字されることがない状態 を言います。この状態においても、ユーザはまだジョブの要求をおこなうことができますが、 これらのジョブはキューの中で、プリンタがスタートする状態になるまで、あるいは、キューの内容が削除されるまで待たされることになります。

キューが禁止状態にあると、 (root 以外の)すべてのユーザがプリンタにジョブを要求することができません。 キューが許可状態にある場合は、ジョブの入力が許可されます。 キューが禁止状態にある場合でも、 プリンタをスタートする状態にすることは可能です。この場合は、 キューが空になるまで、キュー内のジョブの印字が続けられます。

一般的に、lpc(8) コマンドを使用するには root 権限を持っている必要があります。 一般のユーザも lpc(8) コマンドを使うことはできますが、 プリンタの状態を取得することとハングしたプリンタ

を再スタートすることだけに使用が制限されています。

以下に、lpc(8) コマンドに関する説明の要約を述べます。
ほとんどのコマンドでは、操作対象となるプリンタを指定するため *printer-name* 引数を与えます。
printer-name の代わりに `all` が与えられると、操作は /etc/printcap
内にある全プリンタに対しておこなわれることになります。

`abort printer-name`

現在のジョブをキャンセルし、プリンタを停止させます。 キューが許可状態にある場合は、
ユーザはまだジョブを入力することができます。

`clean printer-name`

プリンタのスプーリングディレクトリから、 ジョブの古いファイルを削除します。状況によって、
とりわけ、印字途中でエラーが発生していたり、 管理操作が頻発していた場合には、
ジョブで作られたファイルを LPD が完全に削除しないことがあります。このコマンドでは、
スプーリングディレクトリに入っていないファイルを見つけ出し、 それを削除しています。

`disable printer-name`

キューに新しいジョブを入れることを禁止します。 プリンタが動作しているときは、
キューに残っているジョブの印字は続けられます。ただし、
キューが禁止状態にあったとしても、スーパーユーザ (`root`)
は常にジョブを入力することができます。

このコマンドは、 新しいプリンタやフィルタを設置している間に使用すると有用です。
すなわち、キューを禁止状態にしておくと、 `root` によるジョブのみが入力されます。
そして、他のユーザは、テストが完了し、 `enable` コマンドでキューが再度許可状態になるまで、
ジョブの入力はできなくなります。

`down printer-name message`

プリンタをダウンさせます。これは、 `disable` をおこなった後で、 `stop`
をおこなった場合と等価になります。 *message* は、ユーザが lpq(1)
コマンドでプリンタのキューの状態を調べたり、 `lpc status` でプリンタの状態を調べたときに、
プリンタの状況として表示されるメッセージです。

`enable printer-name`

プリンタのキューを許可状態にします。 ユーザはジョブの入力ができるようになりますが、
プリンタがスタートの状態になるまでは、 プリンタからは何も印字されません。

`help command-name`

command-name コマンドのヘルプメッセージを表示します。 *command-name*
が指定されなかった場合は、 利用できるコマンドの要約が表示されます。

`restart printer-name`

プリンタをスタートさせます。通常のユーザは、 LPD がある異常な状況でハングしたときに限り、
このコマンドを使用することができます。しかし、 `stop` または `down` コマンドにより、
停止状態にあるプリンタをスタートさせることはできません。 `restart` コマンドは、 `abort` の後に
`start` をおこなったことと同じになります。

`start printer-name`

プリンタをスタートさせます。 プリンタのキューにあるジョブを印字することでしょう。

`stop printer-name`

プリンタを停止します。プリンタは、 現在のジョブを終了させ、そして、キューにあるその他のジョブは印字しません。 プリンタが停止状態にあったとしても、まだ、許可状態にあるキューに対して、ジョブを送ることができます。

`topq printer-name job-or-username`

printer-name のキューに対して、ジョブ番号 *job* のジョブ、または、ユーザ *username* から送られたジョブを置き換えて、キューの先頭に持ってきます。 このコマンドに関しては、*printer-name* の代わりに `all` を使用することはできません。

`up printer-name`

プリンタをアップ状態にします。これの反対のコマンドが `down` です。`start` の次に `enable` をおこなったことと等しくなります。

コマンドラインから上記のコマンドを入力すると、 lpc(8) はこれを受け付けます。コマンドが入力されなかった場合は、 lpc(8) は対話モードに入り、 `exit`、`quit`、または、 ファイル終端文字が入力されるまでコマンドの入力ができます。

9.6. 標準スプーラの代替品

このマニュアルを最初から通読されている方ならば、ここまでで、 FreeBSD 付属の LPD スプーリングシステムに関して知っておくべきことすべてを学ばれたことと思います。
多分、このシステムにあるたくさんの欠点について認識できたことでしょう。 そこから "(FreeBSD 上で動作する) スプーリングシステムには他にどのようなものがあるのか" という疑問が自然と湧いてきます。

LPRng

"次世代 LPR" を称するLPRng は、 PLP を完全に書き換えたものです。 Patrick Powell と Justin Mason (PLP の主要な管理者) が共同で LPRng を作成しました。 LPRng の本サイトは http://www.lprng.org/ です。

CUPS

CUPS (the Common UNIX Printing System) は、UNIX® ベースのオペレーティングシステムに対して、 移植性の高い印刷レイヤを提供します。 CUPS は Easy Software Products によって、すべての UNIX® ベンダとユーザに、標準的な印刷ソリューションを普及するために開発されています。

CUPS は、プリントジョブとキューを管理する基盤として Internet Printing Protocol (IPP) を使っています。機能は限定されますが、 ラインプリンタデーモン (LPD)、サーバーメッセージブロック (SMB) や AppSocket (JetDirect とも呼ばれています) プロトコルにも対応しています。 CUPS は、UNIX® に現実的なプリント機能を備えるため、ネットワークプリンタの検索、 PostScript プリンタ記述言語 (PPD) に基づいた印刷オプションを追加します。

CUPS のメインサイトは http://www.cups.org/ です

HPLIP

HPLIP (the HP Linux® Imaging and Printing system) は、 HP アプライアンス用に HP が開発した、プリンタ、スキャナ、ファックスへの対応のためのプログラム群です。
このプログラムでは、印刷機能において CUPS 印刷システムをバックエンドとして利用しています。

HPLIP のメインサイトは、 http://hplipopensource.com/hplip-web/index.html です。

9.7. トラブルシューティング

lptest(1) を使った簡単なテストをおこなった結果、 正しい出力を得られずに、以下に示すような出力が得られるかもしれません。

しばらくしたら出力される、または、 紙の全体が出てこない

プリンタは上で示されたような印字を おこなったのですが、しばらくして止まってしまい、動かなくなってしまいました。 印字された結果をプリンタから取り出すためには、 プリンタにある PRINT REMAINING ボタン、または、FORM FEED ボタンを押す必要があるようです。

この場合は、 おそらくジョブはプリントをする前に更にデータが送られてこないか待ち続けているのでしょう。
この問題を解決するためには、プリンタに FORM FEED 文字（あるいは特定の必要な文字コード）を送るテキストフィルタを使ってください。
プリンタ内部に残ったデータをプリンタにすぐに印字させるには、 普通はこれで十分です。
次のジョブが前のジョブの最終ページの中央の どこかから印字を開始させないためにも、紙の途中で印字のジョブが終了したかどうかを確認するのは有益です。

シェルスクリプト /usr/local/libexec/if-simple を次のように変更して、プリンタへジョブを送信した後に FORM FEED 文字を印字させるようにします。

```
#!/bin/sh
#
# if-simple - Simple text input filter for lpd
# Installed in /usr/local/libexec/if-simple
#
# Simply copies stdin to stdout.  Ignores all filter arguments.
# Writes a form feed character (\f) after printing job.

/bin/cat && printf "\f" && exit 0
exit 2
```

"階段効果" が現れた

出力された紙には次のように印字されていました。

```
!"#$%&'()*+,-./01234
                "#$%&'()*+,-./012345
                              #$%&'()*+,-./0123456
```

あなたは「階段効果」 の新たなる犠牲者になってしまいました。この原因は、

改行を表わすべき文字がなんであるか の解釈が混乱していることにあります。UNIX® スタイルのオペレーティングシステムでは、改行文字は ASCII コード 10 の line feed (LF) の 1 文字が使われています。MS-DOS® や OS/2® などは ASCII コード 10の LF と、ASCII コード 13 の文字 (carriage return または CR) をペアで使います (訳注: Macintosh では CR のみで表現されています)。大抵のプリンタでは、 改行を表わすために MS-DOS® の慣習にしたがいます。

FreeBSD で印字する場合、印字したテキストは LF 文字だけ が使われていました。プリンタでは LF 文字を見つけると、紙を 1 行分送り出しました。しかし、 次の文字を印字するための紙の水平方向の位置は維持されました。すなわち、CR 文字が意味することは、次の文字を印字する位置を紙の左端に動かすことです。

FreeBSD がプリンタに動作をして欲しいと思っている動作を以下に示します。

プリンタが CR を受け取ったとき	CR 動作 (復帰) をおこなう
プリンタが LF を受け取ったとき	CR + LF 動作 (復帰、改行) をおこなう

このように動作させるための方法がいくつかあります。

- これらの文字の解釈を変えるために、
  プリンタの設定スイッチかコントロールパネルを操作する方法。
  どのようにして設定をするかはプリンタのマニュアルを参照してください。

FreeBSD 以外のオペレーティングシステムを切り替えて使う場合、 CR と LF 文字の解釈をそのオペレーティングシステムで使われているようにプリンタを再設定する必要があるかもしれません。 以下に示す解決方法のいずれかを選ぶのがよいかもしれませんね。

- 自動的に LF を CR+LF に変換してくれる FreeBSD 用のシリアルドライバを入手する方法。
  もちろん、このドライバはプリンタ専用に接続される シリアルポート のみで動作します。
  この機能を許可するためには、 `ms#` 項目を使い、 対象プリンタの /etc/printcap ファイルで`onlcr` モードを設定します。
- LF 文字の扱いを一時的に変更するための エスケープコード をプリンタに送る方法。
  プリンタがサポートしているかもしれないエスケープコード については、
  プリンタのマニュアルを参照してください。 適切なエスケープコードが見つかったら、
  最初にそのコードを送り、次にプリントジョブを送信
  するようにテキストフィルタを変更してください。

  次に、Hewlett Packard 社の PCL エスケープコードに対応しているプリンタのためのテキストフィルタの例を示します。 このフィルタでは、プリンタ に LF 文字を LF と CR の 2文字として扱わせます。 その後に、プリンタにジョブを送ります。最後に、ジョブの最終ページの紙を排出するため、FROM FEED 文字を送ります。このフィルタは Hewlett Packard 社のほとんどすべてのプリンタで機能するはずです。

```
#!/bin/sh
#
# hpif - Simple text input filter for lpd for HP-PCL based printers
# Installed in /usr/local/libexec/hpif
#
```

```
# Simply copies stdin to stdout.  Ignores all filter arguments.
# Tells printer to treat LF as CR+LF.  Ejects the page when done.

printf "\033&k2G" && cat && printf "\f" && exit 0
exit 2
```

ホスト orchid の /etc/printcap の例を以下に示します。ここには、一番目のパラレルポートにプリンタ (Hewlett Packard LaserJet 3Si) が一台接続されており、そのプリンタ名は teak です。

```
#
#  /etc/printcap for host orchid
#
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
        :lp=/dev/lpt0:sh:sd=/var/spool/lpd/teak:mx#0:\
        :if=/usr/local/libexec/hpif:
```

LF を CR+LF に置き換える cat コマンドを作る方法も当然考えられます。そして、このコマンドと、*if-simple* の cat の部分を置き換えればよいわけです。具体的にどのようにするかは、 読者への練習問題としましょう。

各行が重ね書きされてしまった

プリンタは紙送りをまったくしませんでした。
テキストすべての行がある行の上で重ねて印字されてしまいました。

この問題は、 階段現象とは "正反対" な問題で、 ほとんどまれにしか起こりません。FreeBSD では行末として扱われる LF 文字が、紙の左端に印字位置を復帰しますが、 紙送りはしない CR 文字として扱われています。

プリンタの設定スイッチかコントロールパネルを使って、 LF と CR の文字を次のような解釈をするようにしてください。

プリンタが受け取ったとき	プリンタがおこなう
CR	CR 動作 (復帰)
LF	CR + LF (復帰、改行)

LF を CR+LF に置き換える cat コマンドを作る方法も当然考えられます。そして、このコマンドと、 if-simple の cat の部分を置き換えればよいわけです。具体的にどのようにするかは、 読者への練習問題としましょう。

プリンタが文字を紛失してしまう

印字しているのですが、 各行の 2 〜 3 文字が印字されません。 プリンタを動かせば動かすほど、もっとたくさんの文字が紛失されていき、 この問題は更に悪くなっていくかもしれませんでした。

この問題は、 シリアルポートを通してコンピュータから送られてくるデータの速度に、プリンタがついていけないことに起因します (この問題は、パラレルポートに接続されたプリンタでは発生することはありません)。 この問題を克服する方法が2つあります。

- プリンタが XON/XOFF のフロー制御をサポートしている場合は、項目 ms# で ixon モードをセットして、FreeBSD にこの機能を使用させてください。
- プリンタが Request to Send / Clear to Send ハードウェアハンドシェイク (通称 RTS/CTS) をサポートしている場合は、項目 ms# で crtscts モードをセットして下さい。それから、プリンタとコンピュータを接続しているシリアルケーブルがハードウェアフロー制御用に正しく配線されたものかどうかを確認してください。

プリンタは意味不明な文字列を印字した

プリンタはランダムなゴミのように見えるものを印字しましたが、意図したテキストは印字してくれませんでした。

この問題は、通常、シリアルポートに接続したプリンタでの通信パラメータの誤りからくる前項とは別の症状です。br 項目の通信速度と ms# 項目を再確認してください。また、プリンタでの設定が /etc/printcap ファイルで設定した内容と一致しているかどうかも確認してください。

simple-if のような単純なフィルタだけの状態で、日本語を含むテキストを印字しようとした場合にも、シリアルポート、パラレルポートの使用に関係なく、このような症状は見られます。日本語プリンタの場合、漢字コードそのものを送信しただけでその漢字を印字してくれるものは、少なくとも訳者は見たことがありません。漢字を印字するための制御コードを別途送信するフィルタが必要となります。また、そのようなフィルタを使用していても、そのフィルタが想定してる漢字コードと異なった文書をプリントしようとしたときもこのような症状は出ます。もちろん、これはプリンタ用の言語を持たないプリンタの話で、PostScript® プリンタなどにプレインテキストを送信しても、日本語対応、非対応に関らず、意味不明な文字列が印字される (もしくは、何も印字されない) ことでしょう。

何も起きない

もしプリンタが何の動作もしないのであれば、ハード的な問題ではなく、多分 FreeBSD の中に問題があります。/etc/printcap ファイルで、デバッグしているプリンタのエントリに (lf 項目で) ログファイルを取るように設定を追加してください。たとえば、プリンタ rattan 用のエントリの項目 lf は次のようになります。

```
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :sh:sd=/var/spool/lpd/rattan:\
        :lp=/dev/lpt0:\
        :if=/usr/local/libexec/if-simple:\
        :lf=/var/log/rattan.log
```

次に、もう一度印字をおこなってみます。そして、発生したと思われるエラーメッセージを見るためにログファイル (上記の例では、/var/log/rattan.log) を調べます。そこで見られたメッセージを元に、問題を解決してみてください。

項目 lf が指定されていない場合、LPD はデフォルトのログファイルとして /dev/console

を使います。

Chapter 10. Linux® バイナリ互換機能

10.1. この章では

FreeBSD は、Linux® とのバイナリ互換機能を提供しています。このバイナリ互換機能を使うことで、ユーザは、ほとんどの Linux® バイナリを変更することなく、FreeBSD システム上にインストールして実行できるようになります。ある状況においては Linux® バイナリを Linux® で動かすよりも FreeBSD で動かすほうが良いパフォーマンスが出るという報告もあります。

しかしながら、いくつかの Linux® に特有なオペレーティングシステムの機能は FreeBSD ではサポートされていません。たとえば、仮想 8086 モードを有効にするような i386™ 特有の呼び出しを過度に使う Linux® バイナリは FreeBSD では動きません。

64 ビットの Linux® バイナリ互換機能は、 FreeBSD 10.3 で追加されました。

この章を読むと、以下のことがわかります。

- FreeBSD システムで Linux® バイナリ互換機能を有効にする方法。
- Linux® 共有ライブラリを追加する方法。
- Linux® アプリケーションを FreeBSD システムにインストールする方法
- FreeBSD における Linux® 互換機能の実装の詳細。

この章を読む前に、以下のことを理解しておく必要があります。

- サードパーティ製ソフトウェア のインストール方法

10.2. Linux® バイナリ互換機能の設定

Linux® ライブラリは、デフォルトでは FreeBSD にインストールされません。また、Linux® バイナリ互換機能も、デフォルトでは有効ではありません。Linux® ライブラリは、手動もしくは FreeBSD Ports Collection を使ってインストールできます。

port を構築する前に、`linux` カーネルモジュールを読み込んでください。このモジュールを読み込んでいないと、構築に失敗してしまいます。

```
# kldload linux
```

64 ビットの互換機能を使うには、以下を実行してください。

```
# kldload linux64
```

以下のようにしてモジュールが読み込まれていることを確認してください。

```
% kldstat
```

```
      Id Refs Address    Size     Name
      1    2 0xc0100000 16bdb8   kernel
      7    1 0xc24db000 d000     linux.ko
```

Linux® ライブラリおよびバイナリの基本セットを FreeBSD システムにインストールする最も簡単な方法は、 emulators/linux_base-c7 package または port を使う方法です。port をインストールするには、 以下のコマンドを実行してください。

```
# pkg install emulators/linux_base-c7
```

起動時から Linux® 互換機能を有効にする場合には、 /etc/rc.conf に以下の行を追加してください。

```
linux_enable="YES"
```

64 ビットのコンピュータでは、 /etc/rc.d/abi により 64 ビット互換のためのモジュールは自動的に読み込まれます。

Linux® バイナリ互換機能のレイヤには、(64 ビット x86 ホストにおける) 32 および 64 ビット Linux バイナリのサポートが追加されたため、
エミュレーション機能をカスタムカーネルに静的にリンクする必要はありません。

10.2.1. 手動によるライブラリの追加のインストール

Linux® バイナリ互換機能を設定した後に、Linux® アプリケーションが必要な共有ライブラリが存在しないというエラーを出した場合には、 Linux® バイナリがどの共有ライブラリを必要としているかを確認して、 手動でインストールしてください。

Linux® システムで、ldd を使うことにより、アプリケーションが必要とする共有ライブラリを調べることができます。 たとえば、linuxdoom が必要とする共有ライブラリを調べるには、 Doom がインストールされている Linux® システム上で、以下のコマンドを実行してください。

```
% ldd linuxxdoom
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
```

Linux® システムでの出力の最後のカラムに表示されているすべてのファイルを FreeBSD システムの /compat/linux の下にコピーしてください。コピーしたら、最初のカラムに示されるファイル名でコピーしたファイルに対してシンボリックリンクを張ってください。 この例では、FreeBSD システムで以下のようになります。

```
/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
```

ldd の出力の最初のカラムに表示されているメジャーバージョンが同じ Linux® 共有ライブラリが既にインストールされている場合は、
最後のコラムにある名前のファイルを新たにコピーする必要はありません。
既にあるライブラリで動作するはずです。 ただ、新しいバージョンの共有ライブラリがある場合には、
コピーすることをお奨めします。 新しいライブラリにシンボリックリンクを変更したら、
古いライブラリは削除してかまいません。

たとえば、以下のライブラリがすでに FreeBSD システムに存在するとします。

```
/compat/linux/lib/libc.so.4.6.27
/compat/linux/lib/libc.so.4 -> libc.so.4.6.27
```

そして、ldd の出力が以下のように、 バイナリが新しいバージョンを必要とする場合を考えます。

```
libc.so.4 (DLL Jump 4.5pl26) -> libc.so.4.6.29
```

存在しているライブラリの最後の番号が 1 つか 2 つ古いだけなので、
わずかに古いライブラリでもプログラムは動作するはずです。 しかしながら、libc.so
を新しいバージョンに置き換えるのが安全です。

```
/compat/linux/lib/libc.so.4.6.29
/compat/linux/lib/libc.so.4 -> libc.so.4.6.29
```

通常は、Linux® のバイナリが必要とする共有ライブラリを探す必要があるのは、 FreeBSD のシステムに Linux® のプログラムをインストールする最初の数回だけです。 それが過ぎれば、十分な Linux® の共有ライブラリがシステムに存在するので、 新しくインストールした Linux® のバイナリも追加の作業をせずに動作させることができるようになります。

10.2.2. Linux® の ELF バイナリのインストール

ELF のバイナリを使うためには、 追加の作業が必要です。 マークのない (unbranded) ELF バイナリを実行しようとすると、 以下のようなエラーメッセージが表示されてしまうことでしょう。

```
% ./my-linux-elf-binary
ELF binary type not known
Abort
```

FreeBSD のカーネルが FreeBSD の ELF バイナリと Linux® のバイナリとを見分けられるようにするために、brandelf(1) を以下のようにして使ってください。

```
% brandelf -t Linux my-linux-elf-binary
```

GNU のツール群が ELF バイナリに自動的に適切なマークを付加するようになったので、この作業は通常必要ありません。

10.2.3. Linux® RPM ベースのアプリケーションのインストール

Linux® RPM ベースのアプリケーションをインストールするには、最初に archivers/rpm4 package または port をインストールしてください。インストールすると、このコマンドを root 権限で使うことで、.rpm をインストールできます。

```
# cd /compat/linux
# rpm2cpio < /path/to/linux.archive.rpm | cpio -id
```

必要に応じて、インストールした ELF バイナリに brandelf を実行してください。綺麗にアンインストールできないかもしれませんので注意してください。

10.2.4. ホストネームリゾルバの設定

DNS がうまく動作しなかったり、以下のようなエラーメッセージが表示される場合は、/compat/linux/etc/host.conf ファイルを以下のように設定する必要があります。

```
resolv+: "bind" is an invalid keyword resolv+:
"hosts" is an invalid keyword
```

ファイルの内容を以下のように設定してください。

```
order hosts, bind
multi on
```

この設定では /etc/hosts を最初に検索し、次に DNS を検索するように指定します。/compat/linux/etc/host.conf が存在しない場合には、Linux® アプリケーションは /etc/host.conf を使用しようとし、FreeBSD の文法とは互換性がないと警告を出力します。/etc/resolv.conf を利用してネームサーバの設定をしていない場合には、bind を削除してください。

10.3. 高度なトピックス

この節では、Linux® バイナリ互換機能がどのような仕組みで動作をしているかを説明します。以下の文章は FreeBSD chat メーリングリスト に投稿された Terry Lambert (tlambert@primenet.com) 氏のメール (Message ID: <199906020108.SAA07001@usr09.primenet.com>) をもとにしています。

FreeBSD は、"実行クラスローダ (execution class loader)" と呼ばれる抽象的な機構を持っています。これは execve(2) システムコールへの楔という形で実装されています。

歴史的には、UNIX® のローダはマジックナンバー (一般的にはファイルの先頭の 4 ないし 8 バイトの部分) の検査を行ない、システムで実行できるバイナリかどうかを検査し、もしそうならバイナリローダを呼び出すというようになっていました。

もし、そのシステム用のバイナリでない場合には、execve(2)
システムコールの呼び出しは失敗の戻り値を返し、
シェルがシェルコマンドとして実行しようと試みていたわけです。この仮定は
"現在利用しているシェルがどのようなものであっても"デフォルトでした。

後に sh(1) に変更が加えられ、先頭の 2 バイトを検査した結果 :\n であれば代わりに csh(1) を呼び出す、というようになりました。

FreeBSD は、単一のローダではなく、ローダの一覧を走査します。
動作しているシェルインタプリタもしくはシェルスクリプトとして、 該当するものが存在しなければ、#!
ローダが用いられます。

Linux® ABI をサポートするため、FreeBSD は ELF バイナリを示すマジックナンバを確認します。 ELF
ローダは、特殊なマーク *(brand)* があるかどうか探します。このマークとは、ELF
イメージのコメントセクションのことです。SVR4/Solaris™ の ELF
バイナリには、このセクションは存在しません。

Linux® バイナリを実行するためには、brandelf(1) を使って Linux のマークが
付けられていなければなりません。

```
# brandelf -t Linux file
```

ELF ローダが Linux マークを確認すると、ローダは proc 構造体内の
ある一つのポインタを置き換えます。システムコールは全て、
このポインタを通してインデックスされます。さらに、そのプロセスには Linux®
カーネルモジュールに必要なシグナルトランポリンコード (訳注: シグナルの伝播を実現するコード)
用の特殊なトラップベクタの設定や、 他の (細かな) 調整のための設定が行なわれます。

Linux® システムコールベクタは、さまざまなデータに加えて sysent[]
エントリーのリストを含んでおり、 それらのアドレスはカーネルモジュール内にあります。

Linux® バイナリがシステムコールを発行する際、トラップコードは proc
構造体を用いてシステムコール関数ポインタを 解釈します。そして FreeBSD ではなく Linux®
用のシステムコールエントリポイントを得るわけです。

Linux® モードは状況に応じて
ファイルシステム本来のルートマウントポイントを置き換えてファイルの参照を行ないます。
これは、union を指定してマウントされたファイルシステムが行なっていることと同じです。
ファイルを検索する際にはまず /compat/linux/original-path
を調べます。見つけられなかったときには、/original-path を調べます。
こうすることで、他のバイナリを要求するバイナリの実行を可能にしています。 たとえば、Linux®
用ツールチェインは Linux® ABI サポート環境下で完全に動作します。 またこれは、もし対応する
Linux® バイナリが存在しない場合に Linux® バイナリが FreeBSD バイナリをロードしたり、
実行したりすることが可能であること、 その Linux® バイナリに自分自身が Linux®
上で実行されていないことを 気付かせないようにする目的で、uname(1) コマンドを /compat/linux
ディレクトリに置くことができる、 ということを意味します。

要するに、Linux® カーネルが FreeBSD カーネルの内部に存在しているわけです。
カーネルによって提供されるサービス全ての実装の基礎となるさまざまな関数は FreeBSD

システムコールテーブルエントリと Linux®
システムコールテーブルエントリの両方で共通に利用されています。
これらにはファイルシステム処理、仮想メモリ処理、シグナル伝送、 System V IPC が含まれますが、 FreeBSD バイナリは FreeBSD グルー (訳注: glue; 二者の間を仲介するという意味) 関数群、 そして Linux® バイナリは Linux® グルー関数群を用いる、 という点だけが異なります。 FreeBSD の グルー関数群は、 カーネルの中に静的にリンクされ、 Linux® の
グルー関数群は静的にリンクすることも、
カーネルモジュールを介して利用することもできるようになっています。

技術的には、これはエミュレーションではなく、 ABI の実装です。 よく "Linux® エミュレーション "と呼ばれるのは、 この機能が初めて実装された頃、 この機能を表現する言葉がなかったためです。 コードをコンパイルしてはいないので、 FreeBSD 上で Linux® バイナリを実行するという表現は、 厳密に考えると適切ではありません。

Part III: システム管理

以下の章では、FreeBSD のシステム管理の面について書かれています。
各章のはじめでは、その章で学ぶ内容や、
読者が実際に取り組む前に知っておくべきことについて説明します。

各章は、必要になった時に個別に参照できるように構成されています。
どの順番で読んでも構いませんし、FreeBSD を使うのに、
すべてを読み通す必要がある、というわけでもありません。

Chapter 11. 設定とチューニング

11.1. この章では

システムを正しく設定することは、メンテナンスや将来の更新の際の作業の量を減らします。
この章では FreeBSD システムの管理上の設定の側面について記述します。

またこの章では FreeBSD システムのパフォーマンスを最適化する チューニングについても記述します。

この章を読むと、以下のことがわかります。

- rc.conf の設定と /usr/local/etc/rc.d スタートアップシステムの基礎
- ネットワークデバイスに対する、仮想ホストの設定方法
- /etc ディレクトリ内のさまざまな設定ファ イルの使い方
- `sysctl` 変数を使った FreeBSD のチューニング方法
- ディスク性能のチューニング方法と、カーネルの制限の変更方法

この章を読む前に、以下のことをやっておくとよいでしょう。

- Unix と FreeBSD の基本を理解する (UNIX の基礎知識)。
- FreeBSD のソースコードを最新に保つこと (FreeBSD のアップデートとアップグレード) と、カーネルコンフィグレーションおよび構築の基礎 (FreeBSD カーネルのコンフィグレーション) に親しんでおく。

11.2. 中核となる設定

システムの設定情報が収められている主な場所は /etc/rc.conf です。
このファイルにはシステムの起動時にシステムの設定を行なうものをはじめ
多岐に渡る設定情報が含まれています。 そのファイル名はダイレクトに、それが rc*
ファイル群の設定情報であることを示しています。

管理者は /etc/defaults/rc.conf のデフォルトの設定を rc.conf ファイルにエントリを作ることで上書きすべきです。 デフォルトのファイルをそのまま /etc にコピーするのはやめるべきです。 それはデフォルト値であってサンプルではないのです。
システム固有のすべての変更は rc.conf ファイ ルの中でするべきです。

管理の手間を減らす為、クラスター化されたアプリケーションには
サイト共通の設定とシステム固有の設定を分離するさまざまな戦略が適用できます。
推奨されるアプローチは、サイト共通の設定は /etc/rc.conf.site のような別のファイルに置き、
それをシステム固有の設定情報しか含ませない /etc/rc.conf からインクルードすることです。

rc.conf は sh(1) によって読み込まれているので、これはじつに簡単に達成できます。 たとえば、

- rc.conf:

```
. rc.conf.site
hostname="node15.example.com"
```

```
network_interfaces="fxp0 lo0"
ifconfig_fxp0="inet 10.1.1.1"
```

- rc.conf.site:

```
defaultrouter="10.1.1.254"
saver="daemon"
blanktime="100"
```

rc.conf.site ファイルは rsync のようなプログラムを使うことで全システムに配布でき、一方 rc.conf ファイルはユニークなままを保つことができます。

システムを sysinstall(8) や make world 等で更新した場合 rc.conf ファイルは上書きされません。なのでシステムの設定情報が失われることもありません。

11.3. アプリケーションの設定

基本的に、インストールされたアプリケーションには独自の文法を持つ 固有の設定ファイルがあります。
これらのファイルがベースシステムから分離されているということは重要で、
このためパッケージ管理ツールによる配置と管理が容易になっています。

基本的に、それらのファイルは /usr/local/etc にインストールされます。
設定ファイルの数が多数にのぼるアプリケーションに対しては、
それら用にサブディレクトリが作られます。

通常、ports やパッケージがインストールされると
設定ファイルのサンプルが一緒にインストールされます。 大抵、識別のためにサフィックスとして
".default" がついています。 アプリケーションのための設定ファイルがまだ存在していなければ、
.defaults ファイルをコピーすることで作成できます。

/usr/local/etc/apache ディレクトリの例をご覧ください。

```
-rw-r--r--  1 root  wheel   2184 May 20  1998 access.conf
-rw-r--r--  1 root  wheel   2184 May 20  1998 access.conf.default
-rw-r--r--  1 root  wheel   9555 May 20  1998 httpd.conf
-rw-r--r--  1 root  wheel   9555 May 20  1998 httpd.conf.default
-rw-r--r--  1 root  wheel  12205 May 20  1998 magic
-rw-r--r--  1 root  wheel  12205 May 20  1998 magic.default
-rw-r--r--  1 root  wheel   2700 May 20  1998 mime.types
-rw-r--r--  1 root  wheel   2700 May 20  1998 mime.types.default
-rw-r--r--  1 root  wheel   7980 May 20  1998 srm.conf
-rw-r--r--  1 root  wheel   7933 May 20  1998 srm.conf.default
```

ファイルサイズの差から、srm.conf ファイルだけが変更されていることが分かります。 後に apache を更新した時にも、 この変更されたファイルは上書きされることはありません。

11.4. サービスの起動

一つのシステムでサービスをいくつも立ち上げているということは よくあることです。
それらには独自の立ち上げかたがあることがあり、 それぞれ有利な点があります。

Ports collection やパッケージからインストールしたソフトウェアは しばしば /usr/local/etc/rc.d にスクリプトを置き、 システムが起動した時には `start`、システムをシャッ トダウンする時には `stop` を引数にして実行します。 これは `root` で実行すべき、または `root` で起動することを期待されているシステム ワイドなサービスを起動する場合に推奨される方法です。
これらのスクリプトはパッケージの一部としてインストール時に記録され、
パッケージとともに削除されます。

/usr/local/etc/rc.d にある 一般的なスクリプトは次のようなものです。

```
#!/bin/sh
echo -n ' FooBar'

case "$1" in
start)
        /usr/local/bin/foobar
        ;;
stop)
        kill -9 `cat /var/run/foobar.pid`
        ;;
*)
        echo "Usage: `basename $0` {start|stop}" >&2
        exit 64
        ;;
esac

exit 0
```

このスクリプトはその目的を果すべく起動時に `start`、 シャットダウン時に `stop` をつけて呼ばれます。

サービスの中には固有のポートに接続を受けたときに inetd(8) から起動されるものもあります。
これはメールリーダサーバ (POP や IMAP 等) の場合によくあります。 これらのサービスは /etc/inetd.conf ファイルを編集することで有効化されます。 このファイルの編集に関する詳細は inetd(8) を見てください。

これらの他に /etc/rc.conf による有効化/無効化がカバーされていないサービスもあります。
それらは伝統的に /etc/rc.local にコマンドを書き込むことで実行されていました。 FreeBSD 3.1 にはデフォルトの /etc/rc.local は存在していません。 もし管理者によって作られていれば、
その時は一般的なやりかたとして認められるべきでしょう。 rc.local は最後の場所と考えられているということを 知っておいてください。
サービスを起動させるのにもっといい場所があるなら そこから始めてください。

/etc/rc.conf でその他のコマンドを実行しないでください。
そのかわり、デーモンの起動やブート時のコマンド実行は /usr/local/etc/rc.d にスクリプトを配置してください。

この他にサービスの起動に cron(8) を利用することもできます。 このアプローチには、cron(8) がそのプロセスを crontab の所有者権限で実行したり、サービスが非特権ユーザによって立ち上げられ管理されるなどといった有利な点が いくつもあります。

これで cron(8) の機能の利点を得ることができます。 日時の指定を @reboot で置き換えることでジョブはシステムがブートした直後、cron(8) が起動した時に実行されます。

11.5. バーチャルホスト

FreeBSD の非常にありふれた用途の一つにバーチャルサイトの ホスティングがあります。
これは一つのサーバがネットワークには複数のサーバとして現れるものです。
これは一つのネットワークインタフェイスに 複数のアドレスを割当てることで実現されます。

ネットワークインタフェイスは "真の" アドレスを 一つと "別名" のアドレスを複数持ちます。これらの別名は通常 /etc/rc.conf に別名のエントリを置くことで追加されます。

fxp0 インタフェイスへの別名のエント リは以下の様なものです。

```
ifconfig_fxp0_alias0="inet xxx.xxx.xxx.xxx netmask xxx.xxx.xxx.xxx"
```

別名のエントリは alias0 から始まり昇順に命名されなければなり ません (たとえば、_alias1, _alias2 の様になります)。設定プロセス は最初に欠けた番号のところで停まります。

別名のネットマスクの計算は重要ですが、幸いなことに非常に簡単です。
個々のインタフェイスについてそのネットワークのネットマスクを正しく
表現しているアドレスが必ず一つ必要です。
そのネットワークに所属しているそれ以外のアドレスのネットマスクは すべて 1 でなければなりません。

例として、fxp0 インタフェイスが二つ のネットワークに接続されているものを考えてみましょう。一つはネットマスクが 255.255.255.0 である 10.1.1.0 ネットワークで、もう一つはネットマスクが 255.255.255.240 である 202.0.75.16 ネットワークです。 システムは 10.1.1.0 には 10.1.1.1 として、202.0.75.20 には 202.0.75.17 として現れるようにします。

以下のエントリはネットワークインタフェイスを上述の環境に正しく 設定するものです。

```
ifconfig_fxp0="inet 10.1.1.1 netmask 255.255.255.0"
ifconfig_fxp0_alias0="inet 10.1.1.2 netmask 255.255.255.255"
ifconfig_fxp0_alias1="inet 10.1.1.3 netmask 255.255.255.255"
ifconfig_fxp0_alias2="inet 10.1.1.4 netmask 255.255.255.255"
ifconfig_fxp0_alias3="inet 10.1.1.5 netmask 255.255.255.255"
ifconfig_fxp0_alias4="inet 202.0.75.17 netmask 255.255.255.240"
ifconfig_fxp0_alias5="inet 202.0.75.18 netmask 255.255.255.255"
ifconfig_fxp0_alias6="inet 202.0.75.19 netmask 255.255.255.255"
ifconfig_fxp0_alias7="inet 202.0.75.20 netmask 255.255.255.255"
```

11.6. 設定ファイル

11.6.1. /etc のレイアウト

設定のための情報が含まれているディレクトリはたくさんあります。
それぞれ以下のものを含んでいます。

/etc	システム全般の設定情報。 ここにあるデータはシステム 固有のものです。
/etc/defaults	デフォルトのシステム設定ファイル。
/etc/mail	追加的な sendmail(8) の設定、他の MTA の設定ファイル。
/etc/ppp	ユーザモード、およびカーネルモードの ppp プログラムの設定。
/etc/namedb	named(8) のデータのデフォルトの置場。通常 boot ファイルはここに置かれ、 /var/db に置かれた他のデータを 参照するディレクティブを含みます。
/usr/local/etc	インストールされたアプリケーションの設定ファイル。 アプリケーションごとのサブディレクトリを含んでいることがあります。
/usr/local/etc/rc.d	インストールされたアプリケーションの起動/停止 スクリプト。
/var/db	永続的なシステム固有のデータファイル。 たとえば named(8) のゾーンファイル、データベースファイル等。

11.6.2. ホスト名

/etc/resolv.conf

/etc/resolv.conf は FreeBSD に インターネットドメインネームシステム (DNS)
にどのようにアクセスするかを指定します。

resolv.conf の最もよくあるエントリは

`nameserver`	リゾルバが問い合わせるべきネームサーバの IP アドレス。 サーバはリストの順に 3 番目まで問い合わせられます。
`search`	ホスト名をルックアップするための検索リスト。 通常、ローカルなホスト名のドメインから決定されます。
`domain`	ローカルドメイン名。

基本的な resolv.conf。

```
search example.com
nameserver 147.11.1.11
nameserver 147.11.100.30
```

search オプションと domain オプションは、 どちらか一方しか使ってはいけません。

DHCP を利用している場合、dhclient(8) は通常 resolv.conf を DHCP サーバから受け取った情報で書き換えます。

/etc/hosts

/etc/hosts は古きインターネットを 偲ばせるシンプルなテキストのデータベースです。 これはホスト名と IP アドレスをマッピングする DNS や NIS と組み合わせて使われます。 LAN でつながれているローカルな計算機は、 名前引きを簡単にするために named(8) サーバを立ち上げるかわりにここに書くことができます。 さらに /etc/hosts はインターネット名のローカルなレコードを提供し、
よくアクセスされる名前を外部に問い合わせるのを減らすためにも使えます。

```
# $FreeBSD$
#
# Host Database
# This file should contain the addresses and aliases
# for local hosts that share this file.
# In the presence of the domain name service or NIS, this file may
# not be consulted at all; see /etc/nsswitch.conf for the resolution order.
#
#
::1                     localhost localhost.my.domain myname.my.domain
127.0.0.1               localhost localhost.my.domain myname.my.domain

#
# Imaginary network.
#10.0.0.2               myname.my.domain myname
#10.0.0.3               myfriend.my.domain myfriend
#
# According to RFC 1918, you can use the following IP networks for
# private nets which will never be connected to the Internet:
#
#       10.0.0.0        -   10.255.255.255
#       172.16.0.0      -   172.31.255.255
#       192.168.0.0     -   192.168.255.255
#
# In case you want to be able to connect to the Internet, you need
# real official assigned numbers.  PLEASE PLEASE PLEASE do not try
# to invent your own network numbers but instead get one from your
# network provider (if any) or from the Internet Registry (ftp to
# rs.internic.net, directory `/templates').
#
```

/etc/hosts は、 次のようなごく簡単なフォーマットになっています。

```
[インターネットアドレス] [正式なホスト名] [別名1] [別名2] ...
```

例:

```
10.0.0.1 myRealHostname.example.com myRealHostname foobar1 foobar2
```

これ以上の情報は hosts(5) をあたってください。

11.6.3. ログファイルに関係する設定

syslog.conf

syslog.conf は syslogd(8) プログラムのための設定ファイルです。 これはどのタイプの syslog メッセージを対応する ログファイルに記録するかを指定します。

```
# $FreeBSD$
#
#       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) manual page.
*.err;kern.debug;auth.notice;mail.crit          /dev/console
*.notice;kern.debug;lpr.info;mail.crit;news.err /var/log/messages
security.*                                      /var/log/security
mail.info                                       /var/log/maillog
lpr.info                                        /var/log/lpd-errs
cron.*                                          /var/log/cron
*.err                                           root
*.notice;news.err                               root
*.alert                                         root
*.emerg                                         *
# uncomment this to log all writes to /dev/console to /var/log/console.log
#console.info                                   /var/log/console.log
# uncomment this to enable logging of all log messages to /var/log/all.log
#*.*                                            /var/log/all.log
# uncomment this to enable logging to a remote log host 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
!startslip
*.*                                             /var/log/slip.log
!ppp
```

```
*.*                                             /var/log/ppp.log
```

これ以上の情報は syslog.conf(5) のマニュアルページに あたってください。

newsyslog.conf

newsyslog.conf は、通常 cron(8) によって予定を決めて実行されるプログラム newsyslog(8) のための設定ファイルです。 newsyslog(8) は、 ログファイルをいつ保存して再編するかを決定します。 logfile は logfile.0 に移され、logfile.0 は logfile.1 に、そして以下同様に移されます。 また、ログファイルを gzip(1) 形式で保存することもできます。 この場合ファイル名は logfile.0.gz, logfile.1.gz の様になります。

newsyslog.conf はどのログファイルが管理され、どのくらいの期間保存され、 そしていつ touch されるかを指定します。 ログファイルはあるサイズに到達するか、ある決められた時刻・日時で再編されあるいは保存されます。

```
# configuration file for newsyslog
# $FreeBSD$
#
# filename          [owner:group]    mode count size when [ZB] [/pid_file] [sig_num]
/var/log/cron                           600  3     100  *     Z
/var/log/amd.log                        644  7     100  *     Z
/var/log/kerberos.log                   644  7     100  *     Z
/var/log/lpd-errs                       644  7     100  *     Z
/var/log/maillog                        644  7     *    @T00  Z
/var/log/sendmail.st                    644  10    *    168   B
/var/log/messages                       644  5     100  *     Z
/var/log/all.log                        600  7     *    @T00  Z
/var/log/slip.log                       600  3     100  *     Z
/var/log/ppp.log                        600  3     100  *     Z
/var/log/security                       600  10    100  *     Z
/var/log/wtmp                           644  3     *    @01T05 B
/var/log/daily.log                      640  7     *    @T00  Z
/var/log/weekly.log                     640  5     1    $W6D0 Z
/var/log/monthly.log                    640  12    *    $M1D0 Z
/var/log/console.log                    640  5     100  *     Z
```

これ以上の情報は newsyslog(8) のマニュアルページに あたってください。

11.6.4. sysctl.conf

sysctl.conf は rc.conf によく似ています。 値は変数=値のかたちでセットされます。 指定された値はシステムがマルチユーザモードに移行した後でセットされます。
すべての変数がこのモードで設定可能というわけではありません。

以下は sysctl.conf のサンプルで 致命的なシグナルを記録しないように、また Linux プログラムにそれらが実際は FreeBSD 上で動いていることを知らせる様に チューニングしています。

```
kern.logsigexit=0       # Do not log fatal signal exits (e.g. sig 11)
```

```
compat.linux.osname=FreeBSD
compat.linux.osrelease=4.3-STABLE
```

11.7. sysctl によるチューニング

sysctl(8) は稼働中の FreeBSD システムに変更を加えるためのインタフェイスです。これには経験を積んだ管理者用の TCP/IP スタックや仮想メモリシステムのパフォーマンスを劇的に改善する先進的なオプションが含まれます。500 を越えるシステム変数を sysctl(8) で読んだり セットしたりできます。

本質的には sysctl(8) の機能は次の二つ、システムの設定を読むことと変更することです。

読み取り可能なすべての変数を表示するには以下のようにします。

```
% sysctl -a
```

個々の変数、たとえば kern.maxproc を読むには以下のようにします。

```
% sysctl kern.maxproc
kern.maxproc: 1044
```

特定の変数をセットするには、直感的な文法 変数=値 を使ってください。

```
# sysctl kern.maxfiles=5000
kern.maxfiles: 2088 -> 5000
```

sysctl 変数の値は通常、文字列、数値、真偽値のいずれかです。(真偽値は yes の場合には 1 で no の場合には 0 です)。

11.8. ディスクのチューニング

11.8.1. sysctl 変数

vfs.vmiodirenable

vfs.vmiodirenable sysctl 変数のデフォルトは 1 (オン) で、0 (オフ) または 1 (オン) にセットすることができます。このパラメータはディレクトリがシステムによってどのようにキャッシュされるかを制御します。ほとんどのディレクトリは小さく、ファイルシステムにおいては単一フラグメント (典型的には 1K) であり、バッファキャッシュではさらに小さくなっています (典型的には 512 バイト)。しかしデフォルトモードで動作している時は、大量のメモリを搭載していてもバッファキャッシュは固定数のディレクトリしかキャッシュしません。この sysctl をオンにすると、バッファキャッシュが VM ページキャッシュを、ディレクトリをキャッシュするために使うことを可能にします。
これによる利点は、全てのメモリがディレクトリを

キャッシュするのに使えるようになるということです。
欠点は、キャッシュに使われる最小のメモリの大きさが 512 バイトではなく 物理ページサイズ (大抵は 4K) になることです。 多数のファイルを操作するサービスを稼動しているなら、常にこのオプションをオンにすることを推奨します。 そのようなサービスには、web キャッシュや大規模なメールシステム、 ニューズシステムなどが含まれます。
このオプションは一般にメモリを消費しますが、 性能を削減することはありません。
ただし実験して調べてみるべきでしょう。

`hw.ata.wc`

FreeBSD 4.3 では IDE のライトキャッシュがオフになりました。 これは IDE ディスクへの書き込み帯域幅を減らしてしまうことになりますが、
ハードドライブベンダに起因するデータの一貫性に関する
重大な問題のために必要なことだと考えられました。 基本的には、書き込み完了時期について IDE ドライブが嘘をつくという問題です。 IDE ライトキャッシュがオンであると IDE ハードドライブはデータを順番に書きこまないばかりか、
ディスクの負荷が高い時にはいくつかのブロックの書き込みを 無期限に延期してしまいます。
クラッシュや電源故障の場合、 ファイルシステムの重大な破壊をもたらします。
したがって私たちはデフォルトを安全側に変更しました。
残念ながらこれは大変な性能の低下をもたらし、
私たちはあきらめてこのリリース後にオンに戻しました。 `hw.ata.wc` sysctl 変数を見てデフォルトをチェックしてみるべきです。 もし IDE ライトキャッシュがオフになっていたら、 hw.ata.wc カーネル変数を 1 に戻すことでオンに戻すことができます。
これはブート時にブートローダから行わなければなりません。
カーネルがブートした後に行っても効果はありません。

詳しくは ata(4) を見てください。

11.8.2. ソフトアップデート

tunefs(8) プログラムはファイルシステムを細かくチューニングするのに使えます。このプログラムにはさまざまなオプションがありますが、ここではソフトアップデートをオンオフすることだけを考えま す。以下の様にして切り替えます。

```
# tunefs -n enable /filesystem
# tunefs -n disable /filesystem
```

ファイルシステムはマウントされているあいだは tunefs(8) で変更することができません。
ソフトアップデートを有効にする いい機会はシングルユーザモードでどのパーティションもマウントされていない時です。

FreeBSD 4.5 からは、ファイルシステム生成時に newfs(8) の `-U` オプションを使ってソフトアップデートを有効化できるようになりました。

ソフトアップデートはメタデータの性能、 主にファイルの作成と削除の性能を劇的に改善します。
すべてのファイルシステムでソフトアップデートを有効にすることを推奨します。
ソフトアップデートに関して、2 つの欠点を意識すべきです。 1 つめは、ソフトアップデートはクラッシュ時におけるファイルシス テムの一貫性は保証しますが、物理ディスクの更新が何秒か (1 分に達することもあります!) 遅れる可能性が高いことです。

システムがクラッシュした場合、より多くの作業結果が消えてしまうかもしれません。 2つめは、ソフトアップデート はファイルシステムブロックを解放するのを遅らせるということです。あるファイルシステム (たとえばルートファイルシステム) が満杯近くの時にそれに対する大規模な更新、たとえば make installworld をすると、空き領域を使い果たして更新が失敗してしまうことがあります。

11.9. Kernel 制限のチューニング

11.9.1. File/Process 制限

kern.maxfiles

kern.maxfiles はあなたのシステムの要求に応じて増減させることができます。この変数はあなたのシステムのファイル記述子の最大値を示します。ファイル記述子テーブルが溢れるような時には、システムメッセージバッファに頻繁に file: table is full と表示されます。これは、 dmesg コマンドで確認できます。

ファイル、ソケット、パイプ (fifo) は それぞれオープンされるとファイル記述子を一つ消費します。大規模なプロダクションサーバでは その時実行されているサービスの種類や数に応じてはあっさり数千のファイル記述子が必要になります。

kern.maxfile のデフォルト値はカーネル コンフィグレーションファイルの MAXUSERS オプションで決まります。kern.maxfiles は MAXUSERS の値に比例して増加します。カスタムカーネルをコンパイルする際は、このカーネルコンフィグレーションオプションをシステムの利用法に合わせて設定するとよいでしょう。カーネルは、この数値からほとんどの制限の初期値を 決定します。業務用マシンに、実際に256 名のユーザが一度に接 続することはないかもしれませんが、大規模なウェブサーバに必要なリソースは同程度になります。

FreeBSD 4.5 からは、 カーネルコンフィグレーションファイルで MAXUSERS を 0 に設定すると、システムの RAM 容量に基づいて適切なデフォルト値が選択されます。

11.9.2. ネットワークの制限

カーネルコンフィグレーションオプション NMBCLUSTERS は、そのシステムで利用可能なネットワーク mbuf の量を決定します。通信量の多いサーバで MBUF の量 が少ないと、FreeBSD の性能が低下してしまいます。クラスターつは およそ 2kB のメモリに対応しているので、1024だとカーネルメモリ から約 2 MB をネットワークバッファに予約することになります。どれだけ必要になるかを、簡単な計算で出すことができます。同時に最 大 1000接続までゆくウェブサーバがあり、それぞれの接続によって 受信バッファ 16kB と送信バッファ 16kB が消費されるなら、ウェブサーバをまかなうのに 32MB 程度のネットワークバッファが必要になります。経験的に有用な値は、それを 2 倍したものなので、 32MBx2 = 64MB/2K = 32768になります。

Chapter 12. FreeBSD の起動のプロセス

12.1. この章では

計算機を起動しオペレーティングシステムをロードするプロセスは、"ブートストラッププロセス" もしくは "ブート" と呼ばれます。FreeBSD の起動プロセスを使えば、システムをスタートするときに起きることをかなり柔軟にカスタマイズできます。
同じ計算機にインストールされた別のオペレーティングシステムを選択することもできますし、
同じオペレーティングシステムの異なるバージョンを選択することも、
インストールされた別のカーネルを選択することさえできます。

この章では、指定できる設定オプションついて詳しく説明します。FreeBSD カーネルがスタートし、デバイスを検出し、 init(8) を起動するまでに起きることすべてを含む FreeBSD の起動プロセスのカスタマイズ方法について説明します。 これは、起動メッセージのテキストの色が、明るい白から灰色に変わるまでに起きています。

この章を読むと、以下のことが分かります。

- FreeBSD のブートストラップシステムの構成およびそれらが互いにどう関係しているのか
- 起動プロセスを制御するために FreeBSD のブートストラップの各要素に付加できるオプション
- device hints の基本的な記述方法
- シングルユーザもしくはマルチユーザモードでの起動方法、および FreeBSD システムのシャットダウンの方法

この章では Intel x86 および amd64 システム上で動作する FreeBSD の起動プロセスだけを扱います。

12.2. FreeBSD の起動プロセス

計算機の電源を入れ、オペレーティングシステムをスタートさせるのには、
おもしろいジレンマがあります。定義により、 計算機は、オペレーティングシステムが起動するまでは、ディスクからプログラムを動かすことも含めて、何をどうすればよいかまったく知りません。
計算機はオペレーティングシステムなしにディスクからプログラムを実行することができず、
オペレーティングシステムのプログラムがディスク上にあるのなら、
どうやってオペレーティングシステムを起動するのでしょうか?

この問題はほらふき男爵の冒険という本の中に書かれている問題ととてもよく似ています。登場人物がマンホールの下に半分落っこちて、靴紐 (ブートストラップ) をつかんで自分を引っぱり、持ち上げるのです。計算機の黎明期には、ブートストラップという用語でオペレーティングシステムをロードする機構のことを指していました。 いまはこれを縮めて "ブート (起動)" と言います。

x86 ハードウェアでは、基本入出力システム (Basic Input/Output System: BIOS) にオペレーティングシステムをロードする責任があります。BIOS はハードディスク上のマスターブートレコード (Master Boot Record: MBR) を探します。 MBR はハードディスク上の特定の場所になければなりません。BIOS には MBR をロードし起動するのに十分な知識があり、

オペレーティングシステムをロードするために必要な作業の残りは、 場合によっては BIOS の助けを得た上で MBR が実行できることを仮定しています。

FreeBSD は古い標準の MBR、 または新しい GUID Partition Table (GPT) から起動できます。 GPT パーティションは、Unified Extensible Firmware Interface (UEFI) に対応したコンピュータで良く用いられます。 しかしながら、FreeBSD はレガシーな BIOS にのみに対応したコンピュータからも、gptboot(8) により、 GPT パーティションから起動できます。 UEFI からの直接の起動への対応は進行中です。

MBR 内部のコードは、 一般的に ブートマネージャ と呼ばれます。 とりわけユーザとの対話がある場合にそう呼ばれます。
通常ブートマネージャのもっと多くのコードが、ディスクの最初のトラック、またはファイルシステム上におかれます。 ブートマネージャの例としては、Boot Easy とも呼ばれる FreeBSD 標準のブートマネージャの boot0、多くの Linux® ディストリビューションが採用している GNU Grub 等があります。

GRUB のユーザは GNU-provided documentation を参照してください。

ディスク上にインストールされているオペレーティングシステムが 1 つの時は、MBR はディスク上の最初の起動可能な (アクティブな) スライスを探し、
そのスライスにあるコードを起動してオペレーティングシステムの残りをロードします。
ディスク上に複数のオペレーティングシステムが存在しているのなら、
複数のオペレーティングシステムの一覧を表示できて、
起動するオペレーティングシステムを選択できるような、
別のブートマネージャをインストールすることもできます。

FreeBSD のブートストラップシステムの残りは 3 段階に分かれます。 第 1 ステージは、計算機を特定の状態にするために必要なことだけを知っていて、 第 2 ステージを起動します。 第 2 ステージでは、第 3 ステージを起動する前に、 もう少しできることがあります。 第 3 ステージでオペレーティングシステムのロード作業を完了します。 起動作業が 3 段階に分かれているのは、 MBR がステージ 1 とステージ 2 で実行できるプログラムのサイズに制限を課しているからです。
これらの作業をつなぎ合わせることによって、 FreeBSD はより柔軟なローダ (loader) を提供しているのです。

その後カーネルが起動し、デバイスの検出と初期化を開始します。
そしてカーネルの起動が終わると、制御はユーザープロセスの init(8) へ移されます。init(8) はディスクが利用可能であることを確認し、 ファイルシステムのマウント、
ネットワークで利用するネットワークカードのセットアップ、
そしてブート時に起動されるように設定されたプロセスの起動、 といったユーザーレベルでのリソース (資源) 設定を行ないます。

この章では、これらのステージについてより詳細に、また、FreeBSD ブートプロセスにおける対話的な設定方法について説明します。

12.2.1. ブートマネージャ

MBR のブートマネージャのコードは起動プロセスの_第 0 ステージ_と呼ばれることがあります。
デフォルトでは、FreeBSD は boot0 を使います。

FreeBSD のインストーラがインストールする MBR は、/boot/boot0 を基にしています。boot0 のサイズと機能は、スライステーブルおよび MBR 末尾の識別子 0x55AA のため、446 バイトの大きさに制限されます。もし、boot0 と複数のオペレーティングシステムをインストールした場合、
起動時に以下のようなメッセージが表示されます。

例 *24. boot0* のスクリーンショット

```
F1 Win
F2 FreeBSD

Default: F2
```

他のオペレーティングシステムは、FreeBSD の後にインストールを行うと、既存の MBR を上書きしてしまいます。もしそうなってしまったら、もしくは既存の MBR を FreeBSD の MBR で置き換えるには、次のコマンドを使ってください。

```
# fdisk -B -b /boot/boot0 device
```

device は起動するデバイス名で、たとえば 1 番目の IDE ディスクは ad0、2 番目の IDE コントローラに接続されている 1 番目の IDE ディスクは ad2、1 番目の SCSI ディスクは da0 などとなります。MBR の設定をカスタマイズしたい場合は、boot0cfg(8) を参照してください。

12.2.2. 起動ステージ 1 と起動ステージ 2

概念上、第 1 ステージと第 2 ステージはハードディスクの同じ領域上の同一のプログラムの部分部分です。スペースの制約のため 2 つに分割されていますが、いつも一緒にインストールされます。FreeBSD のインストーラまたは bsdlabel は、両者を 1 つにまとめた /boot/boot をコピーします。

これらの 2 つのステージは、ファイルシステムの外部、起動スライスの最初のトラックに置かれ、先頭が最初のセクタにきます。boot0 またはその他のブートマネージャは、起動プロセスを続けるために必要なプログラムがそこにあると想定しています。

最初のステージの boot1 は、512 バイトの大きさでなければならないという制限があるので、非常に単純なプログラムです。このプログラムは boot2 を検索して実行するため、そのスライスの情報を保持する FreeBSD の *BSD* ラベルに関する最低限の情報だけを持っています。

次のステージの boot2 はもう少し高機能です。これは FreeBSD のファイルシステム上でファイルを見つける機能を持ちます。
実行するカーネルやローダを指定するための簡単なインタフェースを提供します。boot2 により起動される loader はさらに高機能で、起動設定が行なえる手段を提供します。ステージ 2 で起動プロセス中断した時には、次のようながインタラクティブなが画面が表示されます。

例 25. boot2 のスクリーンショット

```
>> FreeBSD/i386 BOOT
Default: 0:ad(0,a)/boot/loader
boot:
```

インストールされた boot1 と boot2 を変更するには、 bsdlabel を使ってください。 以下の例では、*diskslice* は起動するディスクとスライスで、たとえば最初の IDE ディスクの 1 番目のスライスは ad0s1 となります。

```
# bsdlabel -B diskslice
```

ad0 のようにディスク名だけを指定すると、 bsdlabel は、スライスを持たない "危険な専用モード"を作成してしまいます。
これはおそらく、あなたが望んでいることではないでしょうから、 Return キーを押す前に、 *diskslice* の部分を二重にチェックしてください。

12.2.3. 起動ステージ 3

loader は三段階の起動プロセスの最終段階です。 これは通常、ファイルシステム上の /boot/loader として存在しています。

loader は、 よりさまざまなコマンド群をサポートした強力なインタプリタによって提供される組み込みコマンド群を利用することで、 インタラクティブな設定手段となるように設計されています。

loader は初期化の際にコンソールとディスクの検出を行ない、 どのディスクから起動しているかを調べます。 そして必要な変数を設定してからインタプリタを起動し、 スクリプトからコマンドを送ったり手でコマンドを入力したりできます。

loader は次に /boot/loader.rc を読み込み、通常、変数の標準値を定義した /boot/defaults/loader.conf と、そのコンピュータにローカルに変数を定義した /boot/loader.conf を読み込みます。 loader.rc はそれらの変数にもとづき、 選択されたモジュールとカーネルをロードします。

loader は最後に、 標準設定で 10 秒のキー入力待ち時間を用意し、 入力がなければカーネルを起動します。 入力があった場合、コマンド群が使えるプロンプトが表示され、 ユーザは変数を調整したり、すべてのモジュールをアンロードしたり、 モジュールをロードしたりすることができます。 その後、最終的な起動や再起動へ移行します。 ローダの組み込みコマンドでは、 もっともよく使われる loader のコマンドをまとめています。 利用可能なコマンドをすべて知りたい場合には、 loader(8) を参照してください。

表 9. ローダの組み込みコマンド

変数	説明
autoboot *seconds*	*seconds* で与えられた時間内に入力がなければ、 カーネルの起動へと進みます。 カウントダウンを表示します。標準設定では 10 秒間です。
boot [-options] [kernelname]	すぐにカーネルの起動へ進みます。 オプション、カーネル名が指定されている場合は、 それらが使われます。 *unload* を実行後、 カーネル名をコマンドラインから指定することができます。 *unload* を実行しないと、 一度読み込まれたカーネルが使われます。 *kernelname* でパスが指定されていない時には、 */boot/kernel* および */boot/modules* から調べられます。
boot-conf	すべてのモジュールの設定を、 起動時と同じように指定された変数 (最も多いのは kernel) にもとづいて自動的に行ないます。 このコマンドは、変数を変更する前に、 最初に unload を行なった場合にのみ有効に働きます。
help [topic]	/boot/loader.help を読み込み、ヘルプメッセージを表示します。 topic に index が指定された場合、 利用可能な topic の一覧を表示します。
include *filename ...*	指定されたファイルを読み込み、行単位で解釈します。 エラーが発生した場合、 include の実行は直ちに停止します。
load [-t type] *filename*	指定されたファイル名のカーネル、 カーネルモジュール、あるいは type に指定された種類のファイルをロードします。 *filename* 以降に指定された引数はファイルへと渡されます。 *filename* でパスが指定されていない時には、 */boot/kernel* および */boot/modules* から調べられます。
ls [-l] [path]	指定された *path* にあるファイルを表示します。 *path* が指定されていなければ、ルートディレクトリを表示します。 -l が指定されていればファイルサイズも表示されます。
lsdev [-v]	モジュールがロード可能なすべてのデバイスを表示します。 もし -v が指定されていれば、 より詳細な出力がされます。
lsmod [-v]	ロード済みのモジュールを表示します。 -v が指定されていれば、 より詳細な内容が出力されます。

変数	説明
more *filename*	LINES 行を表示するごとに停止しながら指定されたファイルを表示します。
reboot	すぐにシステムを再起動します。
set *variable*, set *variable=value*	ローダの環境変数を設定します。
unload	すべてのロード済みモジュールを削除します。

次にあげるのは、ローダの実践的な使用例です。
普段使っているカーネルをシングルユーザモードで起動します。

```
boot -s
```

普段使っているカーネルとモジュールをアンロードし、　古いもしくは別のカーネルをロードするには、以下のように実行してください。

```
unload
load /path/to/kernelfile
```

インストール時のデフォルトカーネルを指定するには、完全修飾の　/boot/GENERIC/kernel を使ってください。　また、システムをアップグレードしたり、もしくはカスタムカーネルを設定した場合に、　直前にインストールされていたカーネルは、/boot/kernel.old/kernel で指定できます。

普段のカーネルで使っているモジュールを指定したカーネルでロードする場合は、　次のようにします。この場合は、完全修飾名を使う必要はありません。

```
unload
set kernel="mykernel"
boot-conf
```

カーネルの自動設定スクリプトをロードします。

```
load -t userconfig_script /boot/kernel.conf
```

12.2.4. 最終ステージ

カーネルがデフォルトの　loader　もしくは　loader　を迂回して　boot2　によって読み込まれると、起動フラグが調べられ、それに応じて動作が調整されます。起動時のカーネルオプション　には、良く使われる起動フラグがまとめられています。　他の起動フラグの詳細については、　boot(8) を参照してください。

表 *10.* 起動時のカーネルオプション

オプション	説明
-a	カーネル初期化中に、 ルートファイルシステムとしてマウントするデバイスを尋ねます。
-C	CDROM からルートファイルシステムを起動します。
-s	シングルユーザモードで起動します。
-v	カーネル起動時に、より詳細な情報を表示します。

カーネルの起動が完了すると、init(8) というユーザプロセスに制御が移されます。 これは /sbin/init、もしくは loader の init_path 変数で指定される場所にあります。
これは起動プロセスの最終ステージです。

起動シーケンスでは、 システム上で利用できるファイルシステムの一慣性を確認します。 もし UFS ファイルシステムにに問題があって fsck が不一致を修復できなければ、管理者が問題を直接解決できるように、init はシステムをシングルユーザモードへと移行させます。
問題がなければ、システムはマルチユーザモードに移行します。

シングルユーザモード

このモードには、ユーザが起動時に -s を指定した場合、あるいは loader で boot_single 変数を設定することによって移行します。 マルチユーザモードから shutdown now を呼び出すことでもこのモードに移行できます。
シングルユーザモードは、以下のメッセージで開始します。

```
Enter full pathname of shell or RETURN for /bin/sh:
```

ユーザが Enter を入力すると、 システムは Bourne シェルを起動します。
別のシェルを使うには、シェルのフルパスを入力してください。

シングルユーザモードは、
通常ファイルシステムの一貫性に問題があって起動できないシステムを修復したり、
起動設定ファイルの間違いを修正するために使われます。 また、root パスワードがわからなくなった場合に、 リセットするために使うことも出来ます。
シングルユーザモードのプロンプトは、
ローカルファイルシステムおよび設定ファイルへのアクセスを与えてくれますが、
ネットワーク接続は出来ません。

シングルユーザモードは、システムの修復には有用ですが、 システムが物理的に安全な場所になければ、セキュリティのリスクがもたらされます。
デフォルトでは、システムに物理的にアクセス可能なユーザは、
シングルユーザモードで起動後はシステムをすべてコントロールできます。

/etc/ttys でシステムの console が insecure に設定されている場合、システムはシングルユーザモードに移行する前に root のパスワードを入力するように求めます。 root パスワードがわからなくなった場合のリセット機能が無効になっている間は、
セキュリティ対策が必要となります。

例 26. /etc/ttys の insecure コンソール

```
# 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 コンソールとは、コンソールが物理的に安全でない (insecure) と考えられるため、root のパスワードを知る人だけがシングルユーザモードを使えるという意味です。

マルチユーザモード

init がファイルシステムが正常であると判断するか、ユーザがシングルユーザモードでのコマンドを終了し、exit を入力してシングルユーザモードを終了すると、システムはマルチユーザモードへ移行し、システムのリソースの設定を開始します。

リソース設定システムはデフォルト設定を /etc/defaults/rc.conf から、また、システム独自の細かな設定を /etc/rc.conf から読み込みます。そして /etc/fstab に記述されるシステムファイルシステムをマウントします。
その後、ネットワークサービス、さまざまなシステムデーモン、
そして最後に、ローカルにインストールされた package の起動スクリプトを実行します。

リソース設定システムについてもっと知りたい場合には、rc(8) を参照してください。また、/etc/rc.d にあるスクリプトを実行してみてください。

12.3. Device Hints

システムの最初のスタートアップ時に、loader(8) は device.hints(5) を読み込みます。このファイルにはカーネル起動の環境変数が格納されており、これらの環境変数は "device hints" と呼ばれることがあります。デバイスドライバは、デバイスを設定するために "device hints" を使用します。

起動ステージ 3 で説明されているように device hints はステージ 3 ブートローダプロンプトでも設定できます。変数は set を用いて追加したり、unset を用いて削除できます。show を用いて一覧を見ることもできます。/boot/device.hints に設定されている変数は、上書きすることもできます。ブートローダで設定した device hints の効果は一時的なものなので、次回起動するときには無効になります。

システムが起動すると、kenv(1) コマンドですべてのカーネル環境変数をダンプすることができます。

/boot/device.hints は 1 行につき一つの変数を設定でき、行頭の "#" はその行がコメントであることを示しています。書式は次の通りです。

```
hint.driver.unit.keyword="value"
```

ステージ 3 ブートローダ で設定するときの書式は次の通りです。

```
set hint.driver.unit.keyword=value
```

ここで、driver はデバイスドライバの名前、 unit はデバイスドライバのユニット番号、 keyword はヒントキーワードです。 キーワードは以下のようなオプションです。

- at: デバイスがどのバスに接続されているか指定します。
- port: 使用する I/O ポートの開始アドレスを指定します。
- irq: 使用する IRQ を指定します。
- drq: 使用する DMA チャネルを指定します。
- maddr: 使用する物理メモリアドレスを指定します。
- flags: デバイスに対してさまざまなフラグを設定します。
- disabled: 1 が設定されていると、そのデバイスは無効になります。

デバイスドライバはこのリスト以外の変数を設定できるかもしれませんし、
このリスト以外の変数を必要とするかもしれないので、
ドライバのマニュアルを読むことをおすすめします。 より多くの情報を知りたければ、device.hints(5), kenv(1), loader.conf(5) および loader(8) を参照してください。

12.4. シャットダウン動作

shutdown(8) を用いてシステムを意図的にシャットダウンした場合、 init(8) は /etc/rc.shutdown というスクリプトの実行を試みます。 そして、すべてのプロセスへ TERM シグナルを送り、続いてうまく終了できなかったプロセスへ KILL シグナルを送ります。

電源管理機能を持ったシステムで稼働している FreeBSD では shutdown -p now によって、直ちに電源を落とすことができます。FreeBSD システムを再起動するには、 shutdown -r now を実行してください。 shutdown(8) を実行するには、 root か、operator のメンバでなければなりません。halt(8) や reboot(8) を利用することもできます。
より多くの情報を得るために、それらのマニュアルページや shutdown(8) を参照してください。

グループのメンバを変更するには、「この章では」 を参照してください。

電源管理機能には acpi(4) がモジュールとして読み込まれるか、
カスタムカーネルにコンパイルされて静的に組み込まれている必要があります。

Chapter 13. セキュリティ

13.1. この章では

物理的もしくは仮想的に関わらず、 セキュリティは幅広いトピックであり、
業界全体がセキュリティとともに成長しています。
システムおよびネットワークを安全にする標準的な方法は数多く文書化されており、 FreeBSD
のユーザも、 攻撃や侵入者から守る方法を理解しなければなりません。

この章では、セキュリティの基礎や技術について説明します。 FreeBSD
システムは、複数のレイヤに関連するセキュリティを提供します。
そして、安全性を高めるためにサードパーティ製のユーティリティを利用することもできます。

この章を読むと、以下のことがわかります。

- FreeBSD における基本的なシステムセキュリティの考え方
- FreeBSD で利用できるさまざまな暗号化手法
- ワンタイムパスワード認証の設定方法
- inetd(8) と組み合わせて TCP Wrappers を設定する方法
- FreeBSD における Kerberos の設定方法
- IPsec を設定して VPN を構築する方法
- FreeBSD にける OpenSSH の設定および使用方法
- ファイルシステム ACL (アクセス制御リスト) の使用方法
- Ports Collection からインストールされたサードパーティ製ソフトウェア packages を Portaudit を使って監査する方法
- FreeBSD セキュリティ勧告の利用方法
- プロセスアカウンティングがどのようなものか、 FreeBSD 上で有効にする方法について
- リソース制限データベースとは何か、 この仕組みを使ったユーザ資源の管理方法

この章を読む前に、次のことが必要になります。

- FreeBSD およびインターネットの基本概念の理解

13.2. はじめに

セキュリティを高めることはすべての人の責任です。
システムに弱い侵入ポイントが存在すると、侵入者は重要な情報を得たり、
ネットワーク全体に被害を及ぼすことができるようになります。
多くのセキュリティのトレーニングでは、 情報システムの機密性 (confidentiality)、 完全性 (integrity) および可用性 (availability) を意味するセキュリティの 3 要素である CIA が取り扱われます。

CIA の 3 要素は、 コンピュータセキュリティの基本となる考えです。
顧客やエンドユーザは、データのプライバシーを期待します。 彼らは、データが変更されないことや、
情報が隠されていることを期待します。 彼らはまた、いつでも情報にアクセスできることを期待します。

これらは、システムの機密性、完全性、可用性を構成します。

セキュリティのプロフェッショナルは、CIA を守るために、多層防衛の戦略を採用します。
この多層防衛戦略ではセキュリティのレイアを複数用意することで、 一つのレイヤが破られても、
セキュリティシステム全体が破られることを防ぎます。
システムの管理者は、ファイアウォールを単に有効にするだけではなく、
ネットワークもしくはシステムを安全に保つ必要があります。 アカウントを監査し、バイナリの完全性、
悪意のあるツールがインストールされていないことを確認する必要があります。 このために、
管理者は脅威がどのようなものかを理解する必要があります。

13.2.1. 脅威

コンピュータセキュリティおける脅威とは何でしょうか？ 長年、脅威はリモートの攻撃者、
すなわち遠隔からの許可のないシステムへのアクセスを企てる人々と考えられていました。
今日では、この定義は従業員、悪意のあるソフトウェア、
不正なネットワークデバイス、自然災害、セキュリティの脆弱性、
そして競合する会社でさえも含めるように拡張されています。

毎日、数千ものシステムおよびネットワークが攻撃され、
数百ものシステムが許可なくアクセスされています。
簡単なアクシデントといったものから、リモートからの攻撃、
産業スパイであったり、以前働いていた従業員からの攻撃といったケースもあります。
システムのユーザとしては、 間違いがセキュリティ違反に繋がった場合には、
可能性のある問題をセキュリティチームに報告することが重要です。 管理者としては、脅威を把握し、
その脅威の影響を小さくするように準備をしておくことが重要です。

13.2.2. ボトムアップアプローチ

セキュリティを考える上で、 しばしばボトムアップアプローチが一番良い方法となります。
この考えでは、管理者が基本的なアカウント、システム設定を行ってから、
サードパーティ製ユーティリティの設定、 そしてネットワークレイヤに設定を広げていきます。
システムポリシーおよび手続きを行う上では、 このような設定の側面があります。

ビジネスの多くの環境では、
使用するデバイスの設定に対するセキュリティポリシがすでに策定されています。
このポリシには、最低限エンドユーザのワークステーション、
デスクトップ、携帯電話やラップトップといったモバイルデバイス、および
製品および開発サーバの両方に対するセキュリティの設定が含まれているべきです。
多くの場合には、コンピュータのセキュリティを考える際に、 標準作業手続書 (SOP)
がすでに存在します。 わからなければ、セキュリティチームに尋ねてください。

13.2.3. システムおよびユーザアカウント

システムを安全にするにあたり、最も適切な出発点は、 アカウントの監査です。
ルートアカウントのパスワードが強力であること、
シェルアクセスを必要としないアカウントは無効にすることを確実におこなってください。
また、権限を必要とするユーザに対しては、 security/sudo をインストールして、
アクセスが必要となるアプリケーションのみにアクセスを許可するようにしてください。 root
ユーザのパスワードは、決して共有すべきではありません。

アカウントへのアクセスを無効にする方法は二通りあります。
一つ目の方法は、アカウントをロックする方法です。例として、 toor
アカウントをロックする方法を以下に示します。

```
# pw lock toor
```

このコマンドは、アカウントの設定を "toor:*:0:0::0:0:Bourne-again Superuser:/root:" から "toor:*LOCKED**:0:0::0:0:Bourne-again Superuser:/root:" へと変更します。

ときには (おそらく追加のサービスのために)、 この方法が使えない場合があります。
そのような場合には、以下の例のように、 シェルを /sbin/nologin に変更することで、
ログインアクセスを拒否できます。

```
# chsh -s /usr/sbin/nologin toor
```

他のユーザのシェルは、スーパーユーザのみが変更できます。
通常のユーザが行おうとすると失敗します。

アカウント情報は、以下のように最後のエントリが "nologin" シェルとなります。

```
toor:*:0:0::0:0:Bourne-again Superuser:/root:/usr/sbin/nologin
```

/usr/sbin/nologin シェルは、 login(1)
コマンドがこのユーザにシェルを割り当てることをブロックします。

13.2.4. アカウントの権限を拡大する

場合によっては、 システム管理者へのアクセスを他のユーザと共有する必要があります。 FreeBSD
はこのために二つの方法を用意しています。 第一の方法は推奨されませんが、
ルートのパスワードを共有し、ユーザを wheel グループに加える方法です。
これを行うにには、/etc/group を編集し、 最初のグループの最後にユーザを追加してください。
ユーザはカンマ区切りで管理されています。

権限の拡大をする適切な方法は、 security/sudo port を使う方法です。 この port
は、追加の監査、よりきめ細かいユーザ管理、および ユーザを service(8)
のような権限が与えられたコマンのみの実行に制限することもできます。

インストールが終わったら、 visudo インタフェースを使って /usr/local/etc/sudoers
ファイルを編集してください。 以下の例では、新しく webadmin グループが作成され、 trhodes
ユーザがこのグループに追加されます。 その後、ユーザに apache24
を再起動するアクセス権限を与えます。 この手続きは以下のようになります。

```
# pw groupadd webadmin -M trhodes -g 6000
```

```
# visudo
```

```
%webadmin ALL=(ALL) /usr/sbin/service apache24 *
```

ローカルのユーザ管理において、 security/sudo は、 非常に貴重なリソースを提供します。
また、パスワードを不必要にして、デフォルトを ssh(1) 鍵の方法だけにすることもできます。 sshd(8) 経由のパスワードによるログインを無効にし、 sudo へのローカルパスワードのみを使うようにするには、 OpenSSH をご覧ください。

13.2.5. パスワード

パスワードは、テクノロジーにおける必要悪です。 パスワードは極めて複雑であるだけではなく、パスワードを保護する強力なハッシュメカニズムもまた必要となります。
この文書を書いている時点では、 FreeBSD は `crypt()` ライブラリで DES, MD5, Blowfish, SHA256 および SHA512 に対応しています。 デフォルトは SHA512 であり、強度の弱い暗号へは変更すべきではありません。 しかしながら、Blowfish を好むユーザもおります。 DES を除く各メカニズムでは、開始の文字、使用しているハッシュメカニズムを識別可能な特徴を持っています。 MD5 メカニズムでは、シンボルは "$" の符号です。 SHA256 または、 SHA512 では、シンボルは "6"、そして Blowfish は "$2a$" です。 暗号強度の弱いパスワードを使用している場合には、次回のログイン時にユーザが passwd(1) を実行して再ハッシュ化することを促すべきです。

この文書を書いている時点で、Blowfish は AES でなければ、 FIPS (Federal Information Processing Standards) に準拠もしていません。 そのため、使用できない環境があります。

ネットワークに接続しているシステムについては、 二要素認証を使用すべきです。 この認証では、通常あなたが所有する要素と知っている要素が用いられます。 FreeBSD のベースシステムに含まれている OpenSSH および ssh-keys では、ネットワークへのすべてのログインにおける二要素認証の交換で、パスワードを使用すべきではありません。 より詳細な情報については、ハンドブックの OpenSSH 節をご覧ください。 Kerberose のユーザは、ネットワークで OpenSSH を実装するために追加の変更が必要になるでしょう。

13.2.6. バックドアおよびルートキット

バックドアおよびルートキットは、 それらがインストールされた後に脅威となります。 インストールされると、この悪意のあるソフトウェアは、 攻撃者のために侵入口を設置します。 実際的には、システムが一度汚染された後に、調査が行われ、 消去されます。 慎重なセキュリティやシステムエンジニアでさえも、攻撃者が残したソフトウェアを見逃してしまうという恐ろしいリスクが存在しています。

バックドアまたはルートキットソフトウェアは、 管理者にとって役に立つことが一つあります。 それは、一度検出すると、 システムのどこかが危険に冒されていることの痕跡となります。 しかし、通常この種のアプリケーションは、とてもうまく隠れています。 バックドアおよびルートキットを検出するツールが存在しており、 それうちの一つが、 security/rkhunter です。

インストール後、以下のコマンドでシステムをチェックできます。 実行すると多くの情報が出力されます。

```
# rkhunter -c
```

このプロセスを実行中に ENTER キーを何度か押す必要があります。
完了すると、ステータスメッセージが画面に表示されます。
このメッセージは、チェックしたファイルの量、疑わしいファイルの数、
可能性のあるルートキット等の情報を含みます。 チェックの最中、隠されたファイル、 OpenSSH プロトコルの選択、そして、
時には、インストールされているソフトウェアの漸弱性のバージョンに関する一般的なセキュリティの警告が出力されます。 すぐに、もしくはより詳細な解析が行われた後に、対応が可能です。

管理者は皆、 担当しているシステム上で何が実行されているかを把握している必要があります。
rkhunter, lsof や netstat(1) および ps(1) といったネイティブのツールは、
システムに関するかなり多くの情報を与えてくれます。
正常な状態がどのような状態であるかを把握しておき、
本来と違う状況になった場合には、質問をしたり、 疑い深くなってください。
セキュリティが破られることを避けることは理想ですが、 破られたことを把握することは必須です。

13.2.7. バイナリ検証

システムファイルおよびバイナリの検証は、 システム管理者およびセキュリティチームに対して、
システムの変更に関する情報を提供してくれるため重要です。
いかなるシステムにおいても、システム管理チームの知らないところで、
内部のコマンドやアプリケーションは変更すべきではありません。
システムの変更ををモニタリングするソフトウェアアプリケーションは、 侵入検知システム (Intrusion Detection System) または IDS と呼ばれます。

FreeBSD は、基本的な IDS システムをネイティブで提供しています。 実際に、毎晩の periodic(8) セキュリティに関するメールの中では、 管理者に変更点を通知します。
情報はローカルに保存されているので、 悪意のあるユーザが変更し、情報を "欺く" 可能性があります。
そのため、バイナリの署名の別のセットを作成して、 読み取り専用の root 所有のディレクトリ、できれば、 USB ディスクまたは rsync サーバといったシステムとは別のシステムに保存してください。

まず最初に、シードを生成する必要があります。
これは、数値定数で、ハッシュ値の生成やハッシュ値の検証で使われます。 このシードがないと、ファイルのチェックサムの値を偽ったり検証が可能になります。 以下の例では、シードは -s フラグで指定されています。 最初に以下のコマンドを用いて /bin のハッシュ値およびチェックサムを生成してください。

```
# mtree -s 3483151339707503 -c -K cksum,sha256digest -p /bin > bin_chksum_mtree
```

このコマンドの出力は以下のようになります。

```
# mtree: /bin checksum: 3427012225
```

bin_cksum_mtree ファイルを見ると、 以下のような出力となります。

```
#          user: root
#       machine: dreadnaught
#          tree: /bin
#          date: Mon Feb  3 10:19:53 2014
# .
/set type=file uid=0 gid=0 mode=0555 nlink=1 flags=none
.               type=dir mode=0755 nlink=2 size=1024 \
                time=1380277977.000000000
    \133        nlink=2 size=11704 time=1380277977.000000000 \
                cksum=484492447 \

sha256digest=6207490fbdb5ed1904441fbfa941279055c3e24d3a4049aeb45094596400662a
    cat         size=12096 time=1380277975.000000000 cksum=3909216944 \

sha256digest=65ea347b9418760b247ab10244f47a7ca2a569c9836d77f074e7a306900c1e69
    chflags     size=8168 time=1380277975.000000000 cksum=3949425175 \

sha256digest=c99eb6fc1c92cac335c08be004a0a5b4c24a0c0ef3712017b12c89a978b2dac3
    chio        size=18520 time=1380277975.000000000 cksum=2208263309 \

sha256digest=ddf7c8cb92a58750a675328345560d8cc7fe14fb3ccd3690c34954cbe69fc964
    chmod       size=8640 time=1380277975.000000000 cksum=2214429708 \

sha256digest=a435972263bf814ad8df082c0752aa2a7bdd8b74ff01431ccbd52ed1e490bbe7
```

コンピュータのホスト名、現在の日付と時間、mtree(8)
を実行したユーザの情報すべてがこのレポートには含まれています。
また、各バイナリに対するチェックサム、サイズ、タイムスタンプおよび SHA256
ダイジェストも含まれています。

バイナリ署名の検証のために、 以下のコマンドを実行すると、現在の署名のリストを読み込み、
結果を出力します。

```
# mtree -s 3483151339707503 -p /bin < bin_chksum_mtree >> bin_chksum_output
```

このコマンドを実行すると、すでにチェックサムを生成している /bin
に対して、同様のチェックサムを生成します。 このコマンドを実行してから変更が行われていないので、
bin_chksum_output への主力は空となります。 変更が行われた場合をシミュレートするために、
/bin/cat ファイルの日付を touch(1) を使って変更して、 再度検証のコマンドを実行してみます。

```
# touch /bin/cat
```

```
# mtree -s 3483151339707503 -p /bin < bin_chksum_mtree >> bin_chksum_output
```

```
# cat bin_chksum_output
```

```
cat changed
    modification time expected Fri Sep 27 06:32:55 2013 found Mon Feb  3 10:28:43 2014
```

security/aide のような、より高度な IDS システムもありますが、ほとんどのケースにおいて、mtree(8) は管理者が必要とする機能を提供します。悪意のあるユーザが、シード値およびチェックサムの出力を見れないようにすることが重要です。

13.2.8. セキュリティのためのシステムの調整

システムの機能の多くは、sysctl(8) を使って調整できます。Denial of Service (DOS) スタイルの攻撃を避けるためのセキュリティ機能に対しても同様です。
この節では、より重要な調整についても触れています。sysctl(8) により、設定が変更された時はいつでも、望まない危害が起こる可能性は高まり、システムの可用性に影響します。システム全体の設定を変更する時には、システムの CIA を考える必要があります。

以下では、sysctl(8) の一覧、および変更がシステムにどのように影響するかを説明します。

デフォルトでは、FreeBSD のカーネルはセキュリティレベル -1 で起動します。このセキュリティレベルは、変更不可のファイルフラグを外したり、すべてのデバイスに対して読み込みおよび書き込みができたりするので、"insecure mode" と呼ばれます。このセキュアレベルは、管理者または init(8) による起動時のスクリプトにより変更されない限り -1 のままです。/etc/rc.conf において、`kern_securelevel_enable` を `YES` とし、`kern_securelevel` に必要とする値を設定することで、システム起動時にセキュアレベルを高めることができます。
これらの設定についてのより詳細な情報については、security(7) および init(8) をご覧ください。

`securelevel` を大きくしすぎると、Xorg が動かなくなったり、他の問題が起きる可能性があります。
デバッグの心づもりをしてください。

つぎに変更を検討すべき sysctl(8) は、net.inet.tcp.blackhole および net.inet.udp.blackhole です。これらを設定すると、閉じたポートに対して届く SYN パケットはドロップされ、RST レスポンスを返しません。通常は、RST を返し、そのポートが閉じられていることを伝えます。これにより、システムに対する "ステルス" スキャンに対し、ある程度の防御となります。net.inet.tcp.blackhole を "2"、net.inet.udp.blackhole を "1" に設定してください。詳細な情報について blackhole(4) をご覧ください。

さらに、net.inet.icmp.drop_redirect および net.inet.ip.redirect も設定すべきです。これら 2 つの sysctl(8) は、リダイレクト攻撃を防ぐ助けとなるでしょう。リダイレクト攻撃は、故意に通常のネットワークでは必要としないような大量の ICMP タイプ 5 のパケットを発生します。そのため net.inet.icmp.drop_redirect を "1"、net.inet.ip.redirect を "0" に設定して下さい。

ソースルーティングは、
内部ネットワーク上でルーティングできないアドレスを検出したりアクセスするための方法です。
通常ルーティングできないアドレスは、意図してルーティングできないようにしているので、この設定はおそらく無効にすべきです。この機能を無効にするには、net.inet.ip.sourceroute および net.inet.ip.accept_sourceroute を "0" に設定してください。

ブロードキャストアドレスに対するすべての ICMP エコーリクエストは、ドロップしてください。
ネットワーク上のコンピュータがサブネットにあるすべてのホストにメッセージを送る必要がある場合には、 メッセージはブロードキャストアドレスに送られます。 外部のホストについては、このような送信をする必要はないので、
外部からブロードキャストへのリクエストをすべて拒否するように、 net.inet.icmp.bmcastecho を "0" に設定してください。

まだ多くの sysctl(8) が security(7) で説明されています。
さらに多くの情報を調べることが推奨されます。

13.3. ワンタイムパスワード

デフォルトで、FreeBSD は One-time Passwords In Everything (OPIE) に対応しています。 OPIE はデフォルトでは MD5 ハッシュを使用します。

三種類の異なる「パスワード」があります。 まず一つ目は、通常の UNIX® スタイル、もしくは Kerberos のパスワードです。 二つ目は、opiekey(1) によって生成され、 opiepasswd(1) およびログインプロンプトが受け付けるワンタイムパスワードです。 三つ目のパスワードは、opiekey(1) と場合により `opiepasswd` に対してワンタイムパスワードを生成するのに使われる "秘密のパスワード" です。

秘密のパスワードは、UNIX® パスワードと何の関連性もありません。
両者を同一に設定することは可能ですが、お奨めしません。古い UNIX® パスワードは長さが 8 文字に制限されていました 。 これに対し、OPIE の秘密のパスワードには 8 文字の制限はありません。 6 語から 7 語からなるパスフレーズがふつうです。ほとんどの部分で、 OPIE システムは UNIX® のパスワードシステムと完全に独立して動作するようになっています。

パスフレーズに加え、OPIE システムにとって重要な 2 種類のデータがあります。一つは "シード (seed: 種)" または "キー (key: 鍵)" と呼ばれるもので、2 つの文字と 5 つの数字で構成されます。もう一つは "シーケンス番号 (iteration count)" で、1 から 100 までの整数です。 OPIE はここまでに述べたデータを利用してワンタイムパスワードを生成します。
その方法は、まずシードと秘密のパスフレーズを連結し、 それに対してシーケンス番号の回数だけ MD5 ハッシュを繰り返し計算します。 そしてその結果を 6 つの短い英単語に変換します。 この 6 つの英単語がワンタイムパスワードです。 認証システム (主は PAM) は、前回最後に受け付けたワンタイムパスワードを記録しています。
そして、その前回のワンタイムパスワードと、 ユーザが入力したワンタイムパスワードを 1 回ハッシュ関数にかけた結果とが一致した場合に、 このユーザは認証されます。
一方向ハッシュ関数を使っているので、
もし正しく認証されたワンタイムパスワードが一回盗聴されたとしても、
次回以降に使われる複数のワンタイムパスワードを生成することは不可能です。
シーケンス番号はログインが成功するたびに一つずつ減らされて、
ユーザとログインプログラムの間で同期が取られます。 シーケンス番号が 1 まで減ったら、 OPIE を再度初期化する必要があります。

このプロセスに関連するいくつかのプログラムがあります。 opiekey(1) は、シーケンス番号と、シードと、 秘密のパスフレーズを受け付けて、ワンタイムパスワード 1 つ、または一連のワンタイムパスワードの一覧を生成します。 opiepasswd(1) は、OPIE の初期化に加え、パスワード、 シーケンス番号やシードを変更するためにも使用されます。
このプログラムを実行するには、秘密のパスフレーズか、
または、シーケンス番号とシードとワンタイムパスワードの 1 組かの、どちらかを与えます。

opieinfo(1) は、認証ファイル (/etc/opiekeys) を調べて、プログラムを起動したユーザの現在のシーケンス番号とシードを表示します。

4 種類の異なる操作があります。1 つ目は、opiepasswd(1) を信頼できる通信路上で利用して、最初にワンタイムパスワードを設定したり、秘密のパスフレーズやシードを変更する操作です。2 つ目は、同じことを行うために opiepasswd(1) を信頼できない通信路上で利用する操作です。この場合は信頼できる通信路経由の opiekey(1) を併用します。3 つ目は、opiekey(1) を使い、信頼できない通信路を通じてログインする操作です。4 番目は、opiekey(1) を使って複数のワンタイムパスワードを一気に生成する操作です。
ここで生成した複数のワンタイムパスワードは、メモしたり印刷したりして携帯し、信頼できる通信路が一切ないところからの接続に利用できます。(訳注: ワンタイムパスワードを記録した紙をなくさないこと! 電話番号や IP アドレス、ユーザ名を一緒にメモしていたら最悪です!!)

13.3.1. 信頼できる通信路での初期化

OPIE を初めて初期化するには、 opiepasswd(1) を実行してください。

```
% opiepasswd -c
[grimreaper] ~ $ opiepasswd -f -c
Adding unfurl:
Only use this method from the console; NEVER from remote. If you are using
telnet, xterm, or a dial-in, type ^C now or exit with no password.
Then run opiepasswd without the -c parameter.
Using MD5 to compute responses.
Enter new secret pass phrase:
Again new secret pass phrase:

ID unfurl OTP key is 499 to4268
MOS MALL GOAT ARM AVID COED
```

`Enter new secret pass phrase:` または `Enter secret password:` というプロンプトに対して、パスワードまたはパスフレーズを入力してください。このパスワードは、ログインするときに使うワンタイムパスワードを生成するために使うものであり、ログインのためのパスワードではありません。"ID" から始まる行は、1 回分のパラメータで、ログイン名とシーケンス番号とシードです。ログインするときには、システム側がこれらのパラメータを覚えていて表示してくれるので、これらのパラメータを覚えておく必要はありません。
最後の行が、今述べたパラメータと入力された秘密のパスワードから計算されたワンタイムパスワードです。 次にログインするときに打ち込むべきワンタイムパスワードがこれです。

13.3.2. 信頼できない通信路での初期化

信頼できない通信路を使って秘密のパスフレーズを初期化または変更するためには、opiekey(1) を実行するための信頼できる通信路を用意しておく必要があります。たとえばそれは、信頼できるマシンのシェルプロンプトだったりするでしょう。(訳注: ここでの通信路とはマシンそのものになります。信頼できるマシンとは、信頼できる人がしっかり管理しているマシンということです)。

他に準備しておくものとして、シーケンス番号 (100 は適切な値といえるでしょう)
と、場合によっては自分で考えた、 またはランダムに生成されたシードがあります。
信頼できない通信路を使うときには、opiepasswd(1) を使ってコンピュータを初期化してください。

```
% opiepasswd

Updating unfurl:
You need the response from an OTP generator.
Old secret pass phrase:
    otp-md5 498 to4268 ext
    Response: GAME GAG WELT OUT DOWN CHAT
New secret pass phrase:
    otp-md5 499 to4269
    Response: LINE PAP MILK NELL BUOY TROY

ID mark OTP key is 499 gr4269
LINE PAP MILK NELL BUOY TROY
```

デフォルトのシードで構わなければ、`Return` を押してください。アクセスパスワードを入れる前に、
あらかじめ用意しておいた信頼できる通信路へ移って、 先ほどと同じパラメータを入力します。

```
% opiekey 498 to4268
Using the MD5 algorithm to compute response.
Reminder: Do not use opiekey from telnet or dial-in sessions.
Enter secret pass phrase:
GAME GAG WELT OUT DOWN CHAT
```

信頼できない通信路の方に戻って、
生成されたワンタイムパスワードをコピーして対応するプログラムに入力します。

13.3.3. ワンタイムパスワードを一つ生成する

OPIE を初期化したら、 ログイン時には以下のようなプロンプトが出てくるでしょう。

```
% telnet example.com
Trying 10.0.0.1...
Connected to example.com
Escape character is '^]'.

FreeBSD/i386 (example.com) (ttypa)

login: <ユーザ名>
otp-md5 498 gr4269 ext
Password:
```

OPIE のプロンプトには便利な機能が備わっています。 パスワードプロンプトに対して、 `Return`
を押すとエコーモードに切り替わり、 タイプした文字がそのまま見えるようになるのです。 これは、

紙に印刷していたりするワンタイムパスワードを手で入力しなければならない場合に役立つ機能です。

次に、 このログインプロンプトに対して入力するワンタイムパスワードを生成してください。
これは、opiekey(1) プログラムを使える信頼できるマシン上で行わなければなりません。
このプログラムには Windows®, Mac OS® および FreeBSD 版があります。 どちらも、
コマンドラインからシーケンス番号とシードを指定しなければなりません。
ログインしようとしているマシンのログインプロンプトから直接カットアンドペーストすると楽でしょう
。

信頼できるシステムで

```
% opiekey 498 to4268
Using the MD5 algorithm to compute response.
Reminder: Do not use opiekey from telnet or dial-in sessions.
Enter secret pass phrase:
GAME GAG WELT OUT DOWN CHAT
```

ワンタイムパスワードが生成されたので、 ログインを続けてください。

13.3.4. 複数のワンタイムパスワードを生成する

都合によっては、 信頼できるマシンや信頼できる通信路が一切確保できないようなことがあるでしょう。
このような場合には、opiekey(1) を使って複数のワンタイムパスワードを生成できます。 たとえば

```
% opiekey -n 5 30 zz99999
Using the MD5 algorithm to compute response.
Reminder: Do not use opiekey from telnet or dial-in sessions.
Enter secret pass phrase: <secret password>
26: JOAN BORE FOSS DES NAY QUIT
27: LATE BIAS SLAY FOLK MUCH TRIG
28: SALT TIN ANTI LOON NEAL USE
29: RIO ODIN GO BYE FURY TIC
30: GREW JIVE SAN GIRD BOIL PHI
```

-n 5 という引数によって 5 個のワンタイムパスワードを順に生成します。 また 30 は、
最後のシーケンス番号となるべき数字です。出力は使う順番とは 逆
に出力されていることに注意してください (訳注:
一番最初に使うワンタイムパスワードは一番最後に出力されたものです)。
もしあなたがセキュリティに偏執するなら、
この結果を紙と鉛筆を使って手で書き移した方がよいかもしれません。
そうでなければ、この結果を印刷すると良いでしょう。 ここで、
出力の各行はシーケンス番号とそれに対応する一回分のワンタイムパスワードです。
消費済みのワンタイムパスワードをペンで消していってください。

13.3.5. UNIX® パスワードの利用を制限する

OPIE は、ログインセッションの IP アドレスをベースとした UNIX® パスワードの使用を制限できます。
関連ファイルは、/etc/opieaccess で、 デフォルトで用意されています。 このファイルの詳細や、
このファイルを使用する際に考慮すべきセキュリィについては opieaccess(5) を確認してください。

以下は opieaccess の例です。

```
permit 192.168.0.0 255.255.0.0
```

この行では、(なりすましされやすい) IP ソースアドレスが、ある値やマスクにマッチするユーザに対して、 UNIX® パスワードをいつでも許可します。

もし opieaccess のどのルールにも一致しなければ、 デフォルトでは非 OPIE ログインは使えません。

13.4. TCP Wrappers

TCP Wrappers は、 すべてのサーバデーモンに対するサポートをその管理下で提供できるように、「inetd 「スーパサーバ」」 の機能を拡張します。 この方法を使うことで、ログへの対応、接続に対してメッセージを返したり、
内部の接続だけを許可するようにデーモンを設定することが可能となります。
これらの機能のいくつかはファイアウォールでも実装できますが、 TCP Wrappers は、システムを守るためのレイヤを追加し、 ファイアウォールが提供する以上の管理機能を提供します。

TCP Wrappers は、 適切に設定されたファイアウォールの置き換えと考えるべきではありません。 TCP Wrappers は、 ファイアウォールや他のセキュリティ強化のツールと組み合わせて使うべきです。

13.4.1. 初期設定

FreeBSD 上で TCP Wrappers を有効にするには、 rc.conf から `-Ww` オプションで inetd(8) サーバが起動されることを確認してください。 その後、/etc/hosts.allow を適切に設定してください。

他の TCP Wrappers の実装と異なり、 hosts.deny は廃止されました。すべての設定オプションは /etc/hosts.allow に書かれている必要があります。

最も簡単な設定におけるデーモンの接続ポリシは、 /etc/hosts.allow の中で、オプションごとに許可またはブロックするように設定するというものです。 FreeBSD のデフォルトの設定では、inetd(8) から起動されたすべてのデーモンの接続を許可します。

基本的な設定は、通常 `daemon : address : action` という形式です。ここで、 `daemon` は、 inetd(8) が起動するデーモンの名前です。 `address` の部分は、有効なホスト名、 IP アドレスまたは、 括弧 ([ ]) で囲まれた IPv6 アドレスです。 `action` は、 `allow` または `deny` です。 TCP Wrappers は、最初にマッチしたルールが適用されます。
これは、設定ファイルに対するルールにマッチするかどうかのスキャンは、
昇順に行われることを意味しています。 マッチすると、ルールが適用され、検索のプロセスは終了します。

例として、POP3 の接続を mail/qpopper デーモン経由で許可するには、以下の行を hosts.allow に追加してください。

```
# This line is required for POP3 connections:
qpopper : ALL : allow
```

この行を追加したら、 inetd(8) を再起動してください。

```
# service inetd restart
```

13.4.2. 高度な設定

TCP Wrappers は、 接続を取り扱う以上の制御を行う高度な設定も提供しています。 ある時は、接続しているホストまたはデーモンにコメントを返すことが適切であることがあります。
別の場合では、おそらくログエントリを記録したり、
管理者にメールで送る必要があることもあるでしょう。 またその他の状況としては、サービスをローカルの接続のみの使用に制限する必要がある場合もあります。
これらはすべて、ワイルドカード と呼ばれる設定のオプション (拡張文字および外部コマンドの実行)で可能となります。

外部コマンド

接続は拒否しなければならないが、
その理由を接続の確立を試みた相手に送りたい状況を考えてください。 このアクションは、twist を使うことで実現可能です。 接続が試みられると、twist はシェルコマンドまたはスクリプトを実行します。 この場合の例は、 hosts.allow に書かれています。

```
# The rest of the daemons are protected.
ALL : ALL \
    : severity auth.info \
    : twist /bin/echo "You are not welcome to use %d from %h."
```

この例では、 "You are not allowed to use daemon from hostname." というメッセージを、アクセスファイルの中で設定されていないすべてのデーモンに対して返します。 接続元に対し、確立された接続が破棄された直後に返答することは有効です。 返信に使われるメッセージは、引用符 (") で囲む 必要 があります。

攻撃者や攻撃者のグループは、
これらのデーモンの接続のリクエストであふれさせることにより、 サーバに対して DoS 攻撃を仕掛けることができます。

他の可能性は spawn を使うことです。 twist と同様に、 spawn は、暗黙のうちに接続を拒否し、外部のシェルコマンドやスクリプトを実行できます。 twist と異なり、spawn は、接続を確立した相手に対し、返事を返すことはありません。
たとえば、以下のような設定の行を考えてみてください。

```
# We do not allow connections from example.com:
ALL : .example.com \
    : spawn (/bin/echo %a from %h attempted to access %d >> \
      /var/log/connections.log) \
    : deny
```

この行は、*.example.com からの接続をすべて拒否します。 ホスト名、IP アドレスおよびアクセスを試みたデーモンが、 /var/log/connections.log に記録されます。

この例では、置換文字 %a および %h が使われています。 置換文字の完全な一覧は hosts_access(5) をご覧ください。

ワイルドカードオプション

ALL オプションは、 デーモン、ドメインまたは IP アドレスのすべてのインスタンスのどれかにマッチするかどうかに使われます。
他のワイルドカードは、偽造された IP アドレスを提供するホストにマッチするかどうかに用いられる PARANOID です。 たとえば、PARANOID を使うことで、 ホスト名と異なる IP アドレスからの接続があった時のアクションを定義できます。 以下の例では、ホスト名から検索される IP アドレスと異なる IP アドレスを持つ sendmail(8) への接続のすべてのリクエストを拒否します。

```
# Block possibly spoofed requests to sendmail:
sendmail : PARANOID : deny
```

クライアントもしくはサーバの DNS の設定が間違っている場合に、 PARANOID ワイルドカードを使うと、 サーバがとても使いづらくなります。
管理者の慎重さが求められます。

ワイルドカードおよび関連する機能についてもっと知りたい場合には、 hosts_access(5) をご覧ください。

上記の設定が動作するには、hosts.allow の中で、最初の設定の行がコメントアウトされている必要があります。

13.5. Kerberos5

Kerberos は、 サーバのサービスによってユーザが安全に認証を受けられるようにするための、ネットワークの付加システムおよびプロトコルです。 Kerberos は、 身元確認プロキシシステムや、信頼される第 3 者認証システムとも説明されます。 ユーザが Kerberos を使って認証を行った後は、通信は暗号化され、 プライバシおよびデータの完全性を保証することができます。

Kerberos の唯一の機能は、 ネットワーク上のユーザの安全な認証を提供することです。 承認 (どのユーザが許可されているか) や監査 (ユーザがどのような作業を行っているか) の機能は提供しません。 Kerberos を使う際は、承認および監査サービスを提供する他のセキュリティの手段との利用が、 推奨されます。

この節では、FreeBSD 用として配布されている Kerberos をセットアップする際のガイドを提供します。完全な説明が必要な場合には、 マニュアルページを参照してください。

この節における Kerberos のインストールのデモでは、以下のような名前空間が使われます。

- DNS ドメイン ("ゾーン") は、 example.org です。
- Kerberos の領域は、 EXAMPLE.ORG です。

Kerberos の設定では、 内部での使用でも実際のドメイン名を使ってください。 DNS

の問題を避けることができ、 他の Kerberos のレルム (realm) との相互運用を保証します。

13.5.1. 歴史

Kerberos は、 ネットワークのセキュリティ問題を解決するために、 MIT で開発されました。 Kerberos プロトコルは、 必ずしも安全ではないインターネット接続においても、 サーバに対して (逆もまた同様に)、 強い暗号を使って身元を証明します。

Kerberos は、 ネットワーク認証プロトコルの名前であり、 Kerberos telnet のように、 このプログラムを実装しているプログラムを表すための形容詞でもあります。
プロトコルの現在のバージョンはバージョン 5 で、 RFC 1510 として文書化されています。

このプロトコルのいくつものフリーの実装が、 さまざまなオペレーティングシステムで利用できます。 最初の Kerberos を開発したマサチューセッツ工科大学 (MIT) は、 開発した Kerberos パッケージを継続的に保守しています。 アメリカ合衆国では暗号製品として良く使われていますが、 歴史的には、 アメリカ合衆国 の輸出規制により制限されてきました。 MIT で実装された Kerberos は、 security/krb5 package または port から利用できます。 バージョン 5 のもう一つの実装が、 Heimdal Kerberos です。 この実装は、アメリカ合衆国の外で開発されたため、 輸出の制限を避けることができます。 Heimdal Kerberos は security/heimdal> package または port からインストールできますが、最小構成は FreeBSD の base インストールに含まれています。

以下の説明では FreeBSD に含まれている Heimdal ディストリビューションの使用を想定しています。

13.5.2. Heimdal KDC の設定

鍵配布センター (KDC) は、 Kerberos が提供する中心的な認証サービスで、 Kerberos チケットを発行するコンピュータです。 KDC は、 Kerberos のレルムの中のすべてのコンピュータから "信頼"されています。 そのため、厳重なセキュリティに対する配慮が必要となります。

Kerberos サーバの実行にコンピュータのリソースはほとんど必要ありませんが、 セキュリティの観点から、KDC としてのみ機能する専用のコンピュータが推奨されます。

KDC を設定するにあたって、 KDC として動作するために、 適切に /etc/rc.conf が設定されていることを確認してください。 必要に応じて、 システムの設定を反映するようにパスを調整する必要があります。

```
kerberos5_server_enable="YES"
kadmind5_server_enable="YES"
```

次に、/etc/krb5.conf を以下のように編集してください。

```
[libdefaults]
    default_realm = EXAMPLE.ORG
[realms]
    EXAMPLE.ORG = {
        kdc = kerberos.example.org
        admin_server = kerberos.example.org
    }
```

```
[domain_realm]
    .example.org = EXAMPLE.ORG
```

/etc/krb5.conf の中で、KDC は、完全修飾されたホスト名 kerberos.example.org を使うことが想定されています。KDC が異なるホスト名を持つ場合には、名前の解決が行われるように、適切に CNAME (エイリアス) エントリをゾーンファイルに追加してください。

適切に DNS サーバが設定されている大きなネットワークでは、上記の例は、以下のように整理されます。

```
[libdefaults]
      default_realm = EXAMPLE.ORG
```

そして、example.org ゾーンファイルには、以下の行が付け加えられます。

```
_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
```

クライアントが、Kerberos サービスを見つけるためには、/etc/krb5.conf を完全に設定するか、/etc/krb5.conf を最低限に設定し、さらに DNS サーバを適切に設定する 必要 があります。

次に Kerberos データベースを作成してください。このデータベースには、マスター鍵により暗号化されたすべてのプリンシパルの鍵が含まれています。このパスワードは、/var/heimdal/m-key に保存されるため、覚える必要はありません。マスター鍵を作成するには、kstash(8) を実行して、パスワードを入力してください。

マスター鍵を作成したら、kadmin -l を使ってデータベースを初期化してください。このオプションを使うと、kadmin(8) は、kadmind(8) ネットワークサービスを使わず、ローカルのデータベースファイルを直接変更します。これにより、データベースを作成する前に、データベースへの接続を試みてしまうという、卵が先か鶏が先かという問題を回避できます。kadmin(8) プロンプトで、init を使って、レルムに関する初期のデータベースを作成してください。

最後に、kadmin(8) プロンプトで add を使って最初のプリンシパルを作成して下さい。差し当たりは、プリンシパルに対するデフォルトのオプションに従ってください。後で modify を使うことで、変更することができます。kadmin(8) プロンプトで ? と入力すると、利用可能なオプションを確認できます。

データベース作成のセッションの例は以下のようになります。

```
# kstash
Master key: xxxxxxxx
```

```
Verifying password - Master key: xxxxxxxx

# kadmin -l
kadmin> init EXAMPLE.ORG
Realm max ticket life [unlimited]:
kadmin> add tillman
Max ticket life [unlimited]:
Max renewable life [unlimited]:
Attributes []:
Password: xxxxxxxx
Verifying password - Password: xxxxxxxx
```

次に KDC サービスを起動してください。 `service kerberos start` および `service kadmind start` を実行してサービスを起動してください。 この時点で、kerberos 化されたデーモンが走っていなくても、 KDC のコマンドラインから、作成したばかりの (ユーザ) プリンシパルのチケットを入手したり、 一覧を表示することができることを確認できます。

```
% kinit tillman
tillman@EXAMPLE.ORG's Password:

% klist
Credentials cache: FILE:/tmp/krb5cc_500
    Principal: tillman@EXAMPLE.ORG

  Issued           Expires          Principal
Aug 27 15:37:58  Aug 28 01:37:58  krbtgt/EXAMPLE.ORG@EXAMPLE.ORG
```

必要がなくなった時には、チケットを破棄できます。

```
% kdestroy
```

13.5.3. Heimdal Kerberos サービスを有効にする。

最初に /etc/krb5.conf を KDC からクライアントコンピュータへ、 scp(1) または物理的にリムーバブルディスクを使うといった安全な方法でコピーしてください。

次に /etc/krb5.keytab を作成してください。 これが Kerberos 化されたデーモンを提供するサーバとワークステーションの間での大きな違いです: サーバには keytab が置かれている必要があります。 このファイルには、サーバのホスト鍵が含まれています。 この鍵により、ホストおよび KDC が他の身元の検証ができます。 鍵が公開されてしまうと、サーバのセキュリティが破られてしまうため、
このファイルは安全にサーバに転送しなければなりません。

一般的には、kadmin(8) を使って、 keytab をサーバに転送します。 ホストプリンシパル (KDC 側の krb5.keytab) も kadmin(8) を使って作成するので便利です。

すでにチケットを入手し、そのチケットは、 kadmin(8) インタフェースで使用できることが kadmind.acl で許可されている必要があります。 アクセスコントロールリストの設計の詳細については、

info heimdal の "Remote administration" というタイトルの章をご覧ください。 リモートからの kadmin アクセスを有効にする代わりに、 管理者は、ローカルコンソールまたは ssh(1) を用いて安全に KDC に接続し、 kadmin -l を使用して、 ローカルで管理作業を行うことができます。

/etc/krb5.conf をインストールしたら、 Kerberos サーバから add --random-key を使ってください。 このコマンドは、サーバのホストプリンシパルを追加します。 そして、ext を用いて、 サーバのホストプリンシパルを keytab に抽出してください。 以下は、使用例です。

```
# kadmin

kadmin> add --random-key host/myserver.example.org
Max ticket life [unlimited]:
Max renewable life [unlimited]:
Attributes []:
kadmin> ext host/myserver.example.org
kadmin> exit
```

ext は、デフォルトでは、抽出された鍵を /etc/krb5.keytab に保存します。

KDC 上で kadmind(8) を走らせていない場合で、 リモートから kadmin(8) に接続出来ない場合には、 ホストプリンシパル (host/myserver.EXAMPLE.ORG) を直接 KDC 上で追加し、 その後、以下のように KDC 上の /etc/krb5.keytab の上書きを避けるため、 一時ファイルに抽出してください。

```
# kadmin
kadmin> ext --keytab=/tmp/example.keytab host/myserver.example.org
kadmin> exit
```

その後、scp(1) またはリムーバブルディスクを使って、 keytab を安全にサーバコンピュータにコピーしてください。 KDC 上の keytab を上書きすることを避けるため、 デフォルトとは異なる名前を指定してください。

これでサーバは、 krb5.conf を使って KDC と通信ができるようになりました。 そして、krb5.keytab によって身元を証明できるようになったので、 Kerberos サービスを有効にする準備が出来ました。 この例では、 telnetd(8) サービスが /etc/inetd.conf で有効に設定され、 service inetd restart によって、 inetd(8) サービスを再起動します。

```
telnet    stream  tcp     nowait  root    /usr/libexec/telnetd  telnetd -a user
```

重要な変更箇所は、-a 認証がユーザに設定されていることです。 詳細については、 telnetd(8) を参照してください。

13.5.4. Heimdal Kerberos クライアントを有効にする

クライアントコンピュータの設定は簡単です。 /etc/krb5.conf のみが必要です。 このファイルをセキュリティ的に安全な方法で、KDC からクライアントコンピュータへコピーしてください。

クライアントから、kinit(1), klist(1) および kdestroy(1) を使用し、上記で作成したプリンシパルに対するチケットの入手、表示、削除を行い、クライアントコンピュータを試験してください。 Kerberos アプリケーションを使って Kerberos が有効なサーバに接続することもできるはずです。もしうまく機能しない場合でも、チケットを入手できるのであれば、問題はおそらくサーバにあり、クライアントまたは KDC の問題ではないと考えられます。

Kerberos 化されたアプリケーションを試験する際には、tcpdump(1) といったパケットスニファを使用して、パスワードが平文で送られていないことを確認してください。

コア以外の さまざまな Kerberos クライアントアプリケーションが利用可能です。 FreeBSD の "最小" インストールでは、インストールされる Kerberos 化された唯一のサービスは、telnetd(8) です。

Heimdal port は、Kerberos 化されている ftpd(8), rshd(8), rcp(1), rlogind(8) および他のあまり一般的ではないプログラムをインストールします。 MIT port も、すべての Kerberos クライアントアプリケーションをインストールします。

13.5.5. ユーザ設定ファイル: .k5login および .k5users

レルムのユーザは、一般的には、ローカルユーザアカウントに対応する Kerberos プリンシパルを持ちます。しかしながら、時々 Kerberos プリンシパルに対応しないローカルユーザアカウントへのアクセスが必要となることがあります。たとえば、`tillman@EXAMPLE.ORG` が、ローカルユーザアカウント `webdevelopers` へのアクセスが必要となることがあります。そして、他のプリンシパルが同じローカルアカウントにアクセスが必要になることもあります。

ユーザのホームディレクトリに置かれた .k5login および .k5users ファイルを使うことで、この問題を解決出来ます。たとえば、以下の行を含む .k5login を `webdevelopers` のホームディレクトリに置くと、一覧にある両方のプリンシパルは、共有のパスワードを必要としなくても、このアカウントにアクセス出来ます。

```
tillman@example.org
jdoe@example.org
```

.k5users の詳細については、ksu(1) を参照してください。

13.5.6. Kerberos Tips, Tricks, およびトラブルシューティング

- Heimdal または MITKerberos ports のどちらを使う場合でも、`PATH` は、Kerberos 版のクライアントアプリケーションが、システムにあるアプリケーションより先に見つかるように設定されていることを確認してください。
- レルムにあるすべてのコンピュータの間で時刻が同期していないと、認証に失敗してしまいます。NTP を用いた、時刻の同期方法については、「NTP」をご覧ください。
- MIT および Heimdal 間の運用は、標準化されていない kadmin(8) を除けばうまく機能します。
- ホスト名が変更された場合は、`host/` プリンシパルを変更し、keytab をアップデートする必要があります。 Apache の www/mod_auth_kerb で使われる `www/` プリンシパルのような特別な keytab エントリでも必要となります。

- レルムの中のすべてのホストは、DNS、もしくは、最低限/etc/hostsにおいて正引きおよび逆引き両方で名前解決できる必要があります。 CNAME は動作しますが、A および PTR レコードは、正しく適切な位置に記述されている必要があります。名前が解決できない場合のエラーメッセージは、次の例のように、直感的に原因が分かるようなものではありません。 `Kerberos5 refuses authentication because Read req failed: Key table entry not found`.
- KDC に対しクライアントとして振る舞うオペレーティングシステムの中には、 ksu(1) に対して、 `root` 権限に setuid を許可しないものがあります。 この設定では、 ksu(1) は動作しないことを意味します。 これは KDC のエラーではありません。
- MITKerberos において、プリンシパルが、デフォルトの 10 時間を超えるチケットの有効期限としたい場合には、 kadmin(8) のプロンプトで `modify_principal` を使って、対象のプリンシパルおよび `krbtgt` プリンシパル両方の有効期限の最大値を変更してください。 プリンシパルは、 `kinit -l` を使用して、長い有効期限のチケットを要求できます。 *

トラブルシューティングのために、KDC でパケットスニファを走らせ、一方で、ワークステーションにおいて kinit(1) を実行すると、 kinit(1) を実行するやいなや、パスワードを入力し終わる前でも、 Ticket Granting Ticket (TGT) が送られてきます。これに関する説明は、以下の通りです。 Kerberos サーバは、いかなる未承認のリクエストに対して、 自由に TGT を送信します。 しかしながら、すべての TGT は、ユーザのパスワードから生成された鍵により、暗号化されています。そのため、ユーザがパスワードを入力した時には、 パスワードは KDC には送られません。その代わりこのパスワードは、kinit(1) がすでに入手した TGT の復号化に使われます。もし、復号化の結果、 有効なチケットで有効なタイムスタンプの場合には、 ユーザは、有効な Kerberos クレデンシャルを持ちます。 このクレデンシャルには、 Kerberos サーバ自身の鍵により暗号化された実際の TGT とともに、将来 Kerberos サーバと安全な通信を確立するためのセッション鍵が含まれています。 この暗号の 2 番目のレイヤは、 Kerberos サーバが、 各 TGT の真偽の検証を可能にしている部分です。

- たとえば一週間といった長い有効期限のチケットを使いたい場合で、 OpenSSH を使って、チケットが保存されているコンピュータに接続しようとする場合は、 Kerberos `TicketCleanup` が sshd_config において `no` と設定されているか、チケットが、ログアウト時に削除されることを確認してください。
- ホストプリンシパルは長い有効期限のチケットを持つことができます。 もし、ユーザプリンシパルが 1 週間の有効期限を持ち、 接続しているホストが、9 時間の有効期限を持っている場合には、ユーザキャッシュは有効期限が切れたホストプリンシパルを持つことになり、 想定したように、チケットキャッシュが振る舞わないことが起こりえます。
- kadmind(8) で説明されているような、 特定の問題のあるパスワードが使われることを避けるために krb5.dict を設定する時には、パスワードポリシが割り当てられたプリンシパルにのみ適用されることを覚えていてください。 krb5.dict で使われている形式では、 一行に一つの文字列が置かれています。 /usr/share/dict/words にシンボリックリンクを作成することは、有効です。

13.5.7. MIT port との違いについて

MIT と Heimdal 版の大きな違いは、 kadmin(8) に関連しています。 このプログラムは、異なる (ただし等価な) コマンド群を持ち、そして、 異なるプロトコルを使用します。 もし KDC に MIT を使用している場合には、 Heimdal 版の kadmin(8) を使って KDC をリモートから (逆も同様に) 管理できないことを意味しています。

クライアントアプリケーションでは、同じタスクを行う際に、
若干異なるコマンドラインのオプションが使われることもあります。 MIT Kerberos ウェブサイト に書かれているガイドに従うことが推奨されます。 path の問題について注意してください。 MIT port はデフォルトで /usr/local/ にインストールします。 そのため、もし PATH においてシステムのディレクトが最初に書かれている場合には、 MIT 版ではなく、"通常の" システムアプリケーションが起動してしまいます。

FreeBSD の MITsecurity/krb5 port において、 telnetd(8) および klogind 経由でのログインが奇妙な振る舞いをすることを理解するには、 port からインストールされる /usr/local/share/doc/krb5/README.FreeBSD を読んで下さい。 "incorrect permissions on cache file" の振る舞いを修正するには、 フォワードされたクレデンシャリングの所有権を適切に変更できるように、 login.krb5 バイナリが認証に使われる必要があります。

rc.conf を以下のように変更する必要もあります。

```
kerberos5_server="/usr/local/sbin/krb5kdc"
kadmind5_server="/usr/local/sbin/kadmind"
kerberos5_server_flags=""
kerberos5_server_enable="YES"
kadmind5_server_enable="YES"
```

これを行うのは、 MIT Kerberos のアプリケーションは、 /usr/local 構造の下にインストールされるためです。

13.5.8. Kerberos で見つかった制限を緩和する

Kerberos は、**All or Nothing** アプローチです。

ネットワーク上で有効なすべてのサービスは、 Kerberos 化されるか、
または、ネットワーク攻撃に対して安全であるべきです。
さもないと、ユーザのクレデンシャルが盗まれ、 利用されることが起きるかもしれません。 この例は、 Kerberos 化されたすべてのリモートシェルです。 パスワードを平文で送るような POP3 メールサーバは変換していません。

Kerberos は、 シングルユーザのワークステーションでの使用を想定しています。

マルチユーザの環境では、 Kerberos は安全ではありません。 チケットは /tmp に保管され、
このチケットは、すべてのユーザが読むことができるためです。
もし、ユーザがコンピュータを他のユーザと同時に共有していると、
他のユーザは、そのユーザのチケットを盗んだり、 コピーが出来てしまいます。

この問題は、`-c` コマンドラインオプションまたは、好ましくは `KRB5CCNAME` 環境変数によって克服されます。この問題への対応には、チケットをユーザのホームディレクトリに保存し、ファイルの許可属性を設定することが一般的に行われます。

KDC は、単一障害点である

設計上、KDC は、マスターパスワードのデータベースと同様に安全である必要があります。KDC では、絶対に他のサービスを走らせるべきではありませんし、物理的に安全であるべきです。Kerberos は、KDC 上で、ファイルとして保存されている同じ "マスター" 鍵で暗号化されたすべてのパスワードを保存しているので、非常に危険です。

マスター鍵が漏洩しても、懸念するほど悪いことにはなりません。マスター鍵は、Kerberos データベースの暗号時にのみ、乱数を生成するためのシードとして使われます。KDC へのアクセスが安全である限りにおいては、マスター鍵を用いて、それほど多くのことはできません。

さらに、KDC が利用できないと、認証ができないため、ネットワークサービスを利用できなくなります。この攻撃による被害は、ひとつのマスタ KDC とひとつまたはそれ以上のスレーブ、そして、セカンダリもしくは PAM を用いたフォールバック認証を注意深く実装することにより軽減できます。

Kerberos の欠点

Kerberos は、ユーザ、ホストおよびサービスの間での認証を可能にしますが、KDC とユーザ、ホストまたはサービスとの間の認証のメカニズムは提供しません。これは、トロイの木馬の kinit(1) が、すべてのユーザ名とパスワードを記録できることを意味しています。security/tripwire のような、ファイルシステムの完全性を確認するためのツールにより、この危険性を軽減することができます。

13.5.9. Kerberos および ssh(1) を用いたアクセスの問題

Kerberos と ssh(1) を使う場合には、両者に関して知っておかねばならない問題がいくつかあります。Kerberos は大変優れた認証プロトコルですが、Kerberos 化された telnet(1) および rlogin(1) には、バイナリストリームを扱うのに不向きになるようなバグがあります。デフォルトでは、Kerberos は `-x` を使わない限りセッションを暗号化してくれません。一方 ssh(1) では、デフォルトですべてを暗号化してくれます。

ssh(1) はとても良く動作しますが、デフォルトで暗号鍵を転送してしまいます。このため、ssh(1) を安全なワークステーションから、安全でないマシンへのアクセスに使っているユーザに、セキュリティリスクを引き起こします。鍵そのものが見えてしまうわけではありませんが、ssh(1) は login している間、転送用ポートを作ります。攻撃者が安全でないマシンの `root` を破ったら、このポートを使って、この暗号鍵でロックが外れる他のマシンへのアクセスを得てしまいます。

可能な時はいつでも、スタッフのログインには Kerberos を組み合せた ssh(1) を使用することを勧めます。ssh(1) は、Kerberos 対応機能と一緒にコンパイルできます。このようにすることで、見えてしまう可能性のある SSH 鍵への依存を減らし、一方で、Kerberos 経由によりパスワードが保護されます。
鍵は、安全なマシンからの自動化されたタスクのみに使用すべきです。Kerberos はこの用途には不向きです。また、SSH の設定で鍵転送をしないようにするか、あるいは authorized_keys の `from=IP/DOMAIN` を使用して、特定のマシンからログインしてきたときのみ鍵が有効にすることをお勧めします。

13.5.10. リソースおよび他の情報源

- The Kerberos FAQ
- Designing an Authentication System: a Dialog in Four Scenes
- RFC 1510, The Kerberos Network Authentication Service (V5)
- MIT Kerberos home page
- Heimdal Kerberos home page

13.6. OpenSSL

FreeBSD には、OpenSSL ツールキットが含まれています。 OpenSSL は、
通常の通信層の上位にあるトランスポート層を暗号化し、
多くのネットワークアプリケーションおよびサービスと組み合わせて使用できます。

OpenSSL は、 メールクライアントの暗号化された認証、
クレジットカードでの支払いといったウェブベースの取引などで使われます。 www/apache22 および mail/claws-mail といった多くの port では、 OpenSSL とともに構築するコンパイルに対応しています。

多くの場合、Ports Collection は、 make の WITH_OPENSSL_BASE が明示的に "yes" に設定されていないと、 security/openssl port の構築を試みます。

FreeBSD に含まれている OpenSSL のバージョンは、Secure Sockets Layer v2/v3 (SSLv2/SSLv3) および Transport Layer Security v1 (TLSv1) ネットワークセキュリティプロトコルに対応しており、
多目的な暗号化ライブラリとして使うことができます。

OpenSSL は、 IDEA アルゴリズムに対応していますが、
合衆国の特許により、デフォルトでは無効になっています。
もし使用したいのであれば、ライセンス条項を必ず確認し、
ライセンス条項に合致するのであれば、 /etc/make.conf において MAKE_IDEA
変数を設定してください。

最も一般的な OpenSSL の利用方法のひとつは、
ソフトウェアアプリケーションが使えるように証明書を提供することです。
これらの証明書により、会社または個人の公開鍵が、
改ざんやなりすましが行われていないことを確認できます。 もし問題となっている証明書が、"認証局" (CA) により検証されなければ、 警告が表示されます。 CA は、VeriSign のような会社で、個人または会社の公開鍵の検証を行えるように、 証明書に署名を行います。
証明書を作成するには費用がかかり、 証明書の使用は必要条件ではありませんが、
証明書を使うことで、 ユーザを安心させることができます。

13.6.1. 証明書の作成

以下のコマンドにより、証明書を作成できます。

```
# openssl req -new -nodes -out req.pem -keyout cert.pem
Generating a 1024 bit RSA private key
..............++++++
```

```
.....................................++++++
writing new private key to 'cert.pem'
-----
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]:US
State or Province Name (full name) [Some-State]:PA
Locality Name (eg, city) []:Pittsburgh
Organization Name (eg, company) [Internet Widgits Pty Ltd]:My Company
Organizational Unit Name (eg, section) []:Systems Administrator
Common Name (eg, YOUR name) []:localhost.example.org
Email Address []:trhodes@FreeBSD.org

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:SOME PASSWORD
An optional company name []:Another Name
```

"Common Name" プロンプト直後に表示されているのは、 ドメイン名です。
このプロンプトでは、検証するサーバ名の入力が必要となります。
ドメイン名以外を入力すると、役に立たない証明書が作成されます。
他のオプションとして、有効期限を指定したり、 別の暗号化アルゴリズムを選択することができます。
オプションの完全なリストは、 openssl(1) で説明されています。

このコマンドを実行したディレクトリには、 2 つのファイルが作成されているはずです。 1 つは、証明書要求 req.pem です。 このファイルを CA に送ると、 CA は含まれている内容を検証し、検証に成功すると、証明書要求に署名を行い、 作成された証明書を送り返します。
もうひとつ、cert.pem と呼ばれるファイルが生成されます。 これは証明書の秘密鍵であり、どのようなことがあっても保護しなくてはなりません。 もし、他の人の手に渡ると、手に入れた人は、ユーザまたはサーバになりすますことができてしまいます。

CA の署名が必要ない場合には、 自己署名証明書を作成できます。 最初に RSA の鍵を生成してください。

```
# openssl dsaparam -rand -genkey -out myRSA.key 1024
```

次に、CA 鍵を生成してください。

```
# openssl gendsa -des3 -out myca.key myRSA.key
```

この鍵を使って証明書を作成してください。

```
# openssl req -new -x509 -days 365 -key myca.key -out new.crt
```

新しく 2 つのファイルがこのディレクトリに作成されます。 プライベート鍵 myca.key および 証明書 new.crt です。 これらのファイルを、好ましくは /etc 以下で、 root のみが読むことのできるディレクトリに置く必要があります。 許可属性は 0700 が適切です。 許可属性は chmod(1) を使って設定できます。

13.6.2. 証明書の使用

証明書の一つの利用方法は、SendmailMTA への接続を暗号化することです。 これにより、 ローカルの MTA 経由でメールを送信するユーザが、 テキスト認証を使用しなくてもすむようになります。

いくつかの MUA は、 ユーザが証明書をローカルにインストールしていないと、 エラーを出力します。 証明書のインストールに関する詳細な情報については、 ソフトウェアに付随の文書を参照してください。

Sendmail を設定するには、以下の行をローカルの .mc ファイルに含めてください。

```
dnl SSL Options
define(`confCACERT_PATH',`/etc/certs')dnl
define(`confCACERT',`/etc/certs/new.crt')dnl
define(`confSERVER_CERT',`/etc/certs/new.crt')dnl
define(`confSERVER_KEY',`/etc/certs/myca.key')dnl
define(`confTLS_SRV_OPTIONS', `V')dnl
```

この例では、 ローカルで証明書および鍵ファイルは、ローカルの /etc/certs/ に置かれています。 ファイルの編集を保存し終わったら、 /etc/mail において make install と入力することで、ローカルの .cf ファイルを再構築する必要があります。 その後、make restart と入力して、Sendmail デーモンを再起動してください。

すべてがうまくいっていれば、 /var/log/maillog にはエラーメッセージは出力されず、 Sendmail がプロセスの一覧に表示されます。

以下は簡単な試験の例で、telnet(1) を使って、 メールサーバに接続しています。

```
# telnet example.com 25
Trying 192.0.34.166...
Connected to  example.com
Escape character is '^]'.
220 example.com ESMTP Sendmail 8.12.10/8.12.10; Tue, 31 Aug 2004 03:41:22 -0400 (EDT)
ehlo example.com
250-example.com Hello example.com [192.0.34.166], pleased to meet you
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-8BITMIME
250-SIZE
250-DSN
250-ETRN
250-AUTH LOGIN PLAIN
250-STARTTLS
250-DELIVERBY
```

```
250 HELP
quit
221 2.0.0 example.com closing connection
Connection closed by foreign host.
```

出力に "STARTTLS" 行が表示されれば、 すべてが適切に機能しています。

13.7. VPN over IPsec

13.7.1. IPsec を理解する

この節では、IPsec を設定する過程を説明します。 IPsec を設定するためには、カスタムカーネルの構築方法をよく知っている必要があります (FreeBSD カーネルのコンフィグレーション をご覧ください)。

IPsec は、インターネットプロトコル (IP) レイヤのトップにあるプロトコルです。二つもしくはそれ以上のホスト間で安全に通信することを可能にします。 FreeBSD の IPsec "ネットワークスタック" は、 IPv4 および IPv6 の両方に対応している KAME 実装をベースとしています。

IPsec は二つのサブプロトコルから構成されます。

- *Encapsulated Security Payload (ESP)*: このプロトコルは、Blowfish, 3DES といった対称暗号アルゴリズムを使ってデータを暗号化することで、サードパーティのインタフェースから IP パケットデータを保護します。
- *Authentication Header AH(AH)*: このプロトコルは、暗号チェックサムを計算し、IP パケットのヘッドフィールドを安全なハッシュ関数でハッシュ化することで、 IP パケットヘッダをサードパーティのインタフェースやなりすましから守ります。ハッシュを含む追加のヘッダが追加され、 パケット情報の検証が可能になります。

ESP および AH は、使用する環境に合わせて、 一緒に使うことも別々に使うこともできます。

IPsec は、直接二つのホスト間のトラフィックを暗号化する *Transport Mode*、もしくは "virtual tunnels" を構築する *Tunnel Mode* のどちらでも用いることができます。後者のモードはより一般的には、 *Virtual Private Network (VPN)* として知られています。 FreeBSD での IPsec サブシステムに関するより詳細な情報については、 ipsec(4) を参照してください。

カーネルに IPsec のサポートを追加するには、カスタムカーネルコンフィグレーションファイルに以下のオプションを追加してください。

```
options   IPSEC        #IP security
device    crypto
```

IPsec のデバッグサポートが必要であれば、 以下のカーネルオプションを追加してください。

```
options   IPSEC_DEBUG  #debug for IP security
```

13.7.2. 家庭と会社間の VPN

VPN の構成についての標準はありません。 VPN は、数多くの技術と共に実装することが可能です。その各技術には、それ自身の長所と短所があります。 この節では、以下のシナリオに対して VPN を実装する戦略について説明します。

- 少なくとも 2 つのサイトがあり、 それぞれのサイトは内部で IP を使っています。
- 2 つのサイトは、FreeBSD で運用されているゲートウェイを通して、インターネットに接続しています。
- それぞれのネットワークのゲートウェイは、 少なくとも一つのパブリック IP アドレスを持っています。
- 2 つのネットワークの内部アドレスは、 パブリックでもプライベート IP アドレスでも構いません。しかしながら、アドレス空間は衝突してはいけません。 たとえば、両方のネットワークが `192.168.1.x` を使ってはいけません。

FreeBSD 上で IPsec を設定する。

最初に Ports Collection から security/ipsec-tools をインストールしてください。 このソフトウェアは、設定をサポートする数多くのアプリケーションを提供します。

次に、パケットをトンネリングし、 両方のネットワークが適切に通信するように、 2 つの gif(4) 疑似デバイスを作成します。 `root` 権限で以下のコマンドを実行してください。ただし、実行する際には、以下のコマンドの中の *internal* および *external* を、 2 つのゲートウェイの内部および外部インタフェースの実際の IP アドレスに置き換えてください。

```
# ifconfig gif0 create
```

```
# ifconfig gif0 internal1 internal2
```

```
# ifconfig gif0 tunnel external1 external2
```

この例では、会社の LAN の外部 IP アドレスを `172.16.5.4`、 内部 IP アドレスを `10.246.38.1` とします。また、家庭 LAN の外部 IP アドレスを `192.168.1.12`、 内部のプライベート IP アドレスを `10.0.0.5` とします。

この説明で分かりにくい場合は、以下の ifconfig(8) コマンドの出力例をご覧ください。

```
Gateway 1:

gif0: flags=8051 mtu 1280
tunnel inet 172.16.5.4 --> 192.168.1.12
inet6 fe80::2e0:81ff:fe02:5881%gif0 prefixlen 64 scopeid 0x6
inet 10.246.38.1 --> 10.0.0.5 netmask 0xffffff00

Gateway 2:
```

```
gif0: flags=8051 mtu 1280
tunnel inet 192.168.1.12 --> 172.16.5.4
inet 10.0.0.5 --> 10.246.38.1 netmask 0xffffff00
inet6 fe80::250:bfff:fe3a:c1f%gif0 prefixlen 64 scopeid 0x4
```

設定が完了したら、両方の内部 IP アドレスは、ping(8) で到達できるようになっているはずです。

```
priv-net# ping 10.0.0.5
PING 10.0.0.5 (10.0.0.5): 56 data bytes
64 bytes from 10.0.0.5: icmp_seq=0 ttl=64 time=42.786 ms
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=19.255 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=20.440 ms
64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=21.036 ms
--- 10.0.0.5 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 19.255/25.879/42.786/9.782 ms

corp-net# ping 10.246.38.1
PING 10.246.38.1 (10.246.38.1): 56 data bytes
64 bytes from 10.246.38.1: icmp_seq=0 ttl=64 time=28.106 ms
64 bytes from 10.246.38.1: icmp_seq=1 ttl=64 time=42.917 ms
64 bytes from 10.246.38.1: icmp_seq=2 ttl=64 time=127.525 ms
64 bytes from 10.246.38.1: icmp_seq=3 ttl=64 time=119.896 ms
64 bytes from 10.246.38.1: icmp_seq=4 ttl=64 time=154.524 ms
--- 10.246.38.1 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 28.106/94.594/154.524/49.814 ms
```

予想通り、プライベートアドレスを使って、 両方のネットワークから ICMP パケットを送受信できます。
次に、どちらのネットワークからもメッセージを送信できるように、
パケットのルーティング情報を両方のゲートウェイに設定する必要があります。
これは以下のコマンドで設定できます。

```
# corp-net# route add 10.0.0.0 10.0.0.5 255.255.255.0
```

```
# corp-net# route add net 10.0.0.0: gateway 10.0.0.5
```

```
# priv-net# route add 10.246.38.0 10.246.38.1 255.255.255.0
```

```
# priv-net# route add host 10.246.38.0: gateway 10.246.38.1
```

これで、ネットワーク内のコンピュータは、
ゲートウェイおよびゲートウェイの奥のコンピュータから到達可能となっています。　もう一度　ping(8)

で確認してください。

```
corp-net# ping 10.0.0.8
PING 10.0.0.8 (10.0.0.8): 56 data bytes
64 bytes from 10.0.0.8: icmp_seq=0 ttl=63 time=92.391 ms
64 bytes from 10.0.0.8: icmp_seq=1 ttl=63 time=21.870 ms
64 bytes from 10.0.0.8: icmp_seq=2 ttl=63 time=198.022 ms
64 bytes from 10.0.0.8: icmp_seq=3 ttl=63 time=22.241 ms
64 bytes from 10.0.0.8: icmp_seq=4 ttl=63 time=174.705 ms
--- 10.0.0.8 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 21.870/101.846/198.022/74.001 ms

priv-net# ping 10.246.38.107
PING 10.246.38.1 (10.246.38.107): 56 data bytes
64 bytes from 10.246.38.107: icmp_seq=0 ttl=64 time=53.491 ms
64 bytes from 10.246.38.107: icmp_seq=1 ttl=64 time=23.395 ms
64 bytes from 10.246.38.107: icmp_seq=2 ttl=64 time=23.865 ms
64 bytes from 10.246.38.107: icmp_seq=3 ttl=64 time=21.145 ms
64 bytes from 10.246.38.107: icmp_seq=4 ttl=64 time=36.708 ms
--- 10.246.38.107 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 21.145/31.721/53.491/12.179 ms
```

トンネリングの設定は以上のように簡単ですが、
リンクを安全にするには、もう少し掘り下げた設定が必要となります。 以下の設定では、事前共有 (PSK) RSA 鍵を使います。 IP アドレスを除けば、両方のゲートウェイの /usr/local/etc/racoon/racoon.conf は同じで、以下のようになります。

```
path    pre_shared_key  "/usr/local/etc/racoon/psk.txt"; #location of pre-shared key
file
log     debug;  #log verbosity setting: set to 'notify' when testing and debugging is
complete

padding # options are not to be changed
{
        maximum_length  20;
        randomize       off;
        strict_check    off;
        exclusive_tail  off;
}

timer   # timing options. change as needed
{
        counter         5;
        interval        20 sec;
        persend         1;
#       natt_keepalive  15 sec;
        phase1          30 sec;
        phase2          15 sec;
```

```
}

listen  # address [port] that racoon will listen on
{
        isakmp          172.16.5.4 [500];
        isakmp_natt     172.16.5.4 [4500];
}

remote  192.168.1.12 [500]
{
        exchange_mode   main,aggressive;
        doi             ipsec_doi;
        situation       identity_only;
        my_identifier   address 172.16.5.4;
        peers_identifier        address 192.168.1.12;
        lifetime        time 8 hour;
        passive         off;
        proposal_check  obey;
#       nat_traversal   off;
        generate_policy off;

                        proposal {
                                encryption_algorithm    blowfish;
                                hash_algorithm          md5;
                                authentication_method   pre_shared_key;
                                lifetime time           30 sec;
                                dh_group                1;
                        }
}

sainfo  (address 10.246.38.0/24 any address 10.0.0.0/24 any)    # address
$network/$netmask $type address $network/$netmask $type ( $type being any or esp)
{                               # $network must be the two internal networks you are
joining.
        pfs_group       1;
        lifetime        time    36000 sec;
        encryption_algorithm    blowfish,3des,des;
        authentication_algorithm        hmac_md5,hmac_sha1;
        compression_algorithm   deflate;
}
```

利用可能なオプションの説明については、 racoon のマニュアルページを参照してください。

FreeBSD および racoon がホスト間のネットワークトラフィックを暗号化、復号化できるようにするには、 Security Policy Database (SPD) の設定が必要です。

これは、会社のゲートウェイ上で、以下のようなシェルスクリプトで設定できます。このファイルをシステムの初期化中に使われるようにするには、 /usr/local/etc/racoon/setkey.conf に保存する必要があります。

```
flush;
spdflush;
# To the home network
spdadd 10.246.38.0/24 10.0.0.0/24 any -P out ipsec esp/tunnel/172.16.5.4-
192.168.1.12/use;
spdadd 10.0.0.0/24 10.246.38.0/24 any -P in ipsec esp/tunnel/192.168.1.12-
172.16.5.4/use;
```

設定ファイルを適切に置くと、以下のコマンドにより、 両方のゲートウェイ上で racoon を起動できます。

```
# /usr/local/sbin/racoon -F -f /usr/local/etc/racoon/racoon.conf -l
/var/log/racoon.log
```

出力は以下のようになるでしょう。

```
corp-net# /usr/local/sbin/racoon -F -f /usr/local/etc/racoon/racoon.conf
Foreground mode.
2006-01-30 01:35:47: INFO: begin Identity Protection mode.
2006-01-30 01:35:48: INFO: received Vendor ID: KAME/racoon
2006-01-30 01:35:55: INFO: received Vendor ID: KAME/racoon
n2006-01-30 01:36:04: INFO: ISAKMP-SA established 172.16.5.4[500]-192.168.1.12[500]
spi:623b9b3bd2492452:7deab82d54ff704a
2006-01-30 01:36:05: INFO: initiate new phase 2 negotiation:
172.16.5.4[0]192.168.1.12[0]
2006-01-30 01:36:09: INFO: IPsec-SA established: ESP/Tunnel 192.168.1.12[0]-
>172.16.5.4[0] spi=28496098(0x1b2d0e2)
2006-01-30 01:36:09: INFO: IPsec-SA established: ESP/Tunnel 172.16.5.4[0]-
>192.168.1.12[0] spi=47784998(0x2d92426)
2006-01-30 01:36:13: INFO: respond new phase 2 negotiation:
172.16.5.4[0]192.168.1.12[0]
2006-01-30 01:36:18: INFO: IPsec-SA established: ESP/Tunnel 192.168.1.12[0]-
>172.16.5.4[0] spi=124397467(0x76a279b)
2006-01-30 01:36:18: INFO: IPsec-SA established: ESP/Tunnel 172.16.5.4[0]-
>192.168.1.12[0] spi=175852902(0xa7b4d66)
```

トンネリングが適切に行われているかどうかを確認するため、 別のコンソール上で tcpdump(1) を使い、以下のようなコマンドでネットワークの通信を確認してください。 ただし、以下の例の `em0` の部分は、必要に応じて使用しているネットワークインタフェースに置き換えてください。

```
# tcpdump -i em0 host 172.16.5.4 and dst 192.168.1.12
```

以下のようなデータがコンソールに表示されます。
もし、表示されない場合は、設定に何か問題があるので、
表示されるデータを使ってデバッグする必要があります。

```
01:47:32.021683 IP corporatenetwork.com > 192.168.1.12.privatenetwork.com:
ESP(spi=0x02acbf9f,seq=0xa)
01:47:33.022442 IP corporatenetwork.com > 192.168.1.12.privatenetwork.com:
ESP(spi=0x02acbf9f,seq=0xb)
01:47:34.024218 IP corporatenetwork.com > 192.168.1.12.privatenetwork.com:
ESP(spi=0x02acbf9f,seq=0xc)
```

これで　2　つのネットワークは、　1　つのネットワークのように利用できます。　多くの場合、両方のネットワークはファイアウォールにより保護されています。　両方を流れる通信を許可するには、パケットが両方を行き来できるようにルールを追加する必要があります。　ipfw(8) を使ったファイアウォールの場合は、
ファイアウォールの設定ファイルに、以下の行を追加してください。

```
ipfw add 00201 allow log esp from any to any
ipfw add 00202 allow log ah from any to any
ipfw add 00203 allow log ipencap from any to any
ipfw add 00204 allow log udp from any 500 to any
```

ルール番号は、 現在のホストの設定によっては変更する必要があるでしょう。

pf(4) または ipf(8) を使用しているシステムでは、 以下のルールで上手くいくでしょう。

```
pass in quick proto esp from any to any
pass in quick proto ah from any to any
pass in quick proto ipencap from any to any
pass in quick proto udp from any port = 500 to any port = 500
pass in quick on gif0 from any to any
pass out quick proto esp from any to any
pass out quick proto ah from any to any
pass out quick proto ipencap from any to any
pass out quick proto udp from any port = 500 to any port = 500
pass out quick on gif0 from any to any
```

最後に、システムの初期化中に VPN が起動するように、以下の行を /etc/rc.conf に追加してください。

```
ipsec_enable="YES"
ipsec_program="/usr/local/sbin/setkey"
ipsec_file="/usr/local/etc/racoon/setkey.conf" # allows setting up spd policies on
boot
racoon_enable="yes"
```

13.8. OpenSSH

OpenSSH　はリモートマシンへのセキュアなアクセスに使われるネットワーク接続ツールの集合です。また、TCP/IP　接続を　OpenSSH　接続経由でセキュアにトンネル/フォワードすることもできます。

OpenSSH はすべてのトラフィックを暗号化し、盗聴や接続の乗っ取り等のネットワークレベルの攻撃を事実上無効化します。

OpenSSH は OpenBSD プロジェクトによって維持管理されており、FreeBSD にはデフォルトでインストールされています。 OpenSSH は、SSH バージョン 1 と 2 の両方に互換性があります。

13.8.1. OpenSSH を使うことの利点

データがネットワークを平文で流れてしまうと、
ネットワークをクライアントとサーバの間のどこかで盗聴することで、あなたのユーザ/パスワード情報やセション中を流れるデータを盗むことが可能です。OpenSSH はこれらを予防する為にさまざまな認証と暗号化の方法を提供します。

13.8.2. SSH サーバを有効にする

sshd(8) が有効になっているかどうかを確認するには、 /etc/rc.conf の以下の行を確認してください。

```
sshd_enable="YES"
```

この設定により、次のシステムの初期化時に OpenSSH のデーモンプログラムである sshd(8) が起動します。 もしくは service(8) を使って、すぐに OpenSSH を起動することもできます。

```
# service sshd start
```

13.8.3. SSH クライアント

ssh(1) を使って、sshd(8) が動いているシステムに接続するには、ログインをするユーザ名とホストを指定してください。

```
# ssh user@example.com
Host key not found from the list of known hosts.
Are you sure you want to continue connecting (yes/no)? yes
Host 'example.com' added to the list of known hosts.
user@example.com's password: *******
```

SSH はクライアントが接続した時、サーバの信頼性の検証のために鍵指紋システム (key fingerprint system) を利用します。 初めての接続の際に、ユーザは yes と入力することを要求されます。 これ以降の login では保存されていた鍵指紋を照合することで検証が行われ、ssh(1) クライアントは保存されていた鍵指紋が login しようとした際に送られてきたものと異なっていた場合には警告を表示します。指紋は ~/.ssh/known_hosts に保存されます。

デフォルトでは、sshd(8) の最近の版では SSH v2 の接続のみを受け付けるように設定されています。クライアントは可能であればバージョン 2 を用い、バージョン 1 にフォールバックします。クライアントは、プロトコル v1 と v2 についてそれぞれ、引数 -1 または -2 を渡すことで、利用するプロトコルを指定できます。 クライアントにおけるバージョン 1 への互換性は、

古いバージョンへの上位互換のために維持されています。

13.8.4. Secure copy

ローカルのファイルをリモートマシンへ、
あるいはリモートマシンのファイルをローカルに安全な方法でコピーするには、 scp(1)
を使用してください。

```
#  scp user@example.com:/COPYRIGHT COPYRIGHT
user@example.com's password: *******
COPYRIGHT            100% |*****************************|  4735
00:00
#
```

前回の例でこのホストの指紋がすでに保存されていれば この scp(1) を使う時に検証が行なわれます。

scp(1) に渡される引数は、cp(1) のものと似ており、コピーするファイル (1 つまたは複数) が 1 つめの引数になり、コピー先が 2 つめの引数になります。 ファイルはネットワーク越しに SSH 接続を通して送られるので、 引数に指定するファイルに `user@host:<path_to_remote_file>` という形式をとるものがあります。

13.8.5. 設定

システム全体の設定ファイルは、OpenSSH デーモン、クライアントの両方とも /etc/ssh にあります。

ssh_config はクライアントの動作設定、 sshd_config はデーモンの動作設定を行ないます。
それぞれのファイル毎にマニュアルページが用意されており、
利用可能な設定オプションについて説明されています。

13.8.6. ssh-keygen(1)

パスワードの代わりに ssh-keygen(1) を使ってユーザの認証用の DSA または RSA 暗号鍵を作ることができます。

```
% ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_dsa):
Created directory '/home/user/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_dsa.
Your public key has been saved in /home/user/.ssh/id_dsa.pub.
The key fingerprint is:
bb:48:db:f2:93:57:80:b6:aa:bc:f5:d5:ba:8f:79:17 user@host.example.com
```

ssh-keygen(1) は認証に使う為の公開鍵と秘密鍵のペアを作ります。 DSA または RSA 鍵に応じて、秘密鍵は ~/.ssh/id_dsa または ~/.ssh/id_rsa に保存され、 公開鍵は ~/.ssh/id_dsa.pub または ~/.ssh/id_rsa.pub にそれぞれ保存されます。 公開鍵はセットアップのために、 DSA または RSA のどちらを使う場合にも、 リモートマシンの ~/.ssh/authorized_keys に含まれてなければなりません。

この設定により、パスワードに代わり、 SSH 鍵を使ってリモートマシンに接続できるようになります。

多くのユーザは、鍵が設計上安全と信じ、 パスフレーズなしに鍵を利用しています。 このような使用方法は 危険 です。 管理者が鍵にパスフレーズが設定されているかを確認する方法は、 手動で鍵を調べる方法です。 秘密鍵のファイルに `ENCRYPTED` という単語が含まれている場合には、 鍵の所有者は、パスフレーズを使用しています。 弱いパスフレーズが使われている間、 少なくともシステムが危険にさらされているときには、 他のサイトへのアクセスには、 あるレベルでのパスワード類推が必要となります。 さらに、公開鍵ファイルに `from` を含めることで、 エンドユーザをより安全にできます。 たとえば、 `ssh-rsa` または `rsa-dsa` の前に、 `from="192.168.10.5` を加えることで、 この IP を持つホストからのユーザのみがアクセスできるようになります。

ssh-keygen(1) でパスフレーズを使っている場合は、 秘密鍵を使うためにユーザは毎回パスフレーズを入力する必要があります。 長いパスフレーズを毎回入力しなくてはならない負担は、 ssh-agent(1) を使うと軽減できます。 これについては、 SSH Agent による鍵のキャッシュ で説明されています。

OpenSSH のバージョンによって、 オプションやファイルに違いが出てくることがあります。 ssh-keygen(1) を参照して、 問題が起こることを避けてください。

13.8.7. SSH Agent による鍵のキャッシュ

パスフレーズを毎回入力することなしに、 SSH 鍵を利用できるようにメモリに読み込むには、 ssh-agent(1) および ssh-add(1) を使用してください。

ssh-agent(1) は、 読み込まれた秘密鍵による認証を取り扱います。 ssh-agent(1) は他のアプリケーションの起動に用いられる必要があります。 基本的なレベルではシェル、 またはウィンドウマネージャを起動します。

シェル上で ssh-agent(1) を使うには、 引数としてシェルを起動してください。 次に、ssh-add(1) を実行し、 秘密鍵のパスフレーズを入力することにより、 鍵を追加してください。 一度この過程を終えてしまえば、ユーザは、 対応する公開鍵が置かれているホストに ssh(1) でログインできるようになります。 以下はその例です。

```
% ssh-agent csh
% ssh-add
Enter passphrase for /home/user/.ssh/id_dsa:
Identity added: /home/user/.ssh/id_dsa (/home/user/.ssh/id_dsa)
%
```

Xorg 上で ssh-agent(1) を使うには、 ssh-agent(1) への呼び出しが ~/.xinitrc に置かれている必要があります。 これにより、Xorg 上で起動されるすべてのプログラムにおいて、 ssh-agent(1) サービスが提供されるようになります。 ~/.xinitrc の例は以下となります。

```
exec ssh-agent startxfce4
```

これで、Xorg を開始するときにはいつでも ssh-agent(1) が起動され、 このプログラムから XFCE が起動されます。 Xorg を再起動した後は有効になりますので、 ssh-add(1) を実行して、 すべての SSH 鍵を読み込ませてください。

13.8.8. SSH トンネリング

OpenSSH
は暗号化されたセッションの中に他のプロトコルをカプセル化するトンネルを作ることができます。

以下のコマンドは ssh(1) で telnet(1) 用のトンネルを作成します。

```
% ssh -2 -N -f -L 5023:localhost:23 user@foo.example.com
%
```

この例では、以下のオプションを使っています。

`-2`
: サーバへの接続に ssh(1) バージョン 2 を使うことを指示します。

`-N`
: はトンネルだけでコマンドはないことを示します。 省略されると ssh(1) は通常のセッションを開始します。

`-f`
: ssh(1) にバックグラウンド実行を強制します。

`-L`
: ローカルトンネルを *localport:remotehost:remoteport* という形式で指定します。

`user@foo.example.com`
: 指定したリモート SSH サーバへログインに用いるログイン名。

SSH のトンネルは `localhost` の指定されたポートに listen するソケットを作ることで実現されています。 SSH はローカルのホスト/ポートで受けた接続すべてを SSH 接続経由で指定されたリモートホストのポートへ転送します。

この例では、`localhost` のポート *5023* がリモートマシンの `localhost` のポート *23* に転送されるようになっています。 *23* は telnet(1) で用いられるので、これは SSH トンネルを通る暗号化された man.telnet.1; セッションを作ります。

このようにして SMTP や POP3 および FTP といったセキュアではない TCP プロトコルをカプセル化できます。

例 *27. ssh(1) を用いた SMTP 用の安全なトンネルの作成*

```
% ssh -2 -N -f -L 5025:localhost:25 user@mailserver.example.com
```

```
user@mailserver.example.com's password: *****
% telnet localhost 5025
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 mailserver.example.com ESMTP
```

ssh-keygen(1) と別のユーザアカウントを組み合わせて使うことでより透過的な SSH のトンネル環境を作ることができます。 パスワードを入力するところで暗号鍵を使い、トンネルは別のユーザ権限で実行することが可能です。

実用的な **SSH** トンネルの例

POP3 サーバへの安全な接続

ここでの例は、外部からの接続を受ける SSH サーバがあるとします。 同じネットワークには、POP3 サーバが動いているメールサーバがあるとします。 電子メールを安全なやり方で見るようにするには、SSH サーバへの SSH 接続を行い、 メールサーバへのトンネルを作成することです。

```
% ssh -2 -N -f -L 2110:mail.example.com:110 user@ssh-server.example.com
user@ssh-server.example.com's password: ******
```

トンネルが作成されて動作したら、 メールクライアントに対し localhost のポート 2110 に POP3 リクエストを送るように指示してください。 そこへの接続は、トンネルを経由して安全に mail.example.com に転送されます。

厳格なファイアウォールをすり抜ける

内向けおよび外向きの接続両方をフィルタするファイアウォールルールを課すネットワーク管理者もいます。 たとえば、 リモートのマシンからのアクセスに、ssh(1) および web サーフィンのための 22 番および 80 番ポートにしか接続させてもらえないかもしれません。 この場合 22 または 80 番以外を使う他のサービスへのアクセスを妨げます。

それに対する解決策は、
あなたが接続しているネットワークのファイアウォールの外部にあるマシンに対して SSH 接続を行い、希望するサービスへのトンネルに利用することです。

```
% ssh -2 -N -f -L 8888:music.example.com:8000 user@unfirewalled-system.example.org
user@unfirewalled-system.example.org's password: *******
```

この例では、ストリーミング Ogg Vorbis クライアントを localhost の 8888 番ポートに向けると、music.example.com の 8000 番ポートに転送され、ファイアウォールをすり抜けられます。

13.8.9. AllowUsers オプション

ログインできるユーザや接続元を AllowUsers を使って制限することは、通常は良い考えです。 たとえば、 root が 192.168.1.32 からのみログインできるようにするには、 以下の行を /etc/ssh/sshd_config に追加してください。

```
AllowUsers root@192.168.1.32
```

admin がどこからでもログインできるようにするには、 ユーザ名そのものを記述してください。

```
AllowUsers admin
```

複数のユーザは、以下のように同じ行に追加してください。

```
AllowUsers root@192.168.1.32 admin
```

注意すべきことは、
このコンピュータにログインする必要のあるすべてのユーザを指定することです。
設定されていないと、そのユーザはログインできなくなります。

/etc/ssh/sshd_config への変更が終わったら、 以下を実行して、設定ファイルを sshd(8) に読み込ませてください。

```
# service sshd reload
```

13.8.10. もっと詳しく知りたい人へ

OpenSSH ウェブサイト

クライアントオプションについて ssh(1), scp(1), ssh-keygen(1), ssh-agent(1), ssh-add(1) および ssh_config(5)

サーバオプションについて sshd(8), sftp-server(8), sshd_config(5)

13.9. ファイルシステムアクセス制御リスト (ACL)

アクセス制御リスト (ACL) は、標準的な UNIX® のパーミッションモデルを、 POSIX®.1e に互換する方法で拡張しています。 これにより、管理者がより洗練されたセキュリティモデルを利用し、その恩恵を受けられるようになります。

FreeBSD の GENERIC カーネルは、 UFS ファイルシステム用の ACL サポートを提供します。
カスタムカーネルをコンパイルして使用するユーザは、
カスタムカーネルのコンフィグレーションファイルに以下を追加してください。

```
options UFS_ACL
```

もしこのオプションが組み込まれていなければ、ACL
に対応したファイルシステムをマウントしようとすると、 警告が表示されます。ACL は、ファイルシステムの拡張属性が有効になっていることに依存しています。 拡張属性は、UFS2 でネイティブ対応されています。

UFS1 に拡張属性を付すように設定するのは、 UFS2 よりも高いレベルの管理オーバヘッドが必要になります。 また、UFS2 における拡張属性のパフォーマンスも大きく上がっています。 そのため、アクセス制御リストを利用する上では UFS2 を使うことが推奨されます。

ACL は、マウント時の管理フラグ `acls` で有効にされます。 これは /etc/fstab に記述できます。 マウント時のフラグは、tunefs(8) を使って、ファイルシステムヘッダのスーパブロックにある ACL フラグを変更するという方法で、 常に自動で設定されるようになります。一般的には、下記の理由からスーパブロックフラグを使う方がよいでしょう。

- マウント時に指定した ACL フラグは `mount -u` による再マウントでは変更できません。 完全に umount(8) した上で、新たに mount(8) するしかありません。これは、起動後にルートファイルシステムで ACL を有効にできないことを意味します。 また、ファイルシステムを利用し始めた後では、その配列を変えられないことも意味しています。
- スーパブロックフラグを設定すると、fstab に記述されていなかったり、デバイスの順番が変わってしまっても、常に ACL が有効な状態でマウントされます。 こうすることで、ファイルシステムを ACL を有効にしないままマウントしてしまい、ACL が正しくないかたちで強制されるセキュリティの問題を防ぎます。

予期せず ACL を有効にしないでマウントしてしまうことを防ぐことが望まれます。 ACL を有効にし、その後無効にしてから、拡張属性を取り消さないでまた有効にしてしまうと、 大変な状況になってしまいます。 一般的には、一度ファイルシステムで ACL を有効にしたら、無効にすべきではありません。そうしてしまうと、ファイル保護がシステムのユーザの意図と齟齬をきたす可能性があるばかりか、 ACL を再度有効にすると、 それまでパーミッションが変更されてきたファイルに古い ACL を割り当ててしまい、 予想しない動作につながることも考えられます。

ACL を有効にしたファイルシステムは、 パーミッション設定の表示に `+` (プラス) 記号がつきます。例えば、次のようになります。

```
drwx------  2 robert  robert  512 Dec 27 11:54 private
drwxrwx---+ 2 robert  robert  512 Dec 23 10:57 directory1
drwxrwx---+ 2 robert  robert  512 Dec 22 10:20 directory2
drwxrwx---+ 2 robert  robert  512 Dec 27 11:57 directory3
drwxr-xr-x  2 robert  robert  512 Nov 10 11:54 public_html
```

この例では、ディレクトリ directory1, directory2 および directory3 のすべてで ACL が働いています。一方 public_html は対象外です。

13.9.1. ACL を利用する

getfacl(1) は、 ファイルシステムの ACL を表示します。 たとえば、test の ACL 設定を表示するには、以下のコマンドを実行してください。

```
% getfacl test
    #file:test
    #owner:1001
    #group:1001
    user::rw-
    group::r--
    other::r--
```

このファイルの ACL 設定を変更するには、 setfacl(1) を使用してください。

```
% setfacl -k test
```

ファイルまたはファイルシステムから、 現在設定されている ACL をすべて取り除くには、-k を使ってください。 しかしながら、より好ましい方法は、 -b を使う方法です。 このオプションを使うと、ACL が動作するのに必要な基本のフィールドは残ります。

```
% setfacl -m u:trhodes:rwx,group:web:r--,o::--- test
```

この例では、-m は、デフォルト ACL エントリを修正するために使われています。 先ほどのコマンドで設定は削除されたため、 定義されたエントリはありません。 このコマンドは、デフォルトオプションに戻し、 指定したオプションを割り当てます。 システムに存在しないユーザまたはグループを追加すると、 Invalid argument エラーが出力されてしまいます。

13.10. サードパーティ製ソフトウェアのセキュリティ問題を監視する

近年、セキュリティの分野では、 脆弱性の評価方法に関して多くの改善が行われています。 今日ではどのオペレーティングシステムにおいても、 システムへの侵入の脅威は、 サードパーティ製ユーティリティをインストールし、 設定するほどに増加していきます。

脆弱性を評価することは、セキュリティにおいて主要な要素です。 FreeBSD は、ベースシステムに対して勧告を発行していますが、 すべてのサードパーティ製ユーティリティに対して勧告を発行することは、 FreeBSD プロジェクトの能力を超えています。 サードパーティ製ユーティリティに関わる脆弱性を軽減し、 管理者に対し、既知のセキュリティ問題について警告する方法が存在します。 FreeBSD には、portaudit と呼ばれる追加のユーティリティが、 この目的のために用意されています。

ports-mgmt/portaudit port は、FreeBSD セキュリティチームおよび ports 開発者がアップデートし、管理している、 既知のセキュリティ問題に対するデータベースを入手します。

Ports Collection から portaudit をインストールするには、以下のように実行してください。

```
# cd /usr/ports/ports-mgmt/portaudit && make install clean
```

インストールの途中で、 periodic(8) の設定ファイルはアップデートされ、
毎日のセキュリティに関するスクリプトの実行中に portaudit が出力するように設定されます。
毎日のセキュリティに関するスクリプトの実行結果のメールが読めることを確認してください。
このメールは、root アカウントに送られます。 他の設定は必要ありません。

インストールが終わったら、管理者は以下のコマンドを実行することで、
データベースをアップデートし、インストールされている package の脆弱性を調べることができます。

```
# portaudit -Fda
```

データベースは、 periodic(8) の実行中に自動的にアップデートされます。
先程のコマンドの実行は任意で、
データベースを手動で直ちにアップデートするときに使われます。

Ports Collection からインストールされたサードパーティ製ユーティリティを監査するには、
管理者は以下のコマンドを実行する必要があります。

```
# portaudit -a
```

portaudit は、インストールされている package の中で、
脆弱性のあるものについて以下のようなメッセージを出力します。

```
Affected package: cups-base-1.1.22.0_1
Type of problem: cups-base -- HPGL buffer overflow vulnerability.
Reference: <http://www.FreeBSD.org/ports/portaudit/40a3bca2-6809-11d9-a9e7-
0001020eed82.html>

1 problem(s) in your installed packages found.

You are advised to update or deinstall the affected package(s) immediately.
```

表示されている URL をウェブブラウザで開くと、管理者は、
脆弱性についてより多くの情報を得ることができます。 ここでの出力では、影響するバージョンが
FreeBSD の port バージョンにより示され、
セキュリティ勧告を含む他のウェブサイトが含まれています。

portaudit は強力で、 portmaster port と共に使うときわめて有用なユーティリティです。

13.11. FreeBSD セキュリティ勧告

多くの高品質なオペレーティングシステムのプロジェクト同様、 FreeBSD
プロジェクトはセキュリティチームを持っています。 このチームは責任をもって、各 FreeBSD
リリースに対する保守終了 (End-of-Life (EoL))
日を決めたり、サポートされているリリースに対して、EoL
までセキュリティアップデートを提供しています。 FreeBSD
セキュリティチームおよびサポートされているリリースについての情報は、FreeBSD セキュリティページ
で提供されています。

セキュリティチームの仕事の 1 つは FreeBSD オペレーティングシステムのセキュリティ脆弱性に対応することです。
脆弱性が確認されると、セキュリティチームは脆弱性を修正するために必要となる手続きを検証し、修正を含めるようにソースコードをアップデートします。 その後、詳細を "セキュリティ勧告" として発行しています。 セキュリティ勧告は、 FreeBSD ウェブサイト で公開され、 FreeBSD security notifications メーリングリスト, FreeBSD security メーリングリスト, および FreeBSD announcements メーリングリスト メーリングリストに投稿されます。

この章では、セキュリティ勧告とはどのようなものか説明します。

13.11.1. セキュリティ勧告はどのようなものか？

以下は FreeBSD セキュリティ勧告の例です。

```
=============================================================================
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA512

=============================================================================
FreeBSD-SA-14:04.bind                                        Security Advisory
                                                          The FreeBSD Project

Topic:          BIND remote denial of service vulnerability

Category:       contrib
Module:         bind
Announced:      2014-01-14
Credits:        ISC
Affects:        FreeBSD 8.x and FreeBSD 9.x
Corrected:      2014-01-14 19:38:37 UTC (stable/9, 9.2-STABLE)
                2014-01-14 19:42:28 UTC (releng/9.2, 9.2-RELEASE-p3)
                2014-01-14 19:42:28 UTC (releng/9.1, 9.1-RELEASE-p10)
                2014-01-14 19:38:37 UTC (stable/8, 8.4-STABLE)
                2014-01-14 19:42:28 UTC (releng/8.4, 8.4-RELEASE-p7)
                2014-01-14 19:42:28 UTC (releng/8.3, 8.3-RELEASE-p14)
CVE Name:       CVE-2014-0591

For general information regarding FreeBSD Security Advisories,
including descriptions of the fields above, security branches, and the
following sections, please visit <URL:http://security.FreeBSD.org/>.

I.   Background

BIND 9 is an implementation of the Domain Name System (DNS) protocols.
The named(8) daemon is an Internet Domain Name Server.

II.  Problem Description

Because of a defect in handling queries for NSEC3-signed zones, BIND can
crash with an "INSIST" failure in name.c when processing queries possessing
```

```
certain properties.  This issue only affects authoritative nameservers with
at least one NSEC3-signed zone.  Recursive-only servers are not at risk.

III. Impact

An attacker who can send a specially crafted query could cause named(8)
to crash, resulting in a denial of service.

IV.  Workaround

No workaround is available, but systems not running authoritative DNS service
with at least one NSEC3-signed zone using named(8) are not vulnerable.

V.   Solution

Perform one of the following:

1) Upgrade your vulnerable system to a supported FreeBSD stable or
release / security branch (releng) dated after the correction date.

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 8.3, 8.4, 9.1, 9.2-RELEASE and 8.4-STABLE]
# fetch http://security.FreeBSD.org/patches/SA-14:04/bind-release.patch
# fetch http://security.FreeBSD.org/patches/SA-14:04/bind-release.patch.asc
# gpg --verify bind-release.patch.asc

[FreeBSD 9.2-STABLE]
# fetch http://security.FreeBSD.org/patches/SA-14:04/bind-stable-9.patch
# fetch http://security.FreeBSD.org/patches/SA-14:04/bind-stable-9.patch.asc
# gpg --verify bind-stable-9.patch.asc

b) Execute the following commands as root:

# cd /usr/src
# patch < /path/to/patch

Recompile the operating system using buildworld and installworld as
described in <URL:https://www.FreeBSD.org/handbook/makeworld.html>.

Restart the applicable daemons, or reboot the system.

3) To update your vulnerable system via a binary patch:

Systems running a RELEASE version of FreeBSD on the i386 or amd64
```

```
platforms can be updated via the man:freebsd-update[8] utility:

# freebsd-update fetch
# freebsd-update install

VI.  Correction details

The following list contains the correction revision numbers for each
affected branch.

Branch/path                                                      Revision
- -------------------------------------------------------------------------
stable/8/                                                         r260646
releng/8.3/                                                       r260647
releng/8.4/                                                       r260647
stable/9/                                                         r260646
releng/9.1/                                                       r260647
releng/9.2/                                                       r260647
- -------------------------------------------------------------------------

To see which files were modified by a particular revision, run the
following command, replacing NNNNNN with the revision number, on a
machine with Subversion installed:

# svn diff -cNNNNNN --summarize svn://svn.freebsd.org/base

Or visit the following URL, replacing NNNNNN with the revision number:

<URL:https://svnweb.freebsd.org/base?view=revision&revision=NNNNNN>

VII. References

<URL:https://kb.isc.org/article/AA-01078>

<URL:http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0591>

The latest revision of this advisory is available at
<URL:http://security.FreeBSD.org/advisories/FreeBSD-SA-14:04.bind.asc>
-----BEGIN PGP SIGNATURE-----

iQIcBAEBCgAGBQJS1ZTYAAoJEO1n7NZdz2rnOvQP/2/68/s9Cu35PmqNtSZVVxVG
ZSQP5EGWx/lramNf9566iKxOrLRMq/h3XWcC4goVd+gZFrvITJSVOWSa7ntDQ7TO
XcinfRZ/iyiJbs/Rg2wLHc/t5oVSyeouyccqODYFbOwOlk35JjOTMUG1YcX+Zasg
ax8RV+7Zt1QSBkMlOz/myBLXUjlTZ3Xg2FXVsfFQW5/g2CjuHpRSFx1bVNX6ysoG
9DT58EQcYxIS8WfkHRbbXKh9I1nSfZ7/Hky/kTafRdRMrjAgbqFgHkYTYsBZeav5
fYWKGQRJulYfeZQ90yMTvlpF42DjCC3uJYamJnwDIu8OhS1WRBI8fQfr9DRzmRua
OK3BK9hUiScDZOJB6OqeVzUTfe7MAA4/UwrDtTYQ+PqAenv1PK8DZqwXyxA9ThHb
zKO3OwuKOVHJnKvpOcr+eNwo7jbnHlis0oBksj/mrq2P9m2ueF9gzCiq5Ri5Syag
Wssb1HUoMGwqU0roS8+pRpNC8YgsWpsttvUWSZ8u6Vj/FLeHpiV3mYXPVMaKRhVm
067BA2uj4Th1JKtGleox+Em0R7OFbCc/9aWC67wiqI6KRyit9pYiF3npph+7D5Eq
7zPsUdDd+qc+UTiLp3liCRp5w6484wWdhZO6wRtmUgxGjNkxFoNnX8CitzF8AaqO
```

```
UWWemqWuz3lAZuORQ9KX
=OQzQ
-----END PGP SIGNATURE-----
```

すべてのセキュリティ勧告は以下のフォーマットに基づいています。

- 各セキュリティ勧告には、セキュリティオフィサの PGP 鍵により署名されています。セキュリティオフィサの公開鍵は、OpenPGP 鍵 で検証できます。
- FreeBSD のセキュリティ勧告の名前は常に FreeBSD-SA- で始まり、次に年を表す 2 桁の数字 (14:)、年毎の勧告の番号 (04.) 、そして影響するアプリケーションまたはサブシステムの名前 (bind) が続きます。この例は、2014 年の 4 番目の勧告で BIND に影響する脆弱性に対する勧告を意味しています。
- Topic フィールドでは、脆弱性について明記されています。
- Category では、脆弱性がシステムのどの部分に影響するかを示します。 core, contrib または ports のどれかが示されます。 core カテゴリは、 FreeBSD オペレーティングシステムの core コンポーネントに影響する脆弱性であることを意味します。 contrib カテゴリは、BIND のように FreeBSD に取り込まれているソフトウェアに影響する脆弱性であることを意味します。 ports カテゴリは、Ports Collection からインストールされるソフトウェアに影響する脆弱性であることを示しています。
- Module フィールドは、 影響するコンポーネントについて言及します。 この例では bind モジュールに影響することがわかります。
  そのため、この脆弱性は、オペレーティングシステムとともにインストールされたアプリケーションに影響します。
- Announced フィールドには、セキュリティ勧告が発行された日が記載されています。セキュリティチームによりこの問題が存在することが確認され、パッチが FreeBSD ソースコードリポジトリにコミットされたことを意味します。
- Credits フィールドは、脆弱性を発見し、報告した個人または組織を示します。
- Affects フィールドは、この脆弱性がどの FreeBSD リリースに影響するかを示しています。
- Corrected
  フィールドには、脆弱性が修正された日、時間、タイムゾーン、およびリリースが示されます。この括弧の中では、修正がマージされた各ブランチ、およびそのブランチで対応するリリースのバージョン番号が示されています。リリースの識別子には、バージョン番号、可能な場合はパッチレベルが含まれています。パッチレベルは p に番号が続いたものです。この番号はパッチのシーケンス番号で、この番号を確認することで、ユーザはどのパッチまでがシステムに適用されているかを追跡できます。
- CVE Name フィールドは、該当する脆弱性が cve.mitre.org セキュリティ脆弱性データベースに存在する場合に、脆弱性の番号一覧を示します。
- Background フィールドは、影響しているモジュールに関する情報を示します。
- Problem Description フィールドは、脆弱性について説明します。
  問題のあるコードの情報や、このユーティリティがどのように悪意のある使われ方をされうるかといったことが示されます。
- Impact フィールドは、 この問題がシステムに対して、どのような形式の影響を与えるかについて示します。

- Workaround
  フィールドは、何らかの理由により、すぐにシステムにパッチを当てることのできないシステム管理者に対して、回避方法が存在する場合にその方法を提供します。
- Solution
  フィールドは、影響のあるシステムにパッチを当てる手順を提供します。ここではステップごとにシステムにパッチを当て、安全に動作するように、試験され検証された方法が記載されます。
- Correction Details フィールドは、影響する各 Subversion ブランチに対する修正されたコードが取り込まれたリビジョン番号をを示します。
- References フィールドは、脆弱性に関連する他の情報へのソースを提供します。

13.12. プロセスアカウンティング

プロセスアカウンティングは、 管理者が使用されているシステムのリソースを記録したり、
リソースのユーザへの割り当て、
システムのモニタリングおよびユーザのコマンドの最低限の記録を提供します。

これは実際には、長所と短所があります。 長所の一つは、侵入を入り口の時点で絞ることができます。
短所は、プロセスアカウンティングにより生成されるログの量で、
多くのディスク容量を必要とします。この節では、
管理者を対象にプロセスアカウンティングの基礎を説明します。

13.12.1. プロセスアカウンティングを有効にする

プロセスアカウンティングを使用する前に、 以下のコマンドを使って、
プロセスアカウンティングを有効にしておく必要があります。

```
# touch /var/account/acct
# chmod 600 /var/account/acct
# accton /var/account/acct
# echo 'accounting_enable="YES"' >> /etc/rc.conf
```

一度有効に設定すると、アカウンティングは、 CPU の統計、
実行されたコマンドの情報の追跡を開始します。 すべてのアカウンティングログは、
人が読めるような形式ではなく、 sa(8) を使って見ることができます。
オプションを設定せずに実行すると、 sa(8) はユーザコールの数、全経過時間 (分)、 全 CPU、ユーザの時間 (分)、および I/O 操作の平均数などを出力します。

実行されたコマンドに関する情報を見るには、 lastcomm(1) を使ってください。 このコマンドは、
ユーザが特定の ttys(5) で実行したコマンドを出力します。 たとえば、以下のコマンドは ttyp1
ターミナル上で trhodes が実行した ls(1) の使用について、記録されているすべて示します。

```
# lastcomm ls trhodes ttyp1
```

他にも有用なオプションが多くあり、 lastcomm(1), acct(5) および sa(8) で説明されています。

13.13. リソースの制限

FreeBSD
は、個々のユーザが利用できるシステムのリソース容量を制限する方法をいくつも用意しています。
ディスククォータはユーザが使用できるディスク容量を制限します。　　クォータについては
「ディスククォータ」 で説明されています。

その他のリソースの制限とは、ユーザが消費できる
CPU、メモリなどのリソースを制限する手段のことです。
フラットファイルまたはコマンドによりリソースの制限に関わるデータベースを管理できます。
伝統的な方法では、ログインクラスを　　/etc/login.conf　　を編集することにより定義します。
この方法は、現在でも使われていますが、変更を行うには、このファイルの編集、リソースデータベースの再構築、　　/etc/master.passwd
への必要な変更、さらに、パスワードデータベースの再構築といった、複数回に渡るプロセスが必要です。　　この複数回に渡るプロセスは、　　多くのユーザについて設定する必要がある場合には、
大変な時間の浪費につながる可能性があります。

`rctl`　　を用いると、よりきめ細かにリソースの制限を管理する方法を提供できます。
このコマンドは、ユーザだけではなく、プロセスおよび jails に対してもリソースを制限できます。

この節では、リソースを管理する方法について伝統的な方法と高度な方法の両方について説明します。

13.13.1. ログインクラスの設定

伝統的な方法では、ログインクラスおよびログインクラスに適用するリソースの制限は　　/etc/login.conf
で定義します。　　各ユーザアカウントにはログインクラスが割り当てられています　　(デフォルトでは
`default`　　です)。
それぞれのログインクラスには関連するログインケーパビリティの集合が割り当てられています。
ログインケーパビリティとは、　　名称=値　　の組のことで、名称　　は周知の識別子、値　　は、名称
に応じて処理される任意の文字列です。

/etc/login.conf　　を編集する時には　　/etc/login.conf.db
を次のコマンドを実行してアップデートする必要があります。

```
# cap_mkdb /etc/login.conf
```

リソースの制限は、　　2　　つの点で標準的なログインケーパビリティと異なっています。
第一に、どの制限についても、　　ソフト　　リミットと　　_ハード　　_リミットがあります。
ソフトリミットは、ユーザやアプリケーションが調整できますが、
ハードリミットを超えることはできません。　　ユーザはハードリミットを下げることはできますが、
上げることはスーパユーザのみができます。
第二に、ほとんどのリソースの制限は特定のユーザに対してプロセス毎に適用されるものです。

ログインクラスのリソースの制限　　が最もよく使われるリソースの制限です。
利用可能なすべてのリソースの制限およびのログインケーパビリティの詳細については、　　login.conf(5)
に書かれています。

表 *11.* ログインクラスのリソースの制限

リソースの制限	説明
coredumpsize	プログラムが生成する core ファイルのサイズにかかる制限は、 filesize やディスククォータなどの、 ほかのディスク使用に関する制限に従属します。 この制限は、ディスク領域の消費を制御するあまり厳しくない手段としてよく使われています。 ユーザは core ファイルを自分で生成するわけではなく、削除しないことも多いので、 これを設定すれば大きなプログラムが異常終了してもディスクの空きがなくならずに済みます。
cputime	そのユーザのプロセスが消費できる CPU 時間の上限です。 これを超えたプロセスは、カーネルにより終了されます。 これは、消費される CPU 時間 についての制限であって、 top や ps のフィールドで表示される CPU の割合に関するものではありません。
filesize	ユーザが所有できるファイルの大きさの上限です。 ディスククォータ (crossref:disks[quotas,「ディスククォータ」) と違い、この制限はユーザのファイルをすべてまとめた集合にではなく、個々のファイルにかかります。
maxproc	ユーザが実行できるフォアグラウンドとバックグラウンドプロセス数の上限です。 この上限は、 kern.maxproc で指定されたシステムの制限を超えることはできません。 この値をあまり小さな値に設定すると、大きなプログラムをコンパイルする場合のように、複数のプロセスが実行されるようなタスクにおいて、ユーザの生産性が悪化する可能性があります。
memorylocked	1 つのプロセスが mlock(2) によりメインメモリにロックされることを要求できるメモリの最大容量です。 amd(8) のようなシステムで重要なプログラムは、 メインメモリへロックして、システムがスワップする際に、 ディスクのスラッシングを引き起こさないようにします。
memoryuse	どの時点かを問わず、あるプロセスが消費できる最大のメモリ容量です。 これは、メインメモリとスワップの使用量を合わせたものです。 メモリ消費を抑えるための包括的な制限ではありませんが、手始めにはよいでしょう。
openfiles	あるプロセスが開いておける最大のファイル数です。 FreeBSD では、ファイルは、ソケットや IPC チャンネルを表わすのにも使われているので、あまり低い値に設定しないよう注意してください。 これに対応するシステム全体の制限は sysctl(8) kern.maxfiles で定義されます。
sbsize	あるユーザが消費できるネットワークメモリの上限の量です。 これは、ネットワーク通信を制限するのに使えます。
stacksize	プロセスのスタックサイズの上限です。 あるプログラムが使用しうるメモリの量を制限するには、これだけでは十分ではないので、他の制限と組み合わせて使わなければなりません。

リソースの制限を設定するにあたり、ほかにもいくつか覚えておかなければならないことがあります。

- システム起動時に /etc/rc から起動されたプロセスは、 daemon ログインクラスに割り当てられます。
- システムに付属している /etc/login.conf はほとんどの制限について妥当な値になっていますが、すべてのシステムにおいてふさわしいというわけではありません。

制限をあまり緩くするとシステムを悪用しやすくしてしまいますし、厳しくしすぎると生産性を悪化させてしまいます。

- Xorg
  は多くのリソースを使うだけでなく、より多くのプログラムを並行して使うことをユーザに促します。
- 多くの制限は個々のプロセスにかかるもので、一人のユーザにまとめてかかるものではありません。例えば、`openfiles` を 50 に設定することは、ユーザが動かすそれぞれのプロセスが最大 50 個のファイルを開けるということです。 あるユーザが開けるファイルの総数は、 `openfiles` の値に `maxproc` をかけたものになります。 同じことがメモリ消費量にもあてはまります。

リソースの制限と、ログインクラス、 ログインケーパビリティー般についての詳しい情報は、cap.mkdb(1), getrlimit(2) および login.conf(5) をご覧ください。

13.13.2. リソースの制限を有効にして設定する

`kern.racct.enable` をゼロ以外の値に設定してください。
カスタムカーネルには以下のような特別な設定が必要となります。

```
options         RACCT
options         RCTL
```

システムを再起動して新しいカーネルで立ち上げると、`rctl`
を用いてシステムにルールを設定できるようになります。

ルールの構文は、 subject, subject-id, resource および action を使って管理されます。
以下のルールの例を参照してください。

```
user:trhodes:maxproc:deny=10/user
```

この例では、subject は `user`、subject-id は `trhodes`、resource の `maxproc` はプロセスの最大数、そして action は `deny` と設定されており、 新しいプロセスの生成がブロックされます。 これは、ユーザ `trhodes` のプロセスは 10 個に制限され、それ以上のプロセスは作成できないことを意味しています。
他には、コンソールにログを出力したり、 devd(8) に対し通知したり、プロセスに sigterm を送ったりといった action も利用できます。

ルールを追加する際には、注意すべき点がいくつかあります。 上の例では、プロセスの数が `10` に制限されているため、ログインして `screen`
セッションを実行してしまうと、ユーザによる他のタスクの実行はブロックされてしまうでしょう。
リソースの制限が適応されると、エラーが出力されます。 この例では以下のような出力が行われます。

```
% man test
    /usr/bin/man: Cannot fork: Resource temporarily unavailable
eval: Cannot fork: Resource temporarily unavailable
```

他の例としては、jail がメモリの制限を超えることを防ぐことができます。
このルールは以下のように書くことができます。

```
# rctl -a jail:httpd:memoryuse:deny=2G/jail
```

ルールを /etc/rctl.conf に追加すると、再起動してもルールは持続します。
フォーマットは、ルールから最初のコマンドの部分を除いたものとなります。
たとえば、上のルールを追加するには、以下のように追加してください。

```
# Block jail from using more than 2G memory:
jail:httpd:memoryuse:deny=2G/jail
```

ルールを削除するには、`rctl` に対し、リストから削除するように指定してください。

```
# rctl -r user:trhodes:maxproc:deny=10/user
```

rctl(8) には、ルールをすべて削除する方法が記載されています。
しかしながら、特定のユーザのルールをすべて削除するには、以下のようなコマンドを実行してください。

```
# rctl -r user:trhodes
```

`subjects` をコントロールするリソースは他にも多く用意されています。 これらについて知るには、rctl(8) をご覧ください。

Chapter 14. ストレージ

14.1. この章では

この章では、FreeBSD におけるディスクの使用方法を説明します。これにはメモリディスク、ネットワークに接続されたディスク、および標準的な SCSI/IDE 記憶デバイスが含まれます。

この章では、以下の分野について説明します。

- 物理ディスク上のデータ構成について記述するために FreeBSD が使用する用語 (パーティションおよびスライス)
- システムにハードディスクを追加する方法
- メモリディスクのような仮想ファイルシステムを設定する方法
- 使用できるディスク容量を制限するためにクォータを設定する方法
- 攻撃者から保護するためにディスクを暗号化する方法
- FreeBSD で CD や DVD を作成する方法
- バックアップのためのさまざまな記憶メディアオプション
- FreeBSD で利用できるバックアッププログラムの使用方法
- フロッピーディスクにバックアップする方法
- スナップショットとは何か、そしてそれを効果的に使用する方法

14.2. デバイス名

以下は、FreeBSD で対応している物理記憶デバイスとそれに対応するデバイス名のリストです。

表 *12.* 物理ディスクへの名前付け

ドライブの種類	ドライブのデバイス名
IDE ハードドライブ	`ad`
IDE CD-ROM ドライブ	`acd`
SCSI ハードドライブおよび USB 大容量記憶デバイス	`da`
SCSI CD-ROM ドライブ	`cd`
その他の非標準的 CD-ROM ドライブ	ミツミ CD-ROM は `mcd`, Sony CD-ROM は `scd`, 松下/パナソニック CD-ROM は `matcd`
フロッピードライブ	`fd`
SCSI テープドライブ	`sa`
IDE テープドライブ	`ast`
フラッシュドライブ	DiskOnChip® フラッシュデバイスは `fla`

ドライブの種類	ドライブのデバイス名
RAID ドライブ	Adaptec® AdvancedRAID は aacd, Mylex® は mlxd および mlyd, AMI MegaRAID® は amrd, Compaq Smart RAID は idad, 3ware® RAID はtwed

14.3. ディスクの追加

現在一つしかドライブがない計算機に新しく SCSI ディスクを追加したいとしましょう。まずコンピュータの電源を切り、 コンピュータやコントローラ、ドライブの製造元の説明書に従ってドライブを取り付けます。
このあたりの手順は非常に多岐にわたるため、 詳細はこの文書の範囲外です。

root ユーザでログインします。 ドライブの取り付け後は /var/run/dmesg.boot を調べて新しいディスクが見つかっていることを確認しておきます。
この例では、新しく付けたドライブは da1 で、 我々はそれを /1 にマウントしたいとしましょう (もし IDE ドライブを付けようとしているのなら、デバイス名は 4.0 以前のシステムでは wd1, ほとんどの 4.x システムでは ad1 になるでしょう)。

FreeBSD は IBM-PC 互換のコンピュータで動くため、 PC BIOS のパーティションを考慮に入れる必要があります。 これは従来の BSD パーティションとは異なります。PC ディスクは 4 つまでの BIOS パーティションエントリを持つことができます。 もしそのディスクを本当に FreeBSD 専用にしたい場合には 専用 モードで用いることもできます。 そうでない場合には、FreeBSD は PC BIOS パーティションのどれか一つの中に入れることになります。 FreeBSD では、従来の BSD パーティションと混乱しないように PC BIOS パーティションのことをスライスと呼びます。 また、別の OS がインストールされていたコンピュータで使われていたが FreeBSD 専用にするディスク上でもスライスを用いることができます。 これは、他の OS の fdisk ユーティリティを混乱させないためです。

スライスの場合、ドライブは /dev/da1s1e として加えられるでしょう。これは、SCSI ディスクでユニット番号は 1 (二つめの SCSI ディスク), スライスは 1 (PC BIOS のパーティションが 1) で BSD パーティション e, と読みます。 専用ディスクの場合だと単純に /dev/da1e として加えられるでしょう。

14.3.1. sysinstall(8) の利用

1. sysinstall の操作

   sysinstall の使い易いメニューを利用して、新しいディスクのパーティション分けやラベル付けを行なうことができます。 root ユーザでログインするか su コマンドを用いるかして root 権限を取得します。 /stand/sysinstall を実行して Configure メニューに入ります。FreeBSD Configuration Menu の中でスクロールダウンして Fdisk の項目を選びます。

2. fdisk パーティションエディタ

   fdisk では、ディスク全体を FreeBSD で使うために A を入力します。 "remain cooperative

with any future possible operating systems" と聞かれたら YES と答えます。 W で変更をディスクに書き込みます。ここで q と入力して FDISK エディタを抜けます。
次にマスタブートレコードについて聞かれます。
ここでは既に動いているシステムにディスクを追加しようとしているので None を選びます。

3. ディスクラベルエディタ

次に sysinstall を終了し、 もう一度起動する必要があります。同じ手順を踏んで今度は Label オプションを選択し、 Disk Label Editor に入ります。 ここでは従来の BSD パーティションを作成します。 一つのディスクは a から h までのラベルがついた最大 8 つのパーティションを持つことができます。
いくつかのパーティションラベルは特別な用途に用いられます。 a パーティションはルートパーティション (/) です。したがって、システムディスク (つまり起動ディスク) のみに a パーティションがあるべきです。b パーティションはスワップパーティションに用いられ、
複数のディスクにスワップパーティションを作ることができます。 c は専用モードにおけるディスク全体、
もしくはスライスモードにおけるスライス全体を指します。
他のパーティションは汎用的に用いられます。

sysinstall のラベルエディタは、ルートパーティションでもスワップパーティションでもないパーティションには、e パーティションを採用しようとします。ラベルエディタでファイルシステムを作成するには C を入力してください。 FS (ファイルシステム) かスワップかを聞かれたら FS を選びマウントポイント (たとえば /mnt) を入力します。
インストール後のモードでディスクを追加する場合、 sysinstall は /etc/fstab にエントリを追加しないため、
ここで指定するマウントポイントはそれほど重要ではありません。

さて、ディスクに新しいラベルを書き込み、
そこにファイルシステムを作る準備が整いました。早速 W を叩いて実行しましょう。 sysinstall からの、 新しいパーティションをマウントできない、 というエラーは無視してください。Label Editor から抜け、 sysinstall を終了します。

4. 終了

最後に /etc/fstab を編集し、 新しいディスクのエントリを追加します。

14.3.2. コマンドラインユーティリティの利用

スライスの利用

このセットアップ方法では、
すでにコンピュータに他のオペレーティングシステムがインストールされていても
正しく協調動作することが可能で、他のオペレーティングシステムの fdisk ユーティリティを混乱させることもありません。 新しいディスクにインストールする場合は、この方法を用いることが推奨されています。 後述する 専用モード は、そうしなければならない理由がある時にのみ、 利用するようにしてください。

```
# dd if=/dev/zero of=/dev/da1 bs=1k count=1
# fdisk -BI da1 # 新しいディスクの初期化
# disklabel -B -w -r da1s1 auto # ディスクにラベルを付ける
# disklabel -e da1s1 # 作成したディスクラベルを編集し、パーティションを追加する
# mkdir -p /1
# newfs /dev/da1s1e # 作成したすべてのパーティションに対してこれを繰り返す
# mount /dev/da1s1e /1 # パーティションをマウントする
# vi /etc/fstab # /etc/fstab に適切なエントリを追加する
```

IDE ディスクを使う場合は da の部分を ad とします。4.X より前のシステムでは、 (訳注: ad ではなく) wd としてください。

専用モード

新しいドライブを他の OS と共有しない場合には 専用 モードを用いることもできます。このモードはマイクロソフトの OS を混乱させることを憶えておいてください (しかし、それらによって壊されることはありません)。 一方、IBM の OS/2® はどんなパーティションでも見つけたら理解できなくても "専有" します。

```
# dd if=/dev/zero of=/dev/da1 bs=1k count=1
# disklabel -Brw da1 auto
# disklabel -e da1 # `e' パーティションの作成
# newfs -d0 /dev/da1e
# mkdir -p /1
# vi /etc/fstab # /dev/da1e エントリの追加
# mount /1
```

もう一つの方法は次の通り。

```
# dd if=/dev/zero of=/dev/da1 count=2
# disklabel /dev/da1 | disklabel -BrR da1 /dev/stdin
# newfs /dev/da1e
# mkdir -p /1
# vi /etc/fstab # /dev/da1e エントリの追加
# mount /1
```

FreeBSD 5.1-RELEASE から、従来の disklabel(8) プログラムは bsdlabel(8) ユーティリティに置き換えられました。bsdlabel(8) では、使用されていない数多くのオプションやパラメタが削除されました。 たとえば -r オプションは bsdlabel(8) では取り除かれました。詳細については bsdlabel(8) のマニュアルページを参照してください。

14.4. RAID

14.4.1. ソフトウェア RAID

Concatenated Disk Driver (CCD) の設定

大容量記録に関する解決法を選択する際にもっとも重視すべき要素は、 速度、信頼性、そして費用です。
三つを同時にバランスよく実現することは稀です。
通常、速くて信頼性のある大容量記録装置は高価であり、
費用を抑えようとすると速度または信頼性のどちらかが犠牲になります。

ここで例にあげるシステムの設計においては、
費用が最も重要な要素として、次に速度、最後に信頼性が選択されています。
このシステムでのデータ転送速度は結局のところネットワークによって制限されます。
信頼性は大変重要です。ただし、以下で説明する CCD ドライブは、 データ自体はすでに CD-R に完全にバックアップしてあるもの (したがって交換は簡単にできます)
の、オンラインデータの役割をさせています。

あなた自身の要求事項を決定することは、 大容量記録に関する解決法を選択することの最初の段階です。
もしあなたの要求事項が費用より速度または信頼性を優先するなら、
解決法はこのシステムとは違うものになるでしょう。

ハードウェアのインストール

IDE システムディスクに加えて、Western Digital 製の 30GB, 5400RPM の IDE ディスク三台を使って、以下に説明されているような約 90GB のオンラインストレージとなる CCD ディスクを作成しました。各 IDE ディスクがそれぞれの IDE コントローラとケーブルをもっていることが理想的ですが、費用を最低限にするために、 IDE コントローラを追加していません。その代わり、それぞれの IDE コントローラがマスタデバイスを一つ、 スレーブデバイスを一つ持つように、
ディスクはジャンパを使って設定されています。

再起動の際に、システム BIOS が接続されたディスクを自動的に検出するように設定されました。
より重要なことは、FreeBSD が再起動の際にそれらを検出することです。

```
ad0: 19574MB <WDC WD205BA> [39770/16/63] at ata0-master UDMA33
ad1: 29333MB <WDC WD307AA> [59598/16/63] at ata0-slave UDMA33
ad2: 29333MB <WDC WD307AA> [59598/16/63] at ata1-master UDMA33
ad3: 29333MB <WDC WD307AA> [59598/16/63] at ata1-slave UDMA33
```

FreeBSD がディスクをすべて検出しないときは、
ジャンパを正しく設定してあるか確認してください。多くの IDE ドライブは "ケーブルセレクト" ジャンパを持っています。 これはマスタ /スレーブの関係を設定するジャンパでは ありません。ドライブの文書を参照して、
正しいジャンパ設定を見つけてください。

次に、ファイルシステムの一部分として、 それらをどのように接続するのかを考慮します。vinum(8) および ccd(4) の両方を検討すべきでしょう。この設定では、ccd(4) を選択しました。

CCD の設定

ccd(4) ドライバは、いくつかの同じディスクを使って、
一つの論理的ファイルシステムに連結することができます。 ccd(4) を使用するためには、カーネルが

ccd(4) に対応している必要があります。 次の行をカーネルコンフィギュレーションファイルに追加して、カーネルを再構築し、再インストールしてください。

```
pseudo-device   ccd     4
```

5.X システムでは、 上記の代わりに次の行を追加しなければなりません。

```
device   ccd
```

FreeBSD 5.X では ccd(4) デバイスの数を指定する必要はありません。ccd(4) デバイスドライバは自己複製するようになりました - 新しいデバイスインスタンスは、必要に応じてその都度自動的に作成されます。

FreeBSD 3.0 以降では、 カーネルモジュールを読み込んで ccd(4) に対応することもできます。

ccd(4) を設定するために、まず disklabel(8) を使用してディスクにラベルを書き込まなくてはなりません。

```
disklabel -r -w ad1 auto
disklabel -r -w ad2 auto
disklabel -r -w ad3 auto
```

このコマンドはディスク全体を示す ad1c, ad2c および ad3c に対するディスクラベルを作成します。

FreeBSD 5.1-RELEASE から、従来の disklabel(8) プログラムは bsdlabel(8) ユーティリティに置き換えられました。bsdlabel(8) では、使用されていない数多くのオプションやパラメタが削除されました。 たとえば -r オプションは bsdlabel(8) では取り除かれました。詳細については bsdlabel(8) のマニュアルページを参照してください。

次に、ディスクラベルのタイプを変更します。 disklabel(8) を使用してディスクラベルを編集してください。

```
disklabel -e ad1
disklabel -e ad2
disklabel -e ad3
```

このコマンドは EDITOR 環境変数に設定されているエディタ (一般的には vi(1)) でそれぞれのディスクの現在のディスクラベルを開きます。

変更されていないディスクラベルは以下のようになります。

```
8 partitions:
#        size   offset    fstype   [fsize bsize bps/cpg]
  c: 60074784        0    unused        0     0     0   # (Cyl.    0 - 59597)
```

ccd(4) で使用する e パーティションを作成します。通常では c パーティションの行をコピーすれば良いでしょう。しかし、 fstype は 4.2BSD でなければ なりません。ディスクラベルは以下のようになるでしょう。

```
8 partitions:
#        size   offset    fstype   [fsize bsize bps/cpg]
  c: 60074784        0    unused        0     0     0   # (Cyl.    0 - 59597)
  e: 60074784        0    4.2BSD        0     0     0   # (Cyl.    0 - 59597)
```

ファイルシステムの構築

ccd0c デバイスノードはまだ存在していないかも知れません。そのときは、次のコマンドを実行して作成してください。

```
cd /dev
sh MAKEDEV ccd0
```

FreeBSD 5.0 では devfs(5) が /dev 以下のデバイスノードを自動的に管理するので、MAKEDEVを使用する必要はありません。

すべてのディスクにラベルを書き込んだので、 ccd(4) を構築してください。これを行うためには、以下のようなオプションで ccdconfig(8) を使います。

```
ccdconfig ccd0 32 0 /dev/ad1e /dev/ad2e /dev/ad3e
```

各オプションの使用法と意味は以下の通りです。 * 一番目の引数は設定するデバイスです。この例の場合は /dev/ccd0c です。 /dev/ の部分はオプションです。 * ファイルシステムに対するインタリーブです。インタリーブは、ディスクブロック内のストライプサイズを定義します。 ディスクブロックは通常 512 バイトです。したがって 32 インタリーブは 16,384 バイトとなります。 * これは ccdconfig(8) に対するフラグです。 ドライブミラーリングを有効にしたい場合、 ここにフラグを指定します。この設定では ccd(4) に対するミラーリングは提供しませんので、 0 (ゼロ) を指定しています。 * この ccdconfig(8) に対する最後の引数は、 アレイ内に置くデバイスです。それぞれのデバイスに対する完全なパス名を使用します。

ccdconfig(8) を実行すると ccd(4) が設定されます。これでファイルシステムをインストールすることが可能です。 オプションについて newfs(8) を参照するか、 次のように実行してください。

```
newfs /dev/ccd0c
```

自動的に設定する

一般的に、再起動するたびに ccd(4) をマウントしたいと思うでしょう。これを行うために、まず設定をしなければなりません。次のコマンドを用いて、 現在の設定を /etc/ccd.conf に書き出します。

```
ccdconfig -g > /etc/ccd.conf
```

/etc/ccd.conf が存在すると、再起動の際に /etc/rc スクリプトが ccdconfig -C を実行します。これにより、ccd(4) は自動的に設定された後、マウントされます。

シングルユーザモードで起動している場合には、ccd(4) を mount(8) する前に、アレイを設定するために次のコマンドを実行する必要があります。

```
ccdconfig -C
```

自動的に ccd(4) をマウントするには、/etc/fstab に ccd(4) のエントリ追加します。このように設定すると起動時にマウントされます。

```
/dev/ccd0c              /media      ufs     rw      2       2
```

Vinum ボリュームマネージャ

Vinum ボリュームマネージャは、仮想ディスクドライブを実装したブロックデバイスドライバです。Vinum は、ディスクハードウェアをブロックデバイスインタフェースから分離し、データを配置します。その結果、ディスク記憶装置を従来のスライスで扱うのと比較して、柔軟性、性能および信頼性が向上しています。vinum(8) は RAID-0, RAID-1 および RAID-5 モデル、そしてそれぞれの組合せを実装しています。

vinum(8) の詳細については Vinum ボリュームマネジャ を参照してください。

14.4.2. ハードウェア RAID

FreeBSD は、さまざまなハードウェア RAID コントローラにも対応しています。これらのデバイスはアレイを制御するための特別なソフトウェアを FreeBSD で必要することなく、RAID サブシステムを制御します。

カード上の BIOS を使用して、カードはそれ自身でディスク操作のほとんどを制御します。以下は Promise IDE RAID コントローラを使用した設定の簡単な説明です。このカードがインストールされ、システムが起動したときには、情報の入力を促すプロンプトを表示します。指示にしたがってカードの設定画面に進んでください。接続されたドライブを組み合わせるように設定することができます。設定後、ディスクは FreeBSD に対して単一のドライブのように見えます。他の RAID レベルは適宜設定できます。

14.4.3. ATA RAID1 アレイの再構築

FreeBSD はアレイ内の障害ディスクを動作中に交換できます。ただし、再起動前にそれを検知していることが必要です。

/var/log/messages または dmesg(8) の出力に次のような行があるでしょう。

```
ad6 on monster1 suffered a hard error.
```

```
ad6: READ command timeout tag=0 serv=0 - resetting
ad6: trying fallback to PIO mode
ata3: resetting devices .. done
ad6: hard error reading fsbn 1116119 of 0-7 (ad6 bn 1116119; cn 1107 tn 4 sn 11)
status=59 error=40
ar0: WARNING - mirror lost
```

atacontrol(8) を使用して詳細を調べてください。

```
# atacontrol list
ATA channel 0:
    Master:      no device present
    Slave:   acd0 <HL-DT-ST CD-ROM GCR-8520B/1.00> ATA/ATAPI rev 0

ATA channel 1:
    Master:      no device present
    Slave:       no device present

ATA channel 2:
    Master:  ad4 <MAXTOR 6L080J4/A93.0500> ATA/ATAPI rev 5
    Slave:       no device present

ATA channel 3:
    Master:  ad6 <MAXTOR 6L080J4/A93.0500> ATA/ATAPI rev 5
    Slave:       no device present

# atacontrol status ar0
ar0: ATA RAID1 subdisks: ad4 ad6 status: DEGRADED
```

1. ディスクを安全に取り外すために、 まずアレイから切り離します。

   ```
   # atacontrol detach 3
   ```

2. ディスクを取り外します。
3. スペアのディスクを取り付けます。

   ```
   # atacontrol attach 3
   Master:  ad6 <MAXTOR 6L080J4/A93.0500> ATA/ATAPI rev 5
   Slave:   no device present
   ```

4. アレイを再構築します。

   ```
   # atacontrol rebuild ar0
   ```

5. 再構築コマンドは完了するまで他の操作を受け付けません。しかし、 もう一つ別のターミナルを (Alt + Fn を押して) 開き、 次のコマンドを実行すると進行状態を確認することができます。

```
# dmesg | tail -10
[output removed]
ad6: removed from configuration
ad6: deleted from ar0 disk1
ad6: inserted into ar0 disk1 as spare

# atacontrol status ar0
ar0: ATA RAID1 subdisks: ad4 ad6 status: REBUILDING 0% completed
```

6. 操作が完了するまでお待ちください。

14.5. 光メディア (CD & DVD) の作成と使用

14.5.1. はじめに

CD は他の一般的なディスクと異なる様々な特徴を持っています。
そもそもユーザが書き込むことができません。 また遅延なしで連続的に読み出せるように、
トラック間をヘッドが移動しないですむようにデザインされています。
さらにこのサイズのメディアの中ではシステムをまたぐデータの 移動が比較的簡単でもあります。

CD はトラックの概念を持っていますが、
これはデータを連続的に読み出すためのものであってディスクの物理特性ではありません。 FreeBSD で CD を作成するには、まず CD のトラックとなるデータファイルを用意し、 そのトラックを CD に書き込みます。

ISO 9660 ファイルシステムはこの様な差異を扱うべく設計されました。
その結果、ファイルシステムは一般的に使用するのに差しつかえない程度に
制限されて標準化されています。幸いなことに、ISO 9660
ファイルシステムには拡張機構が提供されています。適切に書かれた CD は、
拡張機構に対応したシステムでは拡張を利用して、そうでないシステムでは
拡張機構を使用しない範囲で動作するようになっています。

sysutils/mkisofs プログラムは ISO 9660
ファイルシステムを含むデータファイルを作成するのに使われます。
これには様々な拡張をサポートするオプションがあり、 以下で説明します。 このソフトウェアは、ports の sysutils/mkisofs からインストールすることができます。

CD に書き込むためのツールは、お使いの CD ライタが ATAPI 接続か否かにも依存します。ATAPI CD ライタなら、ベースシステムの一部である burncd プログラムを使います。SCSI や USB の CD ライタなら、ports の sysutils/cdrecord をインストールして cdrecord プログラムを使うべきでしょう。

burncd が対応しているドライブは限定されています。
ドライブが対応されているかどうかを確認するには、 CD-R/RW supported drives
にある一覧を見てください。

FreeBSD 5.X または FreeBSD 4.8-RELEASE 以降のバージョンを使用している場合、ATAPI/CAM モジュール を使用すると ATAPI ハードウェア上で SCSI ドライブ用の cdrecord および他のツールを使用できるようになります。

14.5.2. mkisofs

sysutils/mkisofs は UNIX® ファイルシステムの名前空間におけるディレクトリツリーのイメージとして ISO 9660 ファイルシステムを作成します。 最も簡単な使い方は以下の通りです。

```
# mkisofs -o imagefile.iso /path/to/tree
```

このコマンドは */path/to/tree* 以下のディレクトリツリーのコピーである ISO 9660 ファイルシステムを含んだ *imagefile.iso* ファイルを作成します。この過程において、ファイル名は標準的な ISO 9660 ファイルシステムの制限に適合するようなファイル名に対応づけられ、ISO ファイルシステムでファイル名を文字化できないファイルは除外されます。

この制限を回避するために利用できるオプションはいくつもあります。 特に -R オプションは UNIX® システムで標準的な Rock Ridge 拡張を有効にします。-J オプションは Microsoft のシステムで標準的な Joliet 拡張を有効にし、 -hfs オプションは Mac OS® で使用されている HFS ファイルシステムを作成するために使われます。

FreeBSD でしか使わないのであれば、-U オプションを使用するとあらゆるファイル名制限を無効にできます。 さらに -R オプションとともに使うことで FreeBSD と同一のファイルシステムイメージを作成できますが、 これは ISO 9660 標準の多くを無視しています。

一般的に使われる最後のオプションは -b オプションです。 これは "El Torito" ブータブル CD を作成するのに使う起動イメージのありかを指定します。
このオプションは引数として起動イメージへのパスを、CD に書き込まれるディレクトリツリーの頂点からの相対位置で取ります。 したがって /tmp/myboot がブート可能な FreeBSD システムで /tmp/myboot/boot/cdboot にブートイメージがあるならば、以下のようにすることで ISO 9660 ファイルシステムのイメージを /tmp/bootable.iso に作成することができます。

```
# mkisofs -U -R -b boot/cdboot -o /tmp/bootable.iso /tmp/myboot
```

この後、カーネルで vn (FreeBSD 4.X) または md (FreeBSD 5.X) が設定されていれば、ファイルシステムを以下のようにしてマウントすることができます。

```
# vnconfig -e vn0c /tmp/bootable.iso
# mount -t cd9660 /dev/vn0c /mnt
```

FreeBSD 4.X および FreeBSD 5.X に対しては以下の通りです。

```
# mdconfig -a -t vnode -f /tmp/bootable.iso -u 0
```

```
# mount -t cd9660 /dev/md0 /mnt
```

/mnt と /tmp/myboot が同一かどうか確認してください。

sysutils/mkisofs には挙動を細かく制御するために他にもたくさんのオプションがあります。 特に、ISO 9660 レイアウトの変更や Joliet および HFS ディスク作成などの 詳細は mkisofs(8) のマニュアルページをご覧ください。

14.5.3. burncd

あなたが持っているのが ATAPI CD ライタなら、CD に ISO イメージを書き込むために burncd コマンドが使えます。 burncd はベースシステムの一部で /usr/sbin/burncd としてインストールされています。 使い方はとても単純でオプションも少ししかありません。

```
# burncd -f cddevice data imagefile.iso fixate
```

以上のコマンドは *imagefile.iso* のコピーを *cddevice* に書き込みます。 デフォルトのデバイスは /dev/acd0c です。 書き込み速度や操作完了後に CD を自動的に取り出す方法、オーディオデータの書き込みなどのオプションについては burncd(8) を見てください。

14.5.4. cdrecord

あなたが持っている CD ライタが ATAPI ではなければ、 CD を書き込むのに cdrecord を使う必要があります。 cdrecord はベースシステムの一部ではなく、 sysutils/cdrtools の port または適切な package を利用してインストールしなければなりません。 なお、ベースシステムを変更するとバイナリに矛盾が発生し、 "コースター" を作ってしまうおそれがあります。 したがって、システムをアップグレードする度にこの port も作り直すか、 あるいは FreeBSD の安定版を追いかけているのならば、新しいバージョンが利用できるようになった時に ports をアップグレードする必要があります。

cdrecord にはたくさんのオプションがありますが、 基本的な使い方は burncd よりもさらに簡単です。 ISO 9660 イメージを書き込むには以下のようにします。

```
# cdrecord dev=device imagefile.iso
```

cdrecord のトリッキーな部分は、使用する dev を見つけるところにあります。 適切な設定を見つけるためには cdrecord の -scanbus フラグを使います。 たとえば、以下のような結果が出力されるでしょう。

```
# cdrecord -scanbus
Cdrecord 1.9 (i386-unknown-freebsd4.2) Copyright (C) 1995-2000 Jörg Schilling
Using libscg version 'schily-0.1'
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) *
```

リストにあるデバイスに対する適切な dev の値がここに示されています。あなたの CD ライタをこのリストから見つけ、カンマで区切られた 3 つの数値を dev の値として使ってください。この例では CRW デバイスは 1,5,0 なので、適切な入力は dev=1,5,0 となります。値を明示するもっと簡単な方法もあります。詳細は cdrecord(1) を見てください。そこにはオーディオトラックを書き込む方法や、
書き込み速度その他を操作する方法も書かれています。

14.5.5. オーディオ CD の複製

CD からオーディオデータを連続したファイルに展開し、ブランク CD にこれらのファイルを書き込むことで、オーディオ CD を複製することができます。 この手順は ATAPI および SCSI ドライブの間で少し異なります。

Procedure: SCSI ドライブ

1. cdda2wav を使用してオーディオを展開します。

```
% cdda2wav -v255 -D2,0 -B -Owav
```

2. cdrecord を使用して .wav ファイルに書き出します。

```
% cdrecord -v dev=2,0 -dao -useinfo  *.wav
```

cdrecord に説明されているように *2.0* が適切に指定されていることを確かめてください。

Procedure: ATAPI ドライブ

1. ATAPI CD ドライバでは、それぞれのトラックを /dev/acddtnn のように利用できます。 ここで *d* はドライブ番号であり、 *nn* は二桁十進のトラック番号です。 一桁の場合 0 を前に付加する必要があります。 したがって、一番目のディスクの一番目のトラックは /dev/acd0t01、二番目のトラックは /dev/acd0t02、三番目のトラックは /dev/acd0t03

などとなります。

適切なデバイスファイルが /dev に存在することを確かめてください。
存在しなければ、たとえば次のようにして作成します。

```
# cd /dev
# sh MAKEDEV acd0t99
```

FreeBSD 5.0 では devfs(5) が /dev にエントリを自動的に作成、管理するので、MAKEDEV を使用する必要はありません。

2. dd(1) を使用して各トラックを展開します。
ファイルを展開する際、ブロックサイズを指定しなければなりません。

```
# dd if=/dev/acd0t01 of=track1.cdr bs=2352
# dd if=/dev/acd0t02 of=track2.cdr bs=2352
...
```

3. burncd を使用して、展開したファイルをディスクに書き込みます。
これらがオーディオファイルであること、そして書き込みが終了したときに burncd がディスクを固定 (fixate) することを明示しなければなりません。

```
# burncd -f /dev/acd0c audio track1.cdr track2.cdr ... fixate
```

14.5.6. データ CD の複製

データ CD を、sysutils/mkisofs を用いて作成されたイメージファイルと機能的に等価なイメージファイルにコピーできます。
これを使用して、すべてのデータ CD を複製することができます。 ここでの例は CDROM デバイスが acd0 であるとしています。あなたの CDROM デバイスに読み替えてください。 CDROM の場合には、パーティション全体またはディスク全体を指定するために c をデバイス名の後に追加しなければなりません。

```
# dd if=/dev/acd0c of=file.iso bs=2048
```

これでディスクイメージを取り出すことができました。すでに説明した方法を用いて CD に書き込むことができます。

14.5.7. データ CD の使用

さて、標準的なデータ CDROM を作成したので、おそらく次はそれをマウントしてデータを読み出したいと思うでしょう。 デフォルトでは mount(8) は、ファイルシステムタイプを ufs としています。 次のように実行しようとすると、

```
# mount /dev/cd0c /mnt
```

Incorrect super block というエラーが返されてマウントできないでしょう。 CDROM は UFS ファイルシステムではないために、 このような手順でマウントしようすると失敗します。 ファイルシステムのタイプが ISO9660 であると mount(8) に教えさえすれば、すべてはうまく動作します。 mount(8) に -t cd9660 オプションを指定することでこれを行います。 たとえば /dev/cd0c の CDROM デバイスを /mnt にマウントしたい場合は、 以下のように実行します。

```
# mount -t cd9660 /dev/cd0c /mnt
```

使用している CDROM インタフェースによっては、 デバイス名 (この例では /dev/cd0c) が異なるかもしれないことに注意してください。 また、-t cd9660 オプションは、単に mount_cd9660(8) を実行します。 この例を以下のように短縮することもできます。

```
# mount_cd9660 /dev/cd0c /mnt
```

一般的にこの方法では、すべてのメーカの データ CDROM を使用することができます。しかしながら、特定の ISO 9660 拡張が施されたディスクでは奇妙な動作をするかもしれません。 たとえば Joliet ディスクは、すべてのファイル名を 2 バイトの Unicode 文字で格納します。 FreeBSD カーネルは (まだ) Unicode を理解できないので、 非英語文字はクエスチョンマークで表示されます (FreeBSD 4.3 以降を使用している場合、CD9660 ドライバには適切な Unicode 変換表を読み込むための急ごしらえのフックが含まれています。 いくつかの共通のエンコードに対するモジュールは sysutils/cd9660_unicode port から利用可能です)。

CDROM をマウントしようとする時に、 Device not configured と表示されるかもしれません。これは、ディスクがトレーにないと CDROM ドライブが判断しているか、ドライブがバス上に認識できないことを通常意味します。 ディスクが挿入されたことを CDROM ドライブが認識するには数秒かかりますので、 辛抱強く待ってください。

バスのリセットに返答するためのタイムアウトが短いために、時々 SCSI CDROM は認識に失敗するかもしれません。SCSI CDROM を持っている場合は、次のオプションをカーネルコンフィギュレーションファイルに追加して、 カーネルを再構築してください。

```
options SCSI_DELAY=15000
```

これより、SCSI バスを起動時に 15 秒間停止させて、 CDROM ドライブがバスリセットに応答する機会を与えます。

14.5.8. Raw データ CD の書き込み

ISO 9660 ファイルシステムを作成すること無く、 ファイルを直接 CD に書き込むこともできます。 この方法をバックアップ目的に使用している人もいます。 これは、標準 CD

を書き込むよりもさらに速く実行することができます。

```
# burncd -f /dev/acd1c -s 12 data archive.tar.gz fixate
```

このように CD に書き込まれたデータを取得するには、raw デバイスノードからデータを読み込まなくてはなりません。

```
# tar xzvf /dev/acd1c
```

このディスクを通常の CDROM としてマウントすることはできません。 このような CDROM は FreeBSD を除いて、他のすべてのオペレーティングシステムでは読み込むことはできません。CD をマウントしたいか、その他のオペレーティングシステムとデータを共有したい場合は、上記に説明したように sysutils/mkisofs を使用しなくてはなりません。

14.5.9. ATAPI/CAM ドライバの使用

このドライバは、ATAPI デバイス (CD-ROM, CD-RW, DVD ドライブなど) へ SCSI サブシステムを通じてアクセスすることを可能にします。これにより、sysutils/cdrdao または cdrecord(1) のようなアプリケーションが使用できるようになります。

このドライバを使用するためには、
カーネルコンフィギュレーションファイルに次の行を追加する必要があります。

```
device atapicam
device scbus
device cd
device pass
```

次の行もカーネルコンフィギュレーションファイルに必要です。

```
device ata
device atapicd
```

両方がすでに存在しなければなりません。

それから再構築し、新しいカーネルをインストールし、コンピュータを再起動します。
起動プロセス中にディスクライタは以下のように表示されるでしょう。

```
acd0: CD-RW <MATSHITA CD-RW/DVD-ROM UJDA740> at ata1-master PIO4
cd0 at ata1 bus 0 target 0 lun 0
cd0: <MATSHITA CDRW/DVD UJDA740 1.00> Removable CD-ROM SCSI-0 device
cd0: 16.000MB/s transfers
cd0: Attempt to query device size failed: NOT READY, Medium not present - tray closed
```

ドライブは /dev/cd0 デバイスを通じてアクセスすることが可能となります。 たとえば、次のようにして CD-ROM を /mnt にマウントします。

```
# mount -t cd9660 /dev/cd0c /mnt
```

root 権限で次のコマンドを実行して、 ライタの SCSI アドレスを得ることができます。

```
# camcontrol devlist
<MATSHITA CDRW/DVD UJDA740 1.00>   at scbus1 target 0 lun 0 (pass0,cd0)
```

したがって、1,0,0 が cdrecord(1) およびその他の SCSI アプリケーションで使用する SCSI アドレスです。

ATAPI/CAM および SCSI システムの詳細は atapicam(4) および cam(4) マニュアルページを参照してください。

14.6. フロッピーディスクの作成と使用

フロッピーディスクにデータを格納することはしばしば役にたちます。
たとえば、ある人が他のリムーバブル記録メディアを何も持っていないときや、
小さなデータを他のコンピュータに移動させる必要があるときです。

この節では、FreeBSD におけるフロッピーディスクの使用方法を説明します。 主に 3.5 インチの DOS フロッピーのフォーマットと操作方法を扱いますが、
他のフロッピーディスクの形式についても概念は似ています。

14.6.1. フロッピーのフォーマット

デバイス

他のデバイスと同様に、フロッピーディスクは /dev にあるエントリを通じてアクセスされます。4.X およびそれ以前のリリースにおいて raw フロッピーディスクにアクセスするには /dev/fdN または /dev/fdNX を使用します。*N* はドライブ番号を表し、 大抵は 0 です。*X* は文字を表します。

5.0 およびそれ以降のリリースでは、単に /dev/fdN を使用します。

4.X およびそれ以前のリリースでのディスクサイズ

/dev/fdN.size というデバイスもあります。 *size* はフロッピーディスクのサイズをキロバイトで示したものです。
これらのエントリは低レベルフォーマットの際に、 ディスクサイズを決定するのに使用されます。
1440kB は以下の例で使用されるサイズです。

時々 /dev 下のエントリは (再) 作成されなければなりません。次のコマンドでこれを行います。

```
# cd /dev && ./MAKEDEV "fd*"
```

5.X およびそれ以降のリリースでのディスクサイズ

FreeBSD 5.0 では devfs(5) が /dev 内のエントリを自動的に管理するので、MAKEDEVを使用する必要はありません。

所望のディスクサイズは fdformat(1) に -f フラグを通して渡されます。対応しているサイズは fdcontrol(8) のマニュアルページに掲載されていますが、最良に動作するのは 1440kB だと助言しておきます。

フォーマット

フロッピーディスクは、使用前に低レベルフォーマットをする必要があります。
通常、ベンダは低レベルフォーマット済みのディスクを出荷していますが、
フォーマットはメディアの品質を確認するよい方法です。より大きな (または小さな)
ディスクサイズにすることも可能ですが、ほとんどのフロッピーディスクのサイズは 1440kB
で動作するように設計されています。

フロッピーディスクを低レベルフォーマットするには fdformat(1) を使用する必要があります。
このユーティリティは引数としてデバイス名を指定します。

ディスクが良好かあるいは不良であるかを決定するのに役立つので、
エラーメッセージをすべてメモに取っておいてください。

4.X 以前のリリースでのフォーマット

/dev/fdN.size デバイスを使ってフロッピーをフォーマットします。新しい 3.5
インチフロッピーディスクをドライブに挿入し、以下のコマンドを実行してください。

```
# /usr/sbin/fdformat /dev/fd0.1440
```

5.0 以降のリリースでのフォーマット

/dev/fdN デバイスを使用してフロッピーをフォーマットします。新しい 3.5
インチフロッピーディスクをドライブに挿入し、以下のコマンドを実行してください。

```
# /usr/sbin/fdformat -f 1440 /dev/fd0
```

14.6.2. ディスクラベル

ディスクを低レベルフォーマットしたら、次にディスクラベルを作成する必要があります。
ディスクラベルは後で破棄されますが、
システムがディスクのサイズとジオメトリを決定するのに必要になります。

新しいディスクラベルはディスク全体を引き継ぎ、
フロッピーのジオメトリに関する適切な情報のすべてが含まれます。
ディスクラベルに対するジオメトリの値は /etc/disktab に掲載されています。

次のように disklabel(8) を実行できます。

```
# /sbin/disklabel -B -r -w /dev/fd0 fd1440
```

FreeBSD 5.1-RELEASE から、従来の disklabel(8) プログラムは bsdlabel(8)
ユーティリティに置き換えられました。bsdlabel(8) では、

> 使用されていないオプションおよびパラメタの数多くが削除されました。 たとえば -r オプションは bsdlabel(8) では取り除かれました。詳細については bsdlabel(8) マニュアルページを参照してください。

14.6.3. ファイルシステム

これでフロッピーを高レベルフォーマットする準備ができました。これは FreeBSD がディスクを読み書きする新しいファイルシステムを作成します。
新しいファイルシステムを作成するとディスクラベルは破棄されます。
したがって、ディスクを再フォーマットするときには、 ディスクラベルを再作成しなくてはなりません。

フロッピーのファイルシステムには UFS または FAT を使用できます。 フロッピーに対しては FAT が一般的によりよい選択です。

フロッピー上に新しいファイルシステムを作成するには次のようにします。

```
# /sbin/newfs_msdos /dev/fd0
```

これでディスクが使用できるようになりました。

14.6.4. フロッピーの使用

フロッピーを使用するために、mount_msdos(8) (4.X 以前のリリース) または mount_msdosfs(8) (5.0 以後のリリース) を用いてマウントします。 Ports Collection から emulators/mtools を使用することもできます。

14.7. データテープの作成と使用

一般的なテープメディアには 4mm, 8mm, QIC, ミニカートリッジ、 DLT があります。

14.7.1. 4mm (DDS: Digital Data Storage)

4mm テープはワークステーションのバックアップメディアとして QIC に取って代わりつつあります。この傾向は QIC ドライブの主要なメーカであった Archive を Conner が買収し QIC ドライブの製造を中止したことで加速しました。 4mm ドライブは小型で静かですが 8mm ドライブが持っている信頼性ほど、その評判は良くありません。 また、4mm カートリッジは 8mm カートリッジよりも安価で小型 (3 x 2 x 0.5 インチ、76 x 51 x 12 mm) になっています。 ただし、8mm と同様に、4mm のヘッドはヘリカルスキャン方式 (訳注: VTR と同様の回転ヘッドを使う方式) を採用しているため、比較的寿命が短いです。

ドライブのデータスループットは、150 kB/s から 最大で 500 kB/s 程度です。 データ容量は 1.3 GB から 2.0 GB です。 ドライブのほとんどで利用可能なハードウェア圧縮を使用すると、 容量が約 2 倍になります。 マルチドライブテープライブラリユニットは 1 つの筐体に 6 つのドライブを収容可能で、自動的にテープの交換ができます。 ライブラリの容量は 240 GB に達します。

現在の DDS-3 標準は 12 GB (圧縮時 24 GB) までのテープ容量に対応しています。

8mm ドライブと同様に 4mm ドライブはヘリカルスキャンを使用します。 ヘリカルスキャン方式の利点および欠点はすべて 4mm および 8mm ドライブの両方に当てはまります。

テープは 2,000 回のパスあるいは 100 回フルバックアップした後には交換するべきです。

14.7.2. 8mm (Exabyte)

8mm テープは SCSI テープドライブとして最もよく使われているもので、データ交換用として最良の選択です。ほとんどのサイトには Exabyte 2 GB 8mm テープドライブがあるでしょう。8mm ドライブは信頼性が高く、使いやすく、静かです。カートリッジは安価で小型です (4.8 x 3.3 x 0.6 インチ、122 x 84 x 15 mm)。8mm テープの欠点は、テープとヘッドの相対的な速度が高速なために、
比較的ヘッドとテープの寿命が短いことです。

データスループットは 250 kB/s から 500 kB/s 程度です。データ容量は 300 MB から 7 GB です。ほとんどのドライブで利用可能なハードウェア圧縮を利用すると、 容量が約 2 倍になります。これらのドライブは、単一のユニットから 6 つのドライブと 120 本のテープを一つの筐体に収容したマルチドライブテープライブラリまで利用可能です。
テープはユニットによって自動的に取り換えられます。 ライブラリの容量は 840 GB 以上に達します。

Exabyte の "Mammoth" モデルはテープ 1 本あたり 12 GB (圧縮時 24 GB) に対応し、従来のテープドライブと比べ費用は約 2 倍になります。

データはヘリカルスキャンを用いてテープに記録されます。 ヘッダはメディアに対してある傾き (約 6 度) に配置されます。 テープはヘッドのある円筒の周の 270 度にわたって接触します。テープが円筒面を走行する間、円筒は回転しています。
この結果、高密度のデータのつまったトラックは、
狭い間隔でテープの上端と下端の間を斜めに横切ります。

14.7.3. QIC

QIC-150 テープとドライブは、 おそらく最も一般的に使われているドライブとメディアでしょう。 QIC テープドライブは "現実的な" バックアップドライブとしては最も高価でないものです。欠点はメディアのコストです。QIC テープは 8mm や 4mm テープと比較して GB あたりのデータの保存で 5 倍ほど高価です。しかし、あなたの必要とする量が半ダース程のテープで十分であれば、 QIC は正しい選択かもしれません。QIC は 最も一般的なテープドライブです。 すべてのサイトに QIC ドライブのどれかの容量のものがあります。問題は、 QIC は同じようなテープ (まったく同じ場合もある) に多様な記録密度があることです。QIC ドライブは静かではありません。これらのドライブはデータ記録を開始する前に音をたててシークしますし、
リード、ライト、シークの時にはっきりと聞こえる音を出します。 QIC テープの大きさは (6 x 4 x 0.7 インチ、152 x 102 x 17 mm) です。 1/4 インチ幅のテープも使用している ミニカートリッジ は別に議論します。テープライブラリやチェンジャはありません。

データスループットは ~1500 kB/s から ~5000 kB/s 程度です。データ容量は 400 MB から 150 GB です。ハードウェア圧縮が最近のドライブの多くで利用できます。 QIC ドライブは DAT ドライブに置き換えられつつあり、 あまり頻繁には使用されなくなっています。

データは複数のトラックに分かれてテープに記録されます。
トラックはテープメディアの長さ方向の一端からもう一方の端までです (訳注: 1 トラックの read/write が終わるとテープの走行方向を反転させ 次のトラックの read/write を行います)。トラックの数と、それに対応するトラックの幅はテープの容量によって変わります。 すべてではありませんが、最近のドライブはほとんど、少なくとも読み出しについては (場合によっては書き込みも)

下位互換性があります。 QIC はデータの安全性についてはよいといわれています (ヘリカルスキャンドライブに比べて機構は単純でより丈夫です)。

テープは 5000 回のバックアップで寿命となるでしょう。

14.7.4. XXX* ミニカートリッジ

14.7.5. DLT

DLT はここに示したドライブのタイプの中で最高速のデータ転送レートを発揮します。 1/2 インチ (12.5mm) テープが単リールのカートリッジ (4 x 4 x 1 インチ、100 x 100 x 25 mm) に入っています。
カートリッジのひとつの側面全体がスイングゲートになっています。
ドライブの機構がこのゲートを開け、テープリーダを引き出します。
テープリーダには楕円形の穴があり、 ドライブがテープを "引っ掛ける" のに使います。
巻き取りのためのリールはドライブの中にあります。 ここに挙げた他のカートリッジはすべて (9 トラックテープは唯一の例外です)
送り出しリールと巻き取りリールの両方がカートリッジの中にあります。

データスループットは約 1.5 MB/s で、4mm, 8mm, QIC テープドライブの 3 倍です。データ容量は単一のドライブで 10 GB から 20 GB の範囲です。マルチテープチェンジャ、マルチテープドライブ、5 から 900 巻のテープを 1 から 20 ドライブで扱うマルチドライブテープライブラリがあり、 50 GB から 9 TB の容量が得られます。

圧縮によって、DLT Type IV フォーマットは 70 GB までの容量に対応しています。

データは (QICテープのように) テープの走行方向と平行に複数あるトラックへ記録されます。 2 つのトラックに同時書き込みを行います。 read/write ヘッドの寿命は比較的長いと言えます。
テープの走行が止まればヘッドとテープの間の相対運動は無いからです。

14.7.6. AIT

AIT は、Sony が発表した新しいフォーマットで、 テープ 1 本あたり 50 GB (圧縮時) まで格納できます。
テープにはメモリチップが搭載されており、 テープの内容の索引情報を保持しています。
他のテープではテープ上のファイルの位置を把握するのに数分必要とするのですが、
このテープドライブでは索引情報を読んで直ちに決定することができます。 SAMS:Alexandria のようなソフトウェアは、40 を超える ATI テープライブラリを操作できるのはもちろんのこと、
テープのメモリチップと直接通信して、スクリーンに内容を表示し、
どのファイルがどのテープにバックアップされたかを調べて、 正しいテープを見つけ、読み込み、
テープからデータを復元することができます。

このようなライブラリは大体 $20,000 くらいするので、
愛好家が購入できる価格帯からは外れてしまいます。

14.7.7. 新品のテープを初めて使う場合

全く新品の空テープを読もうとしたり書き込もうとすると、 処理は失敗するでしょう。
次のようなメッセージがコンソールに出力されるでしょう。

```
sa0(ncr1:4:0): NOT READY asc:4,1
```

```
sa0(ncr1:4:0):  Logical unit is in process of becoming ready
```

テープに識別ブロック (Identifier Block:block number 0) がありません。QIC-525 標準を採用したすべての QIC テープドライブは識別ブロックをテープに書き込みます。 2 つの解決方法があります。

- `mt fsf 1` によりテープドライブはテープに識別ブロックを書き込みます。
- フロントパネルのボタンを押してテープを取り出します。

  再びテープを挿入し、データをテープに `dump` します。

  `dump` は `DUMP: End of tape detected` と報告し、 コンソールには `HARDWARE FAILURE info:280 asc:80,96` と表示されるでしょう。

  `mt rewind` を使ってテープを巻戻します。

  次からはテープの操作はうまくいくでしょう。

14.8. フロッピーディスクへのバックアップ

14.8.1. データをバックアップするのにフロッピーは使えますか?

フロッピーディスクは以下の理由によって、
実際にバックアップをつくるための適切なメディアではありません。

- メディアの信頼性が (特に長期間の場合) 低い。
- バックアップとリストアがとても遅い。
- 容量が非常に小さい (1 ダースかそこらのフロッピーディスクに ハードディスク全体をバックアップしていた時代は、 はるか遠くに過ぎ去りました)。

しかしながら、データをバックアップする他の手段がないのなら、
バックアップを取らないよりもフロッピーディスクを使う方がましでしょう。

フロッピーディスクを使用せざるを得ないときは、 品質のよいディスクを使用してください。
事務所のその辺に数年転がっていたフロッピーは使わない方が良いでしょう。
評判のよいメーカの新しいディスクを使用することが理想です。

14.8.2. それではどうやってデータをフロッピーにバックアップするのですか?

フロッピーにバックアップする最もよい方法は、 `-M` (マルチボリューム) オプション付きで tar(1) コマンドを使用することです。これで、
複数のフロッピーにわたってバックアップすることが可能になります。

カレントディレクトリとサブディレクトリ内のすべてのファイルをバックアップするには、
以下のコマンドを (`root` 権限で) 使用します。

```
# tar Mcvf /dev/fd0 *
```

1 枚目のフロッピーが一杯になると、 tar(1) は次のボリュームを挿入するように要求します (tar(1)

はさまざまなメディアを扱えるので、
ボリュームと表示します。この文脈ではフロッピーディスクのことです)。

```
Prepare volume 2 for /dev/fd0 and hit return:
```

指定したファイルがすべて保存されるまで (ボリューム番号を増やしながら) これが繰り返されます。

14.8.3. バックアップを圧縮することはできませんか?

残念なことに tar(1) はマルチボリュームアーカイブに対して、 -z オプションを使うことができません。もちろん、すべてのファイルを gzip(1) で圧縮し、 それらを tar(1) を用いてフロッピーに保存して、それから再び gunzip(1) することはできます。

14.8.4. どのようにしてバックアップをリストアしたらいいのでしょうか?

すべてのアーカイブをリストアするには以下のようにします。

```
# tar Mxvf /dev/fd0
```

特定のファイルだけをリストアするには 2 つの方法があります。 1 つ目は、1 枚目のフロッピーを用いて以下のようにするものです。

```
# tar Mxvf /dev/fd0 filename
```

tar(1) ユーティリティは、 必要なファイルを見つけるまで次のディスクを挿入するように要求します。

もう 1 つは、 必要なファイルがどのフロッピーに保存されているか分かっている場合、そのフロッピーを挿入して上記と同じコマンドを使用するだけでもよいです。 あるフロッピー上にある 1 番目のファイルが、 その前のフロッピーから続いている場合は、そのファイルのリストアを要求していなくても tar(1) はそれをリストアできないと警告することに注意してください!

14.9. バックアップの基本

主なバックアッププログラムは dump(8), tar(1), cpio(1) の三つです。

14.9.1. ダンプとリストア

伝統的な UNIX® のバックアッププログラムは dump と restore です。これらはファイルシステムによって作成されるファイル、リンク、
ディレクトリといった抽象の下位にある、 ディスクブロックの集合としてドライブを操作します。 dump はデバイス上のファイルシステム全体をバックアップします。 ファイルシステムの一部分だけ、または二つ以上のファイルシステムにわたるディレクトリツリーをバックアップすることはできません。 dump はファイルおよびディレクトリをテープに書き込まずに、 ファイルおよびディレクトリを含んだ raw データブロックを書き込みます。

ルートディレクトリで dump を使った場合、/home, /usr など、他の多くのディレクトリはバックアップされません。
これらのディレクトリは通常、
他のファイルシステムへのマウントポイントであったり、
シンボリックリンクとなっているためです。

dump には AT&T UNIX のバージョン 6 (およそ 1975 年) の初期から残っている癖があります。デフォルトのパラメタは、現在利用可能な高密度メディア (最大 62,182 ftpi) ではなく、9 トラックテープ (6250 bpi) に最適な値となっています。現在のテープドライブの容量を利用するために、これらのデフォルト値をコマンドラインで上書きしなければなりません。

rdump と rrestore を用いて他のコンピュータに接続されているテープドライブにネットワーク経由でデータをバックアップすることも可能です。どちらのプログラムもリモートのテープドライブにアクセスするために rcmd および ruserok に依存しています。したがって、バックアップを実行するユーザがリモートコンピュータの .rhosts ファイルに書かれていなければなりません。rdump および rrestore の引数はリモートコンピュータに適切なものを用いなければなりません。FreeBSD コンピュータから komodo と呼ばれる Sun に接続されている Exabyte テープへ rdump するには以下のようにします。

```
# /sbin/rdump 0dsbfu 54000 13000 126 komodo:/dev/nsa8 /dev/da0a 2>&1
```

注意: .rhosts 認証を許可することには、セキュリティに関する暗黙の仮定があります。あなたの置かれている状況を注意深く調べてください。

ssh 越しに dump と restore をより安全な形で使うこともできます。

例 *28. ssh* 越しの dump の利用

```
# /sbin/dump -0uan -f - /usr | gzip -2 | ssh1 -c blowfish \
          targetuser@targetmachine.example.com dd of=/mybigfiles/dump-usr-l0.gz
```

または、環境変数 RSH を設定して、 dump の組み込み機能を利用する。

例 *29.* RSH を設定した *ssh* 越しの dump を利用

```
# RSH=/usr/bin/ssh /sbin/dump -0uan -f
targetuser@targetmachine.example.com:/dev/sa0
```

14.9.2. tar

tar(1) は AT&T UNIX の バージョン 6 (1975 年ごろ) にまで遡ることができます。tar はファイルシステムと協調して動作し、ファイルとディレクトリをテープに書き込みます。tar は cpio(1) で使用可能なフルレンジのオプションには対応していませんが、tar には cpio が使用するような奇妙なコマンドパイプラインは必要ありません。

tar の多くの版はネットワーク経由のバックアップには対応していません。 FreeBSD が使用している GNU 版の tar は、 rdump と同じ構文でリモートデバイスに対応しています。 komodo と呼ばれる Sun に接続された Exabyte テープドライブに対して tar を実行するには以下のようにします。

```
# /usr/bin/tar cf komodo:/dev/nsa8 . 2>&1
```

リモートデバイスに対応していない版に対しては、パイプラインと rsh を使用してリモートテープドライブにデータを送ることができます。

```
# tar cf - . | rsh hostname dd of=tape-device obs=20b
```

ネットワークを越えたバックアップのセキュリティを懸念しているなら、 rsh の代わりに ssh を使うべきです。

14.9.3. cpio

cpio(1) は本来 UNIX® ファイルを磁気メディアで交換するためのプログラムです。 cpio はバイトスワッピング、 多くの異なるアーカイブフォーマットの書き込みオプションがあり (それ以外にも多数のオプションがあります)、 パイプで他のプログラムにデータを渡すこともできます。 この最後にあげた特徴が、cpio をインストールメディアとしては優れた選択肢にしています。 cpio はディレクトリツリーの探索の機能はなく、ファイルリストは stdin からの入力でなくてはなりません。

cpio はネットワーク経由のバックアップには対応していません。 以下のようにパイプラインと rsh を用いてリモートテープドライブにデータを送ることができます。

```
# for f in directory_list; do
find $f >> backup.list
done
# cpio -v -o --format=newc < backup.list | ssh user@host "cat > backup_device"
```

directory_list はバックアップしたいディレクトリのリストで、 *user@host* はバックアップを実行したいユーザとホスト名の組であり、 *backup_device* はバックアップを書き込みたいデバイスです (たとえば /dev/nsa0)。

14.9.4. pax

pax(1) は tar と cpio に対する IEEE/POSIX® の回答です。長年の間、さまざまな版の tar と cpio は互いにわずかに非互換になってきていました。 それらをしらみ潰しに標準化する代わりに、POSIX® は新しいアーカイブユーティリティを作りました。 pax は、いくつもの cpio や tar のフォーマットの読み書きに対応しようと試みているほか、 専用に新しいフォーマットを開発しました。 コマンド群は tar よりも cpio の方にいくぶん似ています。

14.9.5. Amanda

Amanda (Advanced Maryland Network Disk Archiver) は単一のプログラムではなく、 クライアント/サーバ型のバックアップシステムです。 Amanda サーバは、 Amanda クライアントを有するネットワークに接続されたコンピュータからデータを受け取り、

備え付けられたテープドライブにバックアップします。
いくつもの大容量ディスクを備えたサイトでの共通の問題は、
データディレクトリをテープにバックアップするのに時間がかかりすぎることです。 Amanda はこの問題を解決します。 Amanda は "ホールディングディスク" を使用して、同時に複数のファイルシステムのバックアップを行うことができます。 Amanda の設定ファイルにかかれたすべてのファイルシステムのフルバックアップを特定の間隔でとるために "アーカイブセット" と呼ばれるテープグループを作成します。 "アーカイブセット" には夜間に作成されるすべてのファイルシステムの増分 (または差分) のバックアップも含まれます。
障害が起きたファイルシステムのリストアには、
最も新しいフルバックアップと増分のバックアップが必要です。

設定ファイルでは、バックアップの制御と Amanda によるネットワークトラフィック量を設定します。
Amanda は上記のバックアッププログラムのいずれかを使ってデータをテープに書き込みます。
Amanda は port または package として利用可能です。デフォルトではインストールされていません。

14.9.6. 何もしない

"何もしない" というのはコンピュータのプログラムではありませんが、バックアップの戦略として最も広く採用されています。 これには初期投資が必要ありません。従わなければならないバックアップスケジュールもありません。
ただ何もしないだけです。データに何か起きたら苦笑いして耐えてください!

あなたにとって時間やデータの価値が少ないか、 あるいはまったくないのであれば "何もしない" のはあなたのコンピュータに最も適したバックアッププログラムでしょう。
しかし注意してください。UNIX® は便利なツールです。 6 ヶ月も使用していれば、あなたにとって価値のあるファイルの山が出来上がっているでしょう。

"何もしない" ことはコンピュータが同じものをもう一度作り直すことのできる /usr/obj やその他のディレクトリツリーについては適切なバックアップ方法です。
一例として、このハンドブックの HTML 版 または PostScript® 版を構成するファイルがあります。これらの文書形式は SGML ファイルから作成されたものです。 HTML または PostScript® ファイルのバックアップは必要ありません。 SGML ファイルは定期的にバックアップされています。

14.9.7. どのバックアッププログラムが最適ですか?

dump(8) です。以上。 Elizabeth D. Zwicky はここで検討したプログラムすべてについて拷問的なテストを行いました。 すべてのデータと UNIX® ファイルシステムの状態すべてを保存するのに最適なのは、明らかに dump です。 Elizabeth は多種多様の特異な状態 (いくつかはあまり珍しくないものもあります) を含むファイルシステムを作成し、 それらのファイルシステムのバックアップとリストアを行って、それぞれのプログラムのテストを行いました。特異な状態とは、
ホールがあるファイル、ホールとヌルブロックがあるファイル、
奇妙な文字をファイル名に持つファイル、読み取り不可、 書き込み不可のファイル、デバイスファイル、バックアップ中のファイルのサイズ変更、 バックアップ中のファイルの作成および削除、などです。
彼女は 1991 年 10 月の LISA V で結果を発表しています。 torture-testing Backup and Archive Programs を参照してください。

14.9.8. 緊急時のリストア手順

惨事が起きる前に

発生する可能性があるどのような惨事に対しても、 備えるのに必要な手順は以下の 4 ステップだけです。

最初に、 各ディスクのディスクラベルとファイルシステムテーブル (/etc/fstab)、ブートメッセージ全体をそれぞれ 2 枚ずつ印刷します (たとえば `disklabel da0 | lpr`)。

2 番目に、ブートフロッピーと fix-it フロッピー (boot.flp および fixit.flp) にそのシステムのデバイスがすべて含まれているか確認します。
最も簡単に確認する方法は、フロッピーをドライブに入れてマシンをリブートしてブートメッセージを確認することです。 あなたのシステムのデバイスのすべてが含まれ、 機能していれば 3 番目の手順に進んでください。

さもなければ、 そのシステムのすべてのディスクをマウントでき、テープドライブにもアクセスできるカーネルを備えた カスタムブートフロッピーを 2 枚作成する必要があります。 これらのフロッピーディスクには `fdisk`, `disklabel`, `newfs`, `mount` と、利用するバックアッププログラムが入っていなければなりません。
これらのプログラムはスタティックリンクされていなければなりません。 `dump` を使用するのなら、このフロッピーには `restore` も含まれていなければなりません。

3 番目に、定期的にバックアップテープを作成します。
最後のバックアップの後で行われた変更は、回復できずに失われます。
バックアップテープにライトプロテクトを施してください。

4 番目に、フロッピーディスク (boot.flp と fixit.flp、 か、第 2 段階で作成した 2 枚のカスタムブートフロッピーディスクのどちらか) およびバックアップテープのテストをします。
手順のメモを作りましょう。 このメモはブートフロッピー、印刷した紙、バックアップテープと一緒に保存しておきます。 リストアを行うときには、このメモがバックアップテープを壊すのを防ぐくらい取り乱しているかもしれません (どのように? `tar xvf /dev/sa0` の代わりに、うっかり `tar cvf /dev/sa0` と入力してバックアップテープを上書きしてしまうかもしれません)。

訳注

上書きはライトプロテクトをしておけば防げますが、
何らかの原因でプロテクトがはずれているかもしれません。
ちなみに訳者の経験から言えば、 上のようなミスタイプは結構起きます。

安全性を増すために、毎回、 ブートフロッピーを作成し、 2 巻のバックアップテープを取ります。一方を離れた場所に保管します。 離れた場所は同じ事務所の建物の地下室ではいけません。世界貿易センタービルにあった数多くの会社は、 苦い経験によりこの教訓を得ました。離れた場所とは、コンピュータやディスクドライブから十分な距離を取って 物理的に分離されていなければなりません。

例 30. ブートフロッピーを作成するスクリプト

```
#!/bin/sh
#
# create a restore floppy
#
```

```
# format the floppy
#
PATH=/bin:/sbin:/usr/sbin:/usr/bin

fdformat -q fd0
if [ $? -ne 0 ]
then
     echo "Bad floppy, please use a new one"
     exit 1
fi

# place boot blocks on the floppy
#
disklabel -w -B /dev/fd0c fd1440

#
# newfs the one and only partition
#
newfs -t 2 -u 18 -l 1 -c 40 -i 5120 -m 5 -o space /dev/fd0a

#
# mount the new floppy
#
mount /dev/fd0a /mnt

#
# create required directories
#
mkdir /mnt/dev
mkdir /mnt/bin
mkdir /mnt/sbin
mkdir /mnt/etc
mkdir /mnt/root
mkdir /mnt/mnt          # for the root partition
mkdir /mnt/tmp
mkdir /mnt/var

#
# populate the directories
#
if [ ! -x /sys/compile/MINI/kernel ]
then
     cat << EOM
The MINI kernel does not exist, please create one.
Here is an example config file:
#
# MINI - A kernel to get FreeBSD onto a disk.
#
machine         "i386"
cpu             "I486_CPU"
ident           MINI
```

```
maxusers        5

options         INET                    # needed for _tcp _icmpstat _ipstat
                                        #             _udpstat _tcpstat _udb
options         FFS                     #Berkeley Fast File System
options         FAT_CURSOR              #block cursor in syscons or pccons
options         SCSI_DELAY=15           #Be pessimistic about Joe SCSI device
options         NCONS=2                 #1 virtual consoles
options         USERCONFIG              #Allow user configuration with -c XXX

config          kernel  root on da0 swap on da0 and da1 dumps on da0

device          isa0
device          pci0

device          fdc0    at isa? port "IO_FD1" bio irq 6 drq 2 vector fdintr
device          fd0 at fdc0 drive 0

device          ncr0

device          scbus0

device          sc0 at isa? port "IO_KBD" tty irq 1 vector scintr
device          npx0    at isa? port "IO_NPX" irq 13 vector npxintr

device          da0
device          da1
device          da2

device          sa0

pseudo-device   loop            # required by INET
pseudo-device   gzip            # Exec gzipped a.out's
EOM
     exit 1
fi

cp -f /sys/compile/MINI/kernel /mnt

gzip -c -best /sbin/init > /mnt/sbin/init
gzip -c -best /sbin/fsck > /mnt/sbin/fsck
gzip -c -best /sbin/mount > /mnt/sbin/mount
gzip -c -best /sbin/halt > /mnt/sbin/halt
gzip -c -best /sbin/restore > /mnt/sbin/restore

gzip -c -best /bin/sh > /mnt/bin/sh
gzip -c -best /bin/sync > /mnt/bin/sync

cp /root/.profile /mnt/root

cp -f /dev/MAKEDEV /mnt/dev
```

```
chmod 755 /mnt/dev/MAKEDEV

chmod 500 /mnt/sbin/init
chmod 555 /mnt/sbin/fsck /mnt/sbin/mount /mnt/sbin/halt
chmod 555 /mnt/bin/sh /mnt/bin/sync
chmod 6555 /mnt/sbin/restore

#
# create the devices nodes
#
cd /mnt/dev
./MAKEDEV std
./MAKEDEV da0
./MAKEDEV da1
./MAKEDEV da2
./MAKEDEV sa0
./MAKEDEV pty0
cd /

#
# create minimum file system table
#
cat &gt; /mnt/etc/fstab &lt;&lt;EOM
/dev/fd0a    /    ufs    rw  1  1
EOM

#
# create minimum passwd file
#
cat > /mnt/etc/passwd <<EOM
root:*:0:0:Charlie &:/root:/bin/sh
EOM

cat > /mnt/etc/master.passwd <<EOM
root::0:0::0:0:Charlie &:/root:/bin/sh
EOM

chmod 600 /mnt/etc/master.passwd
chmod 644 /mnt/etc/passwd
/usr/sbin/pwd_mkdb -d/mnt/etc /mnt/etc/master.passwd

#
# umount the floppy and inform the user
#
/sbin/umount /mnt
echo "The floppy has been unmounted and is now ready."
```

惨事の後は

重要な問題は、ハードウェアが生き残ったかどうかです。　　定期的にバックアップを取っていれば、

ソフトウェアについて心配する必要はありません。

ハードウェアに障害があれば、 コンピュータを使用する前にその部品を交換してください。

ハードウェアに問題が無ければ、フロッピーを確認してください。
カスタムブートフロッピーディスクを使用しているのであれば、 シングルユーザモードでブートして (boot: プロンプトで -s を入力します)、 次の段落は飛ばしてください。

boot.flp と fixit.flp を使用しているのであればこのまま読み進めてください。 boot.flp フロッピーをフロッピードライブに入れて、 コンピュータを起動してください。 本来のインストールメニューが画面に表示されます。 Fixit–Repair mode with CDROM or floppy. オプションを選択します。指示された通り fixit.flp をいれてください。 restore とその他必要となるプログラムは /mnt2/stand にあります。

そして、ファイルシステムを一つずつ回復します。

最初のディスクのルートパーティションを mount してみてください (たとえば mount /dev/da0a /mnt)。 ディスクラベルが破壊されている場合は、disklabel を用いてあらかじめ印刷して保存しておいた通りにパーティションを作り直し、ディスクラベルを作成してください。 newfs を使用してファイルシステムを作り直します。 ルートパーティションを読み書き可能にマウントし直します (mount -u -o rw /mnt)。 バックアッププログラムとバックアップテープを使用して、 このファイルシステムのデータを回復します (たとえば restore vrf /dev/sa0)。 ファイルシステムをアンマウントします (たとえば umount /mnt)。 障害を受けたファイルシステムそれぞれについて繰り返してください。

システムが動き出したら、 新しいテープにデータをバックアップしてください。 どのような理由で再び事故が起きたり、データが失われるかわかりません。
これに数時間を費すことで、後々の災難から救われます。

14.10. ネットワーク、メモリ、そしてファイルベースのファイルシステム

FreeBSD にはフロッピーや CD, ハードディスクなどの手元の計算機に取り付けたディスクの他に、別の形態のディスク、仮想ディスク、もあります。

これには、Network File System のようなネットワークファイルシステムや Coda, メモリベースのファイルシステムおよびファイルベースのファイルシステムがあります。

稼働させている FreeBSD のバージョンによって、 ファイルベースおよびメモリベースのファイルシステムを作成したり操作するために、異なるツールを使用しなければならないでしょう。

FreeBSD 4.X の使用者は必要なデバイスを作成するために MAKEDEV(8) を使用しなければならないでしょう。FreeBSD 5.0 以降では、devfs(5) がデバイスノードを自動的に割り当ててくれるので、 使用者が意識する必要はありません。

14.10.1. FreeBSD 4.X でファイル中に構築されるファイルシステム

vnconfig(8) ユーティリティを使えば擬似ディスクデバイスを設定し、 有効にすることができます。

vnode とはファイルの内部的な表現方法であり、ファイルに関する操作の中心となるものです。つまり、vnconfig(8) はファイルシステムを生成したり操作したりするためにファイルを用いるのです。 一つ例を挙げると、ファイルに収められたフロッピーや CD-ROM のイメージをマウントするために用いることができます。

vnconfig(8) を使用するためには、 カーネルが vn(4) デバイスに対応している必要があります。そうでなければ、カーネルコンフィギュレーションファイルに次の行を追加してカーネルを再構築し、システムを再起動してください。

```
pseudo-device vn
```

既にあるファイルシステムイメージのマウント

例 *31. FreeBSD 4.X* での *vnconfig* を用いた既存のファイルシステムイメージのマウント

```
# vnconfig vn0 diskimage
# mount /dev/vn0c /mnt
```

vnconfig(8) を用いたファイルシステムイメージの新規作成

例 *32.* `vnconfig` を用いたファイルベースディスクの新規作成

```
# dd if=/dev/zero of=newimage bs=1k count=5k
5120+0 records in
5120+0 records out
# vnconfig -s labels -c vn0 newimage
# disklabel -r -w vn0 auto
# newfs vn0c
Warning: 2048 sector(s) in last cylinder unallocated
/dev/vn0c:     10240 sectors in 3 cylinders of 1 tracks, 4096 sectors
        5.0MB in 1 cyl groups (16 c/g, 32.00MB/g, 1280 i/g)
super-block backups (for fsck -b #) at:
 32
# mount /dev/vn0c /mnt
# df /mnt
Filesystem  1K-blocks     Used    Avail Capacity  Mounted on
/dev/vn0c        4927        1     4532     0%    /mnt
```

14.10.2. FreeBSD 5.X でファイル中に構築されるファイルシステム

mdconfig(8) ユーティリティは FreeBSD 5.X において メモリディスク (md(4)) を設定し、有効にするために使用されます。 mdconfig(8) を使用するためには md(4) モジュールを読み込むか、 カーネルコンフィギュレーションファイルに md(4) デバイスを追加してカーネルを再構築し、システムを再起動してください。

```
device md
```

mdconfig(8) コマンドは、 三つのタイプのメモリベース仮想ディスクに対応しています。 malloc(9) を用いて割り当てられたメモリディスク、 ファイルをベースにしたメモリディスク、 およびスワップ領域をベースにしたメモリディスクです。
想定される使用法は、ファイル内に保持されたフロッピーイメージまたは CD イメージをマウントすることです。

既にあるファイルシステムイメージのマウント

例 *33. FreeBSD 5.X* での `mdconfig` を用いた既存のファイルシステムイメージのマウント

```
# mdconfig -a -t vnode -f diskimage -u 0
# mount /dev/md0c /mnt
```

mdconfig(8) を用いたファイルシステムイメージの新規作成

例 *34.* `mdconfig` を用いたファイルシステムイメージの新規作成

```
# dd if=/dev/zero of=newimage bs=1k count=5k
5120+0 records in
5120+0 records out
# mdconfig -a -t vnode -f newimage -u 0
# disklabel -r -w md0 auto
# newfs md0c
/dev/md0c: 5.0MB (10240 sectors) block size 16384, fragment size 2048
    using 4 cylinder groups of 1.27MB, 81 blks, 256 inodes.
super-block backups (for fsck -b #) at:
 32, 2624, 5216, 7808
# mount /dev/md0c /mnt
# df /mnt
Filesystem  1K-blocks     Used    Avail Capacity  Mounted on
/dev/md0c        4846        2     4458     0%    /mnt
```

`-u` オプションを用いて ユニット番号を指定しない場合、mdconfig(8) は未使用のデバイスを自動的に選択するために md(4) デバイスの auto-unit 機能を使用します。 割り当てられたユニットの名前は md4 のように標準出力に出力されます。mdconfig(8) の詳細についてはマニュアルページを参照してください。

FreeBSD 5.1-RELEASE から、従来の disklabel(8) プログラムは bsdlabel(8) ユーティリティに置き換えられました。bsdlabel(8) では、 使用されていないオプションおよびパラメタの数多くが削除されました。 たとえば `-r` オプションは bsdlabel(8) では取り除かれました。詳細については bsdlabel(8) マニュアルページを参照してください。

mdconfig(8) ユーティリティは大変役に立ちますが、

ファイルベースのファイルシステムを作成するために、
多くのコマンドの入力が必要となります。FreeBSD 5.0 では mdmfs(8)
と呼ばれるツールも用意されています。このプログラムは mdconfig(8) を用いて md(4)
ディスクを設定し、newfs(8) を用いて UFS ファイルシステムを作成し、mount(8)
を用いてマウントします。たとえば、上記と同じファイルシステムを作成し、
マウントしたい場合は、下記のように入力するだけです。

例 *35.* `mdmfs` を用いたファイルベースディスクの設定とマウント

```
# dd if=/dev/zero of=newimage bs=1k count=5k
5120+0 records in
5120+0 records out
# mdmfs -F newimage -s 5m md0 /mnt
# df /mnt
Filesystem 1K-blocks Used Avail Capacity  Mounted on
/dev/md0        4846    2  4458     0%    /mnt
```

ユニット番号を指定せずに `md` オプションを使用した場合、mdmfs(8)
は未使用のデバイスを自動的に選択するために md(4) デバイスの auto-unit
機能を使用します。mdmfs(8) についての詳細はマニュアルページを参照してください。

14.10.3. FreeBSD 4.X でのメモリベースのファイルシステム

md(4) ドライバは FreeBSD 4.X
においてメモリファイルシステムを作成するために単純で効果的な手段です。 メモリを割り当てるために
malloc(9) 関数が使用されます。

vnconfig(8) を用いて作成したファイルシステムを例に取ると、 以下のようにします。

例 *36. FreeBSD 4.X* での *md* メモリディスク

```
# dd if=newimage of=/dev/md0
5120+0 records in
5120+0 records out
# mount /dev/md0c /mnt
# df /mnt
Filesystem  1K-blocks     Used    Avail Capacity  Mounted on
/dev/md0c        4927        1     4532     0%    /mnt
```

詳細については md(4) マニュアルページを参照してください。

14.10.4. FreeBSD 5.X でのメモリベースのファイルシステム

メモリベースおよびファイルベースのファイルシステムに対しても 同じツール (mdconfig(8) または
mdmfs(8)) を使用できます。 メモリベースのファイルシステムに対する記憶領域は malloc(9)
関数を用いて割り当てられます。

例 37. mdconfig を用いたメモリベースディスクの新規作成

```
# mdconfig -a -t malloc -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, 256 inodes.
    with soft updates
super-block backups (for fsck -b #) at:
 32, 2624, 5216, 7808
# mount /dev/md1 /mnt
# df /mnt
Filesystem 1K-blocks Used Avail Capacity  Mounted on
/dev/md1        4846    2  4458     0%    /mnt
```

例 38. mdmfs を用いたメモリベースディスクの新規作成

```
# mdmfs -M -s 5m md2 /mnt
# df /mnt
Filesystem 1K-blocks Used Avail Capacity  Mounted on
/dev/md2        4846    2  4458     0%    /mnt
```

mdconfig(8) のコマンドラインの malloc を swap に置き換えることで、malloc(9) 関数によるファイルシステムを使用する代わりに スワップ領域を使用することが可能です。デフォルトでは mdmfs(8) ユーティリティはスワップベースのディスクを作成します (-M なし)。詳細は mdconfig(8) および mdmfs(8) マニュアルページを参照してください。

14.10.5. システムからメモリディスクを切り離す

メモリベースまたはファイルベースのファイルシステムが使用されていない場合、 すべてのリソースをシステムに開放するべきです。 はじめにファイルシステムをアンマウントします。 次にシステムからディスクを切り離し、リソースを開放するために mdconfig(8) を使用します。

たとえば /dev/md4 によって使用されたすべてのリソースを切り離し、開放するには以下のようにします。

```
# mdconfig -d -u 4
```

mdconfig -l コマンドを使用することによって、設定された md(4) デバイスについての情報を表示することが可能です。

FreeBSD 4.X では vnconfig(8) はデバイスを切り離すのに使用されます。たとえば /dev/vn4 によって使用されたすべてのリソースを切り離し、開放するには以下のようにします。

```
# vnconfig -u vn4
```

14.11. ファイルシステムのスナップショット

FreeBSD 5.0 は Soft Updates と協調するファイルシステムスナップショットという新しい機能を提供します。

スナップショットは指定したファイルシステムのイメージを作成し、
また、ファイルとして扱うことができるようになります。
スナップショットファイルはアクションが実行されるファイルシステム内で作成されなければなりません。 また、ユーザは一つのファイルシステムあたり 20 までスナップショットを作成することができます。
有効なスナップショットはスーパーブロック内に記録されるので、
リブートしてから永続的にアンマウントおよびリマウントを記録します。
スナップショットが必要無くなったときは、 標準の rm(1) コマンドを用いて削除することができます。
スナップショットはどんな順番で削除してもよいのですが、
その他のスナップショットが開放されたブロックのうちいくらかをおそらく必要とするので、
使用されていたすべてのスペースを得られるとは限りません。

初めてスナップショットを作成すると、root でさえも書き込めないように schg フラグ (chflags(1) のマニュアルページを参照) が設定されます。 unlink(1) コマンドは、スナップショットに schg フラグが設定されていてもそれらを削除することのできる例外です。
したがって、スナップショットファイルを削除する前に、 schg フラグをクリアする必要はありません。

スナップショットは mount(8) コマンドを用いて作成されます。 /var のスナップショットを /var/snapshot/snap に作成したいときは、 以下のコマンドを使用します。

```
# mount -u -o snapshot /var/snapshot/snap /var
```

また、スナップショットを作成するのに mksnap_ffs(8) も使えます。

```
# mksnap_ffs /var /var/snapshot/snap
```

スナップショットにはいくつかの利用法があります。

- スナップショットをバックアップ目的に使用する管理者もいます。 なぜならスナップショットは CD やテープに転送できるからです。
- ファイルの完全性を検証するために、 fsck(8) をスナップショットに実行してもよいでしょう。 スナップショットをマウントしたときにそのファイルシステムがクリーンであったとすると、 そのスナップショットをマウントするときはいつでもクリーンな (そして変更のない) 結果を得るでしょう。 これは本質的には バックグラウンド fsck(8) が行うことです。
- スナップショット上で dump(8) ユーティリティを実行すると、 スナップショットのファイルシステムとタイムスタンプが一致するダンプが返されるでしょう。 dump(8) は -L オプションを使用することで、 一つのコマンドでスナップショットをとり、ダンプイメージを作成して、スナップショットを削除することが可能です。

- ファイルシステムの "凍結された" イメージとしてスナップショットを mount(8) します。/var/snapshot/snap のスナップショットを mount(8) するには以下のようにします。

```
# mdconfig -a -t vnode -f /var/snapshot/snap -u 4
# mount -r /dev/md4 /mnt
```

これで /mnt にマウントした 凍結状態の /var ファイルシステム構造を探索できます。すべてがスナップショットが作成された時と同じ状態になるはずです。ただし、
以前に作成されたスナップショットがサイズ 0 のファイルとして現れることが唯一の例外です。
スナップショットの使用を終えた場合、以下のようにアンマウントできます。

```
# umount /mnt
# mdconfig -d -u 4
```

softupdates およびファイルシステムスナップショットに関する詳細については、http://www.mckusick.com/ にある Marshall Kirk McKusick のウェブサイトを参照してください。ここには技術的な論文もあります。

14.12. ファイルシステムクォータ

クォータは OS の持っているオプショナルな機能であり、ファイルシステム毎にユーザやグループのメンバが使用するディスク容量やファイルの数を制限することができます。
この機能は、あるユーザやグループに割り当てられるリソースの量を制限することが望ましいようなタイムシェアリングシステムにおいてよく用いられます。
この機能を用いることによって使用可能なディスク容量の全てを一人のユーザやユーザのグループが使ってしまうことを防ぐことができます。

14.12.1. ディスククォータを使うためのシステム設定

ディスククォータの設定を始める前に、
まずはカーネルにクォータが組み込まれていることを確認しましょう。
カーネルのコンフィグレーションファイルに次の行を入れます。

```
options QUOTA
```

標準の GENERIC カーネルでは、この機能は有効になっていませんので、
ディスククォータを利用するためには上記を設定後カーネルを構築しなおし、
作成されたカスタムカーネルをインストールしなければいけません。
カーネルのコンフィグレーションに関しては FreeBSD カーネルのコンフィグレーション をご覧ください。

次に /etc/rc.conf でディスククォータを有効にする必要があります。 次の行を加えましょう。

```
enable_quotas="YES"
```

起動時の動作をさらに細かくコントロールするためにもう一つ設定用の変数があります。
通常、起動時には quotacheck(8)
によりそれぞれのファイルシステムのクォータの整合性がチェックされます。 quotacheck(8) の役割は、
クォータデータベースのデータが正しくファイルシステム上のデータを反映しているか確認することです
。 これはかなり時間を食う処理であり、 起動にかかる時間に大きな影響を及ぼします。
このステップをとばしたい人のために /etc/rc.conf に次の変数が用意されています。

```
check_quotas="NO"
```

もし 3.2-RELEASE よりも前の FreeBSD を使っているならば設定はもっと単純で、一つの変数のみです。
次の行を /etc/rc.conf で設定してください。

```
check_quotas="YES"
```

最後に、ファイルシステム毎にディスククォータを有効にするために /etc/fstab
を編集する必要があります。 ここでユーザもしくはグループ、
あるいはその両方にクォータを設定することができるのです。

あるファイルシステム上にユーザ毎のクォータを有効にする場合には、 /etc/fstab
中でクォータを有効にしたいファイルシステムエントリのオプション部に userquota を加えます。
例えば次のようになります。

```
/dev/da1s2g   /home    ufs rw,userquota 1 2
```

同様に、グループクォータを有効にするには userquota キーワードの代わりに groupquota を用います。
ユーザとグループの両方のクォータを有効にするには次のようにします。

```
/dev/da1s2g    /home    ufs rw,userquota,groupquota 1 2
```

デフォルトでは、 クォータファイルはそのファイルシステムのルートディレクトリに
ユーザ用、グループ用それぞれ quota.user, quota.group という名前で置かれます。さらに詳しい情報は
fstab(5) をご覧ください。fstab(5)
マニュアルには別の場所を指定することができると書いてはありますが、
あまり勧められません。なぜなら、
様々なクォータ関係のユーティリティがそれにうまく対処できるようにないためです。

この時点で、 一度システムを再起動して新しいカーネルで立ち上げましょう。 /etc/rc
が自動的に適当なコマンドを実行し、 /etc/fstab
で有効にした全てのクォータ用に初期ファイルを作ってくれます。
従って、空のクォータファイルを手で作る必要は一切ありません。

通常の運用では quotacheck(8) や quotaon(8), quotaoff(8)
といったコマンドを手で動かす必要はないのですが、
慣れるためにもこれらのマニュアルは読んでおきましょう。

14.12.2. クォータリミットの設定

一旦クォータを有効にしたら本当に有効になっているのか確認しておきましょう。簡単な方法は次のコマンドを実行することです。

```
# quota -v
```

ディスクの使用状況と、クォータが有効になっているファイルシステムのクォータリミットが一行にまとめて出力されるでしょう。

さあ、edquota(8) でクォータリミットを設定する準備ができました。

ユーザやグループが使用できるディスク容量や作成できるファイルの数に制限をかけるにはいくつかのオプションがあります。割り当てディスク容量を制限 (ブロッククォータ) することもファイル数を制限 (inode クォータ) することも、両者を組み合わせることもできるのです。
これらの制限はそれぞれさらに二つのカテゴリ、
ハードリミットとソフトリミット、に分けることができます。

ハードリミットを越えることはできません。 あるユーザが一旦ハードリミットにたっした場合、そのファイルシステムではそれ以上の割り当ては望めません。 例えばあるファイルシステム上に 500 ブロックのハードリミットが設定されており現在 490 ブロックを使用している場合、さらに 10 ブロックしか使えないのです。 11 ブロックを使おうとすると失敗します。

一方、 ソフトリミットはある限られた時間内であれば越えることができます。この時間は猶予期間として知られており、デフォルトでは 1 週間です。
あるユーザが自分のソフトリミットを猶予期間よりも長い間越えているとソフトリミットはハードリミットに変わり、それ以上使用することはできなくなります。
ユーザがソフトリミットよりも減らせば猶予期間はリセットされます。

以下は edquota(8) コマンドを実行した時に見ることになるであろう例です。 edquota(8) コマンドが起動されると環境変数 EDITOR で指定されるエディタに入ります。 EDITOR が設定されていない場合には vi が起動されます。 ここでクォータリミットを編集します。

```
# edquota -u test
```

```
Quotas for user test:
/usr: blocks in use: 65, limits (soft = 50, hard = 75)
        inodes in use: 7, limits (soft = 50, hard = 60)
/usr/var: blocks in use: 0, limits (soft = 50, hard = 75)
        inodes in use: 0, limits (soft = 50, hard = 60)
```

通常、クォータが有効になっているファイルシステム毎に 2 行あります。
一つはブロックリミット用でもう一つは inode リミット用です。
クォータリミットを変更したいところを書き変えるだけでかまいません。
たとえばこのユーザのブロックリミットを、ソフトリミットは 50 から 500 へ、ハードリミットは 75 から 600 に変更する場合、

```
/usr: blocks in use: 65, limits (soft = 50, hard = 75)
```

から

```
 /usr: blocks in use: 65, limits (soft = 500, hard = 600)
```

へ書き換えます。新しいクォータリミットはエディタを終了すれば設定されます。

ある範囲の UID に対してクォータリミットを設定したい場合がありますが、このような時には edquota(8) コマンドの -p オプションを使うといいでしょう。まず、あるユーザに割り当てたいクォータリミットを設定し、次に `edquota -p protouser startuid-enduid` を実行するのです。例えばユーザ `test` にお望みのクォータリミットが付いているとしましょう。次のコマンドにより 10,000 から 19,999 の間の UID に対して同じクォータリミットを付けることができるのです。

```
# edquota -p test 10000-19999
```

さらに詳しいことは edquota(8) のマニュアルページをご覧ください。

14.12.3. クォータリミットとディスク使用状況のチェック

quota(1) または repquota(8) といったコマンドを使ってクォータリミットやディスクの利用状況を確認することができます。 quota(1) コマンドは個々のユーザやグループのクォータやディスク利用状況を確認するのに使えます。ユーザは自身のクォータ、そして所属するグループのグループのみ確認することができます。スーパーユーザのみが他のユーザや所属していないグループのクォータと利用状況を見ることができます。 repquota(8) コマンドを使うと、クォータが有効になっているファイルシステム用の全てのクォータやディスク容量のサマリを得ることができます。

以下は二つのファイルシステムにクォータ制限がかけられているユーザに対する`quota -v` コマンドの出力例です。

```
Disk quotas for user test (uid 1002):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
           /usr      65*     50      75  5days       7      50      60
      /usr/var       0      50      75              0      50      60
```

上の例で、/usr ファイルシステム上ではこのユーザは現在 50 ブロックというソフトリミットを 15 ブロックオーバーし 5 日間の猶予期間が残っています。アスタリスク `*` はクォータリミットを越えているユーザを示していることに注意してください。

通常、そのユーザが全く使っていないファイルシステムは、 クォータリミットが付けられているとしても quota(1) コマンドの出力には現われません。 `-v` オプションを用いればそのようなファイルシステム、上の例では /usr/var、 を表示することができます。

14.12.4. NFS 上の クォータ

クォータは NFS サーバ上のクォータサブシステムにより実行されます。 rpc.rquotad(8) デーモンにより、NFS クライアント上の quota(1) コマンドは情報を得ることができ、クライアントマシン上のユーザが自分のクォータの統計を見ることができます。

/etc/inetd.conf において以下のように `rpc.rquotad` を有効にしましょう。

```
rquotad/1      dgram rpc/udp wait root /usr/libexec/rpc.rquotad rpc.rquotad
```

そして以下のように `inetd` を再起動します。

```
# kill -HUP `cat /var/run/inetd.pid`
```

14.13. ディスクパーティションの暗号化

FreeBSD は無許可のデータアクセスに対する優れたオンライン保護機能を提供します。 ファイルのパーミッションおよび強制的アクセスコントロール (MAC: Mandatory Access Control) (Mandatory Access Control (MAC) を参照) は、コンピュータが動作中で、OS が実行中であるときに、無許可の第三者がデータにアクセスするのを防ぐことに役立ちます。
しかしながら、攻撃者がコンピュータに物理的にアクセスし、
機密データをコピーし分析するためにコンピュータのハードドライブを別のシステムに移動させることができれば、 OS によって強化された許可属性は意味をなさなくなります。

攻撃者が電源の落ちたコンピュータや ハードドライブを手にいれる手段にかかわらず、 GEOM ベースのディスク暗号化 (gbde: GEOM Based Disk Encryption) は、著しい資源を持ち本気で攻撃を仕掛けるつもりでやってきた攻撃者からさえもコンピュータのファイルシステム上にあるデータを保護することができます。
個々のファイルだけを暗号化する煩わしい方法と異なり、 gbde は全ファイルシステムを透過的に暗号化します。
平文テキストは決してハードドライブのプラッタに関係しません。

14.13.1. カーネルで gbde を有効にする

1. `root` になる

   gbde の設定をするにはスーパユーザの権限が必要になります。 以下のコマンドを実行して、`root` になってください。

   ```
   % su -
   Password:
   ```

2. オペレーティングシステムのバージョンを確かめる

   gbde(4) が動作するには FreeBSD 5.0 以降が必要です。 以下のコマンドを実行して、

オペレーティングシステムのバージョンを確認してください。

```
# uname -r
5.0-RELEASE
```

3. カーネルコンフィギュレーションファイルに gbde(4) 対応を追加する

   お好みのテキストエディタを使用して、
   以下の行をカーネルコンフィギュレーションファイルに加えます。

   ```
   options GEOM_BDE
   ```

   FreeBSD カーネルを設定、再コンパイル、インストールします。 この手順は FreeBSD カーネルのコンフィグレーション で説明されています。

   新しいカーネルで再起動します。

14.13.2. 暗号化されたハードドライブの準備

以下の例では、システムに新しいハードディスクを追加しようとしています。このシステムは単一の暗号化されたパーティションを保持することになります。 このパーティションは /private としてマウントされます。gbde は /home および /var/mail を暗号化するのにも使用できますが、より複雑な指示を必要となるのでこの解説の範疇を越えています。

1. 新しいハードドライブを追加する

   ディスクの追加 で説明されている通りに新しいドライブをシステムに設置します。
   この例では、新しいハードドライブは /dev/ad4s1c パーティションに 加えられたものとします。
   /dev/ad0s1* デバイスは、この例のシステム上に存在する標準的な FreeBSD パーティションを表します。

   ```
   # 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. gbde ロックファイルを保持するディレクトリを作成する

   ```
   # mkdir /etc/gbde
   ```

   gbde ロックファイルには、
   暗号化されたパーティションにアクセスするのに必要となる情報が格納されています。
   ロックファイルにアクセスしない場合、 gbde は 膨大な手動による介在なしには (ソフトウェアは対応していません)、暗号化されたパーティションに含まれるデータを解読することはできないでしょう。
   それぞれの暗号化されたパーティションは別々のロックファイルを使用します。

3. gbde パーティションを初期化する

gbde パーティションは使用する前に初期化されなければなりません。この初期化は一度だけ実行される必要があります。

```
# gbde init /dev/ad4s1c -i -L /etc/gbde/ad4s1c
```

エディタが開くので、 テンプレートをもとにさまざまなオプションを設定してください。 UFS1 または UFS2 で使用するには、sector_size を 2048 に設定してください。

```
$FreeBSD: src/sbin/gbde/template.txt,v 1.1 2002/10/20 11:16:13 phk Exp $
#
# 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
[...]
```

gbde(8) はデータを保護するのに使用するパスフレーズを二度尋ます。パスフレーズはそれぞれ同じでなければなりません。 データを保護する gbde の能力は、あなたが選択したパスフレーズの品質に完全に依存します。

gbde init コマンドは gbde パーティションに対するロックファイルを作成します。この例では /etc/gbde/ad4s1c に格納されます。

gbde ロックファイルは、すべての暗号化されたパーティションの内容とともにバックアップされなければ なりません。 ロックファイルだけを削除している間、ロックファイルなしでは信念の固い攻撃者が gbde パーティションを解読することを防ぐことができない一方で、正当な所有者は、gbde(8) およびこの設計者にまったく支持されない膨大な量の作業なしには、暗号化されたパーティション上のデータにアクセスすることができないでしょう。

4. カーネルに暗号化されたパーティションを接続する

```
# gbde attach /dev/ad4s1c -l /etc/gbde/ad4s1c
```

暗号化されたパーティションを初期化する際に選択したパスフレーズを入力するように求められます。 新しい暗号化デバイスは /dev に /dev/device_name.bde として現れます。

```
# 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. 暗号化デバイス上にファイルシステムを作成する

   カーネルに暗号化デバイスが接続されると、 デバイス上にファイルシステムを作成できます。暗号化デバイス上にファイルシステムを作成するには newfs(8) を使用します。従来の UFS1 ファイルシステムで初期化するより、 新しい UFS2 ファイルシステムで初期化した方が高速なので、 -O2 オプションとともに newfs(8) を使用することが推奨されています。

   FreeBSD 5.1-RELEASE 以降では、-O2 オプションはデフォルトです。

   ```
   # newfs -U -O2 /dev/ad4s1c.bde
   ```

   newfs(8) は、デバイス名に *.bde 拡張子によって認識される、 接続された gbde パーティションに対して実行されなければなりません。

6. 暗号化パーティションをマウントする

   暗号化ファイルシステムに対するマウントポイントを作成します。

   ```
   # mkdir /private
   ```

   暗号化ファイルシステムをマウントします。

   ```
   # mount /dev/ad4s1c.bde /private
   ```

7. 暗号化ファイルシステムが利用可能か確かめる

   これで暗号化ファイルシステムは df(1) で見ることができ、 利用する準備ができました。

   ```
   % 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
   ```

14.13.3. 存在する暗号化ファイルシステムをマウントする

システムを起動する度に、すべての暗号化ファイルシステムは 使用前にカーネルに接続し、エラーの有無をチェックし、マウントする必要があります。 必要なコマンドは root ユーザとして実行されなければなりません。

1. カーネルに gbde パーティションを接続する

```
# gbde attach /dev/ad4s1c -l /etc/gbde/ad4s1c
```

パーティションの暗号化を初期化する際に選択したパスフレーズを入力するように求められるでしょう。

2. ファイルシステムのエラーをチェックする

暗号化ファイルシステムを自動的にマウントするために /etc/fstab に設定を掲載することはまだできないため、 マウントする前に fsck(8) を実行して、ファイルシステムのエラーをチェックしなければなりません。

```
# fsck -p -t ffs /dev/ad4s1c.bde
```

3. 暗号化ファイルをマウントする

```
# mount /dev/ad4s1c.bde /private
```

これで暗号化ファイルシステムが利用できるようになりました。

暗号化パーティションを自動的にマウントする

スクリプトを作成して、暗号化パーティションを自動的に接続、チェック、マウントすることは可能です。しかしながら、 安全上の理由によりスクリプトに gbde(8) パスワードを含めるべきではありません。その代わりに、コンソールまたは ssh(1) による接続からパスワードを入力するようなスクリプトが手動で実行されることが推奨されます。

14.13.4. gbde が採用した暗号の保護

gbde(8) は 128bit AES の CBC モードを使用してセクタペイロードを暗号化します。ディスク上のそれぞれのセクタは異なる AES 鍵で暗号化されます。セクタ鍵がユーザが入力したパスフレーズからどのように導き出されるかを含め、 gbde の暗号手法の設計についての詳細は、 gbde(4) を参照してください。

14.13.5. 互換性に関する問題

sysinstall(8) は gbde 暗号化デバイスと互換性がありません。 sysinstall(8) を実行する前に *.bde デバイスはすべてカーネルから切断されなければなりません。 そうしないと、sysinstall(8) が初めにデバイスを走査する際にクラッシュしてしまうでしょう。
暗号化デバイスを切断するには、以下のコマンドを使用します。

```
# gbde detach /dev/ad4s1c
```

Chapter 15. 地域化 (localization) - i18n/L10n の利用と設定

15.1. この章では

FreeBSD は、 ユーザーおよび貢献者が世界中に分散したプロジェクトです。 そのため、FreeBSD は多くの言語への地域化に対応しており、
ユーザは、英語以外の言語を見たり、入力したり、処理したりできます。
中国語、ドイツ語、日本語、韓国語、フランス語、ロシア語、
ベトナム語など、主要な言語のほとんどから選ぶことができますが、
これらに限定されるわけではありません。

internationalization は、i18n と短縮して表記されます。 これは `internationalization` の最初と最後の間の文字数に由来します。 L10n も同じ命名法を用いて `localization` を縮めたものです。 i18n/L10n された (すなわち国際化/地域化された) 手法、プロトコル、アプリケーションは、自分達の好みの言語を使うことを可能にしてくれます。

この章では、FreeBSD の国際化 (internationalization) と地域化 (localization) 機能について解説します。 この章では、以下の分野について説明します。

- ロケール名がどのように定義されるか。
- ログインシェルでロケールを設定するにはどうするか。
- コンソールを英語以外の言語用に設定するにはどうするか。
- 様々な言語で Xorg を設定するにはどうすればよいか。
- 国際化 (i18n) されたアプリケーションの見つけ方。
- 特定の言語に設定するための情報はどこにあるか。

この章を読む前に、以下のことを理解しておく必要があります。

- サードパーティ製アプリケーションのインストール方法

15.2. 地域化の利用

地域化の設定は、言語コード、 国コード、エンコーディングという三つの要素を基本とします。 ロケール名はこれらから以下のように構成されます。

```
言語コード_国コード.エンコーディング
```

言語コード および 国コード は、 国と言語を特定するために用いられます。 言語および国コード では、言語コード__国コード の例を示します

表 *13.* 言語および国コード

言語_国コード	説明
en_US	英語、合衆国

言語_国コード	説明
ru_RU	ロシア語、ロシア
zh_TW	繁体字中国語、台湾

利用可能なすべてのロケールを調べるには、 以下のように実行してください。

```
% locale -a | more
```

現在のロケールの設定を調べるには、 以下のコマンドを実行してください。

```
% locale
```

言語固有の、C 言語の char で表現できる ISO8859-1, ISO8859-15, KOI8-R, CP437 といったシングルバイトの文字セットについては、 multibyte(3) を参照してください。
現在有効な文字セットのリストは、IANA Registry で確認できます。

いくつかの言語 (例えば中国語や日本語) は、 ASCII 文字では表すことができないので、ワイド文字や多バイト文字を用いた拡張された言語のエンコードが必要となります。 ワイド/多バイトのエンコーディングの例は、EUC および Big5 です。 古いアプリケーションの中には、これらのエンコードを誤ってコントロール文字として認識するものがありますが、
最近のアプリケーションは、大抵これらの文字を認識します。
実装方法にも依りますが、アプリケーションのコンパイル時もしくは configure 時に、ワイド/多バイト文字のサポートを指定する必要があるかも知れません。

FreeBSD では、Xorg 互換のロケール符号を用いています。

以下では、FreeBSD システムにおいてロケールを設定する方法について説明します。 次の節では、i18n に対応するアプリケーションの見つけ方およびコンパイル方法について説明します。

15.2.1. ログインシェルでロケールを設定する

ロケールの設定は、ユーザの ~/.login_conf、 またはユーザのシェルの初期設定ファイルである ~/.profile, ~/.bashrc または ~/.cshrc で行います。

以下の二つの環境変数を設定する必要があります。

- LANG: ロケールを設定します。
- MM_CHARSET: アプリケーションで使用される MIME 文字セットを指定します。

これらの変数は、ユーザのシェルの設定ファイルに加え、 アプリケーション固有の設定ファイル、および Xorg の設定ファイルにおいても指定される必要があります。

必要な変数を割り当てるには、二つの方法があります。 ログインクラス において割り当てる方法 (推奨される方法です)、および 初期化ファイル で指定する方法です。 次の 2 つの節では、この両方の方法について説明します。

ログインクラスを用いる方法

最初に説明する方法は、すべてのシェルにおいて必要なロケール名と MIME 文字セットを環境変数に割り当てます。これは推奨される方法です。この割り当て方法としては、各ユーザが行う方法と、スーパーユーザがすべてのユーザに対して設定する 2 つの方法があります。

以下の簡単な例では、各ユーザのホームディレクトリの .login_conf で、両方の変数に Latin-1 エンコーディングを設定します。

```
me:\
    :charset=ISO-8859-1:\
    :lang=de_DE.ISO8859-1:
```

これは、BIG-5 エンコーディングされた繁体字中国語用の環境変数を設定するユーザの ~/.login_conf の一例です。中国語、日本語、韓国語用のロケール変数を正しく認識しないソフトウェアに対応するため、より多くの変数に対する設定が行われています。

```
#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
```

もう一つの方法では、スーパーユーザがシステム上のすべてのユーザに対する地域化を設定します。/etc/login.conf の以下の変数により、ロケールおよび MIME 文字セットを設定します。

```
language_name|Account Type Description:\
    :charset=MIME_charset:\
    :lang=locale_name:\
    :tc=default:
```

よって、先ほどの例における Latin-1 に対する設定は、以下のようになります。

```
german|German Users Accounts:\
    :charset=ISO-8859-1:\
    :lang=de_DE.ISO8859-1:\
    :tc=default:
```

詳細に関しては login.conf(5) を参照してください。なお、*russian* クラスはあらかじめ定義されています。

/etc/login.conf を編集したら、
忘れずに以下のコマンドを実行してケイパビリティデータベースをアップデートしてください。

```
# cap_mkdb /etc/login.conf
```

エンドユーザは、変更を反映させるために、各自の ~/.login_conf に対して cap_mkdb コマンドを実行する必要があります。

ログインクラスを変更するユーティリティ

/etc/login.conf を手動により編集する方法に加え、
新たに作成するユーザのロケールを設定するためのユーティリティがあります。

vipw を使って新しいユーザを追加する際には、使用する言語を *language* に指定してください。

```
user:password:1111:11:language:0:0:User Name:/home/user:/bin/sh
```

adduser を使って新しいユーザを追加する場合に、
すべてのユーザに対するデフォルトの言語は事前に設定でき、
個々のユーザに対する言語を指定できます。

新しく追加するすべてのユーザが同じ言語を使う場合には、 /etc/adduser.conf で defaultclass=language と設定してください。

新しいユーザを作成するときに、この設定を変更するには、
以下のプロンプトにおいて希望するロケールを指定してください。

```
Enter login class: default []:
```

もしくは、adduser を実行する際にロケールを指定してください。

```
# adduser -class language
```

pw を使って新しいユーザを追加する場合には、 以下のようにしてロケールを指定してください。

```
# pw useradd user_name -L language
```

すでに存在するユーザのログインクラスを変更するには、 chpass を使用してください。
引数として変更するユーザ名を与えて、 スーパーユーザの権限で実行してください。

```
# chpass user_name
```

シェルの初期化ファイルによる方法

この 2 番目の方法は、 使用するシェルごとに手動での設定が必要なため、推奨されません。

シェル毎に設定ファイルが存在し、その構文はシェルに依存します。 たとえば、sh シェルに対するドイツ語の設定では、 そのユーザのシェルを設定するためだけに、 ~/.profile に以下の行を追加ます。 これらの行を /etc/profile または、 /usr/share/skel/dot.profile に追加すると、すべてのユーザのシェルを設定することが可能です。

```
LANG=de_DE.ISO8859-1; export LANG
MM_CHARSET=ISO-8859-1; export MM_CHARSET
```

しかしながら、csh シェルでは、 設定ファイルの名前や構文は異なります。 ~/.login, /etc/csh.login または /usr/share/skel/dot.login では同じ設定です。

```
setenv LANG de_DE.ISO8859-1
setenv MM_CHARSET ISO-8859-1
```

さらに面倒なことに、 Xorg を設定するための ~/.xinitrc における構文は、使用しているシェルに依存します。 以下の例において、最初は sh シェルに対するもので、2 番目が csh シェルに対するものです。

```
LANG=de_DE.ISO8859-1; export LANG
```

```
setenv LANG de_DE.ISO8859-1
```

15.2.2. コンソールの設定

コンソールで利用可能な地域化されたフォントがあります。 利用できるフォントの一覧を調べるには、ls /usr/share/syscons/fonts と入力してください。 コンソールのフォントを設定するには、 .fnt という拡張子を除いた フォント名 を、 /etc/rc.conf に設定してください。

```
font8x16=フォント名
font8x14=フォント名
font8x8=フォント名
```

以下を /etc/rc.conf に追加することで、 キーマップおよびスクリーンマップを指定できます。

```
scrnmap=スクリーンマップ名
keymap=キーマップ名
keychange="ファンクションキー番号の並び"
```

利用可能なスクリーンマップの一覧を調べるには、 ls /usr/share/syscons/scrnmaps と入力してください。 /etc/rc.conf で スクリーンマップ名 を指定する時は、 .csm という拡張子を除いてください。 スクリーンフォントが bit 8 列を使っている時に文字を疑似グラフィクス領域から外に移動するように、 VGA アダプタがフォント文字マトリクスで bit 8 を bit 9 に拡張することに対処するため、

フォントに適切にマップされたスクリーンマップが必要となります。

利用可能なキーマップの一覧を調べるには、 `ls /usr/share/syscons/keymaps` と入力してください。/etc/rc.conf で キーマップ名 を指定する時には、 .kbd という拡張子を除いてください。再起動せずにキーマップを試すには、 kbdmap(1) を使ってください。

ファンクションキーの並びはキーマップで定義されていないので、
端末タイプに合わせたファンクションキーを設定するために `keychange` のエントリが必要となります。

次に /etc/ttys の中のすべての仮想端末のエントリに対して、正しいコンソール端末タイプを設定してください。文字セットに対する定義済みの端末タイプ は、利用可能な端末タイプの一覧です。

表 *14.* 文字セットに対する定義済みの端末タイプ

文字セット	端末タイプ
ISO8859-1 もしくは ISO8859-15	`cons25l1`
ISO8859-2	`cons25l2`
ISO8859-7	`cons25l7`
KOI8-R	`cons25r`
KOI8-U	`cons25u`
CP437 (VGA のデフォルト)	`cons25`
US-ASCII	`cons25w`

ワイド/多バイト文字の言語については、 その言語に対するコンソールを FreeBSD Ports Collection からインストールしてください。 利用可能な ports は、Ports Collection で利用可能なコンソール にまとめてあります。 インストール後、各 port の pkg-message または、マニュアルページを参照して、設定や使用方法を調べてください。

表 *15. Ports Collection* で利用可能なコンソール

言語	**port** の位置
繁体字中国語 (BIG-5)	chinese/big5con
中国語/日本語/韓国語	chinese/cce
中国語/日本語/韓国語	chinese/zhcon
日本語	chinese/kon2
日本語	japanese/kon2-14dot
日本語	japanese/kon2-16dot

/etc/rc.conf において moused を有効にしている場合には、 追加の設定が必要となるでしょう。デフォルトでは、syscons(4) ドライバのマウスカーソルはキャラクタセット中の `0xd0-0xd3` の範囲を占めています。そのため、 利用している言語がこの範囲のキャラクタセットを使っている場合、次の行を /etc/rc.conf に追加して カーソルの占める範囲を移動してください。

```
mousechar_start=3
```

15.2.3. Xorg の設定

Xorg のインストールおよび設定方法は、 X Window System で説明されています。 Xorg を地域化するための追加のフォントおよび入力方法は、 FreeBSD Ports Collection から利用できます。 フォント、メニューなどのアプリケーション固有の国際化 (i18n) の設定は、 ~/.Xresources において指定でき、 グラフィカルアプリケーションのメニューが選んだ言語で表示されます。

X Input Method (XIM) プロトコルは、Xorg で非英字文字を入力するための標準規格です。 FreeBSD Ports Collection から利用可能なインプットメソッドについては、 利用可能なインプットメソッド にまとめられています。 追加の Fcitx および Uim アプリケーションも利用できます。

表 *16.* 利用可能なインプットメソッド

言語	インプットメソッド
中国語	chinese/gcin
中国語	chinese/ibus-chewing
中国語	chinese/ibus-pinyin
中国語	chinese/oxim
中国語	chinese/scim-fcitx
中国語	chinese/scim-pinyin
中国語	chinese/scim-tables
日本語	japanese/ibus-anthy
日本語	japanese/ibus-mozc
日本語	japanese/ibus-skk
日本語	japanese/im-ja
日本語	japanese/kinput2
日本語	japanese/scim-anthy
日本語	japanese/scim-canna
日本語	japanese/scim-honoka
日本語	japanese/scim-honoka-plugin-romkan
日本語	japanese/scim-honoka-plugin-wnn
日本語	japanese/scim-prime
日本語	japanese/scim-skk
日本語	japanese/scim-tables
日本語	japanese/scim-tomoe
日本語	japanese/scim-uim
日本語	japanese/skkinput
日本語	japanese/skkinput3
日本語	japanese/uim-anthy

言語	インプットメソッド
韓国語	korean/ibus-hangul
韓国語	korean/imhangul
韓国語	korean/nabi
韓国語	korean/scim-hangul
韓国語	korean/scim-tables
ベトナム語	vietnamese/xvnkb
ベトナム語	vietnamese/x-unikey

15.3. 国際化 (i18n) に対応したアプリケーションを見つける

国際化 (i18n) されたアプリケーションは、ライブラリとして i18n 化キットを用いてプログラミングされます。これは開発者が単純なファイルを書いて、表示されるメニューやテキストを各国語に翻訳できるようにしてくれます。

FreeBSD Ports Collection の多くのアプリケーションは、いくつかの言語向けのワイド/多バイト文字への対応を組み込んでいます。そのようなアプリケーションの名前には、容易に認識できるように、`-i18n` と付いています。しかしながら、それらのアプリケーションが必要とする言語に対応しているとは限りません。

いくつかのアプリケーションでは、特定の文字セットを使うようにコンパイルできます。これは大抵 Makefile の中で対処されているか、configure に値を渡すことで対応しています。必要な configure の値や port の構築時に使用するコンパイルオプションを決めるための port の Makefile に関するより詳細な情報については、各 FreeBSD port のソースにある i18n 文書を参照してください。

15.4. 特定の言語にロケールを設定する

この節では、FreeBSD システムをロシア語へ地域化するための設定例を示します。後半では、他の言語への地域化に関する情報を提供します。

15.4.1. ロシア語 (KOI8-R エンコーディング)

この節では、FreeBSD システムをロシア語へ地域化するための設定例を示します。各設定に関するより詳しい説明については、地域化の利用 を参照してください。

このロケールをログインシェルに設定するには、以下の行を各ユーザの ~/.login_conf に追加してください。

```
me:My Account:\
    :charset=KOI8-R:\
    :lang=ru_RU.KOI8-R:
```

コンソールを設定するには、/etc/rc.conf に以下の行を追加してください。

```
keymap="ru.utf-8"
```

```
scrnmap="utf-82cp866"
font8x16="cp866b-8x16"
font8x14="cp866-8x14"
font8x8="cp866-8x8"
mousechar_start=3
```

/etc/ttys の各 ttyv エントリにおいて、 端末タイプとして cons25r を指定してください。

プリンタの設定を行うには、
ロシア語用の文字を搭載したほとんどのプリンタはハードウェアコードページ CP866 を使っているため、KOI8-R を CP866 に変換する専用の出力フィルタが必要となります。
この目的のため、FreeBSD はデフォルトフィルタを /usr/libexec/lpr/ru/koi2alt にインストールします。
このフィルタを使うには、/etc/printcap に以下のエントリを追加してください。

```
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:
```

より詳細な説明については printcap(5) を参照してください。

マウントされた MS-DOS® ファイルシステムにおいてロシア語ファイル名を使えるように設定するには、/etc/fstab にエントリを追加するときに、 以下のように -L とロケール名を含めてください。

```
/dev/ad0s2      /dos/c  msdos   rw,-Lru_RU.KOI8-R 0 0
```

詳しくは、mount_msdosfs(8) を参照してください。

Xorg にロシア語のフォントを設定するには、 x11-fonts/xorg-fonts-cyrillic パッケージをインストールしてください。 その後、/etc/X11/xorg.conf の "Files" セクションを確認してください。 既存の FontPath エントリの_前に_以下の行を追加しなければなりません。

```
FontPath   "/usr/local/lib/X11/fonts/cyrillic"
```

他の Cyrillic フォントは、 Ports Collection から利用できます。

ロシア語のキーボードを使えるようにするには、 以下の行を xorg.conf の "Keyboard" セクションに追加します。

```
Option "XkbLayout"   "us,ru"
Option "XkbOptions"  "grp:toggle"
```

このファイルの中で XkbDisable がコメントアウトされていることを確認してください。

grp:toggle では Right Alt を使い、 grp:ctrl_shift_toggle では Ctrl + Shift を使います。
grp:caps_toggle では、 CapsLock を使います。 従来の CapsLock の機能は、 ラテン文字モードの時のみ

Shift + CapsLock で使うことができます。 Xorg では、理由は不明ですが grp:caps_toggle は動作しません。

キーボードに "Windows®" キーがあり、そのキーにいくつかの非英字キーが割り当てられているようなら、 xorg.conf に以下の行を追加してください。

```
Option "XkbVariant" ",winkeys"
```

ロシア語の XKB キーボードは、地域化されていないアプリケーションではうまく動かないかも知れません。地域化されたアプリケーションは少なくともプログラムの最初の方で XtSetLanguageProc (NULL, NULL, NULL); を呼び出すべきです。

Xorg アプリケーションを地域化する方法については、http://koi8.pp.ru/xwin.html を参照してください。 KOI8-R エンコーディングの詳細については、http://koi8.pp.ru/ を参照してください。

15.4.2. 言語固有のリソース

この節では、 他言語へのロケールの設定に関するリソースの一覧を示します。

台湾向けの繁体字中国語への地域化

FreeBSD-Taiwan プロジェクトは、 FreeBSD を中国語化するための手引き http://netlab.cse.yzu.edu.tw/~statue/freebsd/zh-tut/ を提供しています。

ギリシャ語への地域化

FreeBSD におけるギリシャ語のサポートについての記事は、 公式の FreeBSD ギリシャ語ドキュメンテーションの一部として ここ で読むことができます。この文書は、ギリシャ語で書かれています。

日本語/韓国語への地域化

日本語に関しては http://www.jp.FreeBSD.org/ を、韓国語に関しては http://www.kr.FreeBSD.org/ を参照してください。

英語以外の **FreeBSD** ドキュメント

FreeBSD の文書の一部を他の言語に翻訳してくれている貢献者たちがいます。 これらは FreeBSD ウェブサイト のリンクを辿るか /usr/share/doc から入手できます。

Chapter 16. FreeBSD のアップデートとアップグレード

16.1. この章では

あるリリースから次のリリースまでの期間にも、 FreeBSD の開発は休みなく続けられています。
最新の開発ツリーと同期することを好む人がいる一方で、公式のリリース版を好んで使う方もいます。
しかしながら、公式のリリースといえども、
セキュリティや他の重要な修正のため、時にはアップデートが必要となります。 FreeBSD は手元のシステムを最新の開発ツリーと同期するために必要なツールをすべて用意しているので、使用しているバージョンに関わらず、これらのツールを使って簡単にシステムのバージョンをアップグレードできます。 この章では、開発ブランチを追いかける方法、および、FreeBSD システムをアップデートする基本的なツールについて解説します。

この章では以下について説明します。

- freebsd-update もしくは Git を使った FreeBSD システムの更新方法
- インストールされているシステムと、変更が行われていない状態との比較方法
- Git またはドキュメント用の ports を使って、インストールされているドキュメントを最新版にアップデートする方法
- 2 つの開発ブランチ、FreeBSD-STABLE と FreeBSD-CURRENT の違いについて
- ベースシステム全体を再構築しインストールする方法

この章を読む前に、以下の準備をしましょう。

- ネットワーク接続の適切な設定 (高度なネットワーク)
- サードパーティ製のソフトウェアのインストール方法の習得 (アプリケーションのインストール - packages と ports)

この章を通じて、FreeBSD のソースコードのダウンロードやアップデートに `git` が使われています。 必要に応じて devel/git port または package が使われることもあります。

16.2. FreeBSD Update

すみやかにセキュリティパッチを適用し、 オペレーティングシステムをアップグレードして、最新のリリースに保つことは、システム管理における重要な側面です。 これらの処理を行うために FreeBSD には `freebsd-update` と呼ばれるユーティリティが用意されています。

このユーティリティを用いると、FreeBSD のセキュリティおよび eratta アップデートをバイナリによって行うことができます。
手動でパッチもしくは新しいカーネルをコンパイルし、インストールする必要はありません。
バイナリアップデートは、セキュリティチームがサポートしているすべてのアーキテクチャとリリースで利用できます。 https://www.FreeBSD.org/ja/security/ には、サポートが行われているリリースや保守終了予定日の一覧があります。

このユーティリティは、マイナーリリースであったり、他のリリースブランチへのアップグレードにも対

応しています。
新しいリリースにアップデートする前に、アップデートしようとしているリリースのアナウンスに目を通し、重要な情報がないかどうかを確認してください。 リリースのアナウンスは https://www.FreeBSD.org/ja/releases/ で確認できます。

もし crontab(5) の中に freebsd-update(8) の機能が含まれていたら、オペレーティングシステムのアップグレード作業を終えるまでは無効にしてください。

この節では、freebsd-update で使われる設定ファイルの説明、セキュリティパッチの適応方法のデモンストレーション、
オペレーティングシステムをアップグレードする際に考慮すべき点について説明します。

16.2.1. 設定ファイル

freebsd-update のデフォルトの設定ファイルは、そのままでも用いることができます。 /etc/freebsd-update.conf の設定をデフォルトからきめ細かく調整して、アップデートプロセスを制御するユーザもいます。
利用可能なオプションについてはこのファイルのコメントで説明されていますが、以下の項目については補足が必要でしょう。

```
# Components of the base system which should be kept updated.
Components world kernel
```

このパラメータは、FreeBSD のどの部分を最新に維持するかを設定します。
デフォルトでは、ベースシステム全体、そしてカーネルをアップデートします。 src/base や src/sys のように、個々の項目を指定することもできます。
この部分についてはデフォルトのままにしておき、アップデートする項目をユーザがリストに加える形にするのがベストでしょう。
ソースコードとバイナリが同期していないと、長い年月の間に悲惨な結果がもたらされる可能性があります。

```
# Paths which start with anything matching an entry in an IgnorePaths
# statement will be ignored.
IgnorePaths /boot/kernel/linker.hints
```

/bin や /sbin 等の特定のディレクトリをアップデートで変更しないように、これらのパスを追加してください。
このオプションは、ローカルの変更点を freebsd-update が上書きすることを防ぐ目的にも利用できます。

```
# 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
```

このオプションは、指定したディレクトリにある設定ファイルを、ローカルで変更されていない場合のみアップデートします。
ユーザがこれらのファイルを変更していると、変更されたファイルの自動アップデートは行われません。

他に、KeepModifiedMetadata という別のオプションが存在します。 このオプションは、freebsd-update がマージ中に変更点を保存するようにします。

```
# 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
```

freebsd-update がマージすべきファイルが存在するディレクトリの一覧です。
ファイルのマージのプロセスは、mergemaster(8) と同様 diff(1) パッチの連続ですが、選択肢は少なく、マージを承認するか、エディタを起動するか、freebsd-update を中断するかどうかを選んでください。 もし、心配な点があれば、/etc をバックアップしてからマージを承認してください。 mergemaster の詳細な情報については、mergemaster(8) で確認してください。

```
# Directory in which to store downloaded updates and temporary
# files used by FreeBSD Update.
# WorkDir /var/db/freebsd-update
```

ここではすべてのパッチや一次ファイルを置くディレクトリを指定しています。
バージョンをアップグレードするのであれば、この場所には少なくともギガバイトの空き容量が必要です。

```
# 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
```

このオプションを yes に設定すると、freebsd-update は Components のリストが完全に正しいと判断し、このリスト以外の変更点については取り扱いません。 freebsd-update は、効率的に Components リストに属するファイルをアップデートします。

詳細については、freebsd-update.conf(5) を参照してください。

16.2.2. セキュリティパッチの適用

FreeBSD のセキュリティパッチを適用する過程は簡単になりました。 管理者は freebsd-update を使うことで、 システムを完全にパッチがあたった状態に保つ事ができます。 FreeBSD セキュリティ勧告の詳細については、FreeBSD セキュリティ勧告 の節で説明されています。

以下のコマンドを実行すると、FreeBSD
のセキュリティパッチがダウンロードされ、インストールされます。
最初のコマンドは、未対応のパッチがあるかどうかを調べます。
もし未対応のパッチがある場合には、パッチが当てられた際に変更されるファイルのリストが作成されます。 2 番目のコマンドはパッチを適用します。

```
# freebsd-update fetch
```

```
# freebsd-update install
```

アップデートによってカーネルにパッチが適用された場合には、システムを再起動して新しいカーネルで起動する必要があります。
もし、実行中のバイナリにパッチが適用された場合には、パッチが当てられたバイナリが使われるように、影響するアプリケーションを再起動する必要があります。

通常、ユーザはシステムを再起動する必要があります。
カーネルアップデートによりシステムの再起動が必要かどうかを調べるには、freebsd-version -k と uname -r を実行してください。これら 2 つのコマンドの結果が異なる場合には、システムを再起動してください。

毎日一度アップデートがないかどうかを自動的に確認するように設定するには、以下のエントリを /etc/crontab に追加してください。

```
@daily                                  root    freebsd-update cron
```

パッチが存在すると、自動的にダウンロードされますが、適用はされません。
root宛てにメールで、ダウンロードされたパッチを確認し、freebsd-update install とともに手動でインストールする必要のあることが通知されます。

うまく行かなかった場合には、freebsd-update を以下のように実行すると、最後の変更までロールバックできます。

```
# freebsd-update rollback
Uninstalling updates... done.
```

カーネルまたはカーネルモジュールがアップデートされた場合には、
完了後にもう一度システムを再起動して、 影響のあったバイナリを再起動してください。

freebsd-update ユーティリティが自動的にアップデートするカーネルは GENERIC のみです。
カスタムカーネルをインストールしている場合には、freebsd-update によりインストールした後、カーネルを再構築し、もう一度インストールする必要があります。
デフォルトのカーネルの名前は *GENERIC* です。uname(1) コマンドを使ってインストールされているかどうかを確認できます。

GENERIC カーネルを、常に /boot/GENERIC に置いておいてください。さまざまな問題を解決する際や、バージョンをアップグレードする際に助けとなります。GENERIC カーネルを用意する方法については、FreeBSD 9.X 以降のシステムにおけるカスタムカーネル を参照してください。

/etc/freebsd-update.conf のデフォルトの設定を変更しない限り、freebsd-update は、他の更新と共にカーネルソースをアップデートします。
新しいカスタムカーネルの再構築と再インストールは、通常通り行うことができます。

freebsd-update は、常にカーネルをアップデートするとは限りません。freebsd-update install によってカーネルソースが変更されなかった場合には、カスタムカーネルを再構築する必要はありません。しかしながら freebsd-update は、/usr/src/sys/conf/newvers.sh を常にアップデートします。これは、現在のシステムのパッチレベルを uname -r が -p で表示する時にこのファイルが参照されます。

そのため、何も変更されていない場合でも、カスタムカーネルを再構築することにより、uname がシステムの正確なパッチレベルを報告するようになります。
各システムにインストールされているアップデートをすばやく把握できるようになるので、特に複数のシステムを管理するときに助けとなります。

16.2.3. メジャーおよびマイナーバージョンのアップグレード

FreeBSD のマイナーバージョン間のアップグレード、たとえば、FreeBSD 9.0 から FreeBSD 9.1 へのアップグレードは、マイナーバージョン アップグレードと呼ばれます。 メジャーバージョンアップグレードは、FreeBSD 9.X から FreeBSD 10.X へのアップグレードといった、FreeBSD のメジャーバージョンが変わるようなアップグレードのことです。 どちらのアップグレードも、freebsd-update のターゲットにリリース番号を指定する事で実行できます。

カスタムカーネルを使っているシステムでは、アップグレードを行う前に GENERIC カーネルが、/boot/GENERIC に置かれている事を確認してください。 GENERIC カーネルを用意する方法については、FreeBSD 9.X 以降のシステムにおけるカスタムカーネル を参照してください。

以下のコマンドを実行すると、FreeBSD 9.0 のシステムを FreeBSD 9.1 にアップグレードします。

```
# freebsd-update -r 9.1-RELEASE upgrade
```

コマンドを実行すると、freebsd-update は設定ファイルと現在のシステムを評価し、アップデートするために必要な情報を収集します。
画面には、どのコンポーネントが認識され、どのコンポーネントが認識されていないといったリストが表示されます。 たとえば以下のように表示されます。

```
Looking up update.FreeBSD.org mirrors... 1 mirrors found.
Fetching metadata signature for 9.0-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
```

ここで、freebsd-update はアップグレードに必要なすべてのファイルをダウンロードします。
何をインストールし、どのように進むかといった質問をされることもあります。

カスタムカーネルを使っていると、上記のステップ中に以下のような警告が表示されます。

```
WARNING: This system is running a "MYKERNEL" kernel, which is not a
kernel configuration distributed as part of FreeBSD 9.0-RELEASE.
This kernel will not be updated: you MUST update the kernel manually
before running "/usr/sbin/freebsd-update install"
```

この時点ではこの警告を無視してもかまいません。 アップデートされた GENERIC カーネルは、アップグレードプロセスの途中で利用されます。

すべてのパッチがローカルシステムへダウンロードされたら、次にパッチが適用されます。
このプロセスには時間がかかります。 この時間はコンピュータの性能およびワークロードに依存します。
その後、設定ファイルがマージされます。
このプロセスでは、ユーザはファイルをマージするか、画面上にエディタを立ち上げて手動でマージするかを尋ねられます。
プロセスが進むごとに、成功したマージのすべての結果の情報がユーザに示されます。
マージに失敗したり、無視した場合には、プロセスが中断します。 ユーザによっては /etc のバックアップを取り、master.passwd や group のような重要なファイルを後で手動でマージする方もいます。

すべてのパッチは別のディレクトリでマージされており、まだ、システムには反映されていません。
すべてのパッチが正しく適用され、すべての設定ファイルがマージされてプロセスがスムーズに進んだら、ユーザは以下のコマンドを用いて、変更点をディスクに反映してください。

```
# freebsd-update install
```

パッチは最初にカーネルとカーネルモジュールに対して当てられます。
システムがカスタムカーネルを実行している場合には、nextboot(8) を使って次回の再起動時のカーネルを、アップデートされた /boot/GENERIC に設定してください。

```
# nextboot -k GENERIC
```

GENERIC カーネルで再起動する前に、カーネルにシステムが適切に起動するために必要なすべてのドライバが含まれていること、もしアップデートしているコンピュータがリモートでアクセスしているのであれば、ネットワーク接続に必要なすべてのドライバも含まれていることを確認してください。
特に、これまで実行しているカスタムカーネルが、カーネルモジュールとして提供されているビルドインの機能を含んでいるのであれば、これらのモジュールを一時的に /boot/loader.conf の機能を用いて、GENERIC に読み込んでください。
アップグレードプロセスが終わるまでは、重要ではないサービスを無効にするとともに、必要のないディスクやネットワークのマウントなども避けることが推奨されています。

アップデートされたカーネルでコンピュータを再起動してください。

```
# shutdown -r now
```

システムがオンラインに戻ったら、以下のコマンドを使って freebsd-update を再び実行してください。
アップデートプロセスの状態は保存されているので、freebsd-update
を実行すると、古い共有ライブラリおよびオブジェクトファイルを削除するステップに進みます。

```
# freebsd-update install
```

使用しているライブラリのバージョン番号の付けられ方によって、 3
つのインストールフェーズが 2 つになる場合もあります。

アップグレードはこれで終了です。
もしメジャーアップグレードを行った場合には、メジャーバージョンアップグレード後の package のアップグレード で説明されているようにすべての ports および package を再構築してください。

FreeBSD 9.X 以降のシステムにおけるカスタムカーネル

freebsd-update を使う前に、GENERIC カーネルが /boot/GENERIC
に置かれていることを確認してください。
ただ一度だけカスタムカーネルを構築したのであれば、/boot/kernel.old は GENERIC
カーネルそのものです。 このディレクトリの名前を /boot/GENERIC へと変更してください。

もし、2
回以上カスタムカーネルを構築した後であったり、カスタムカーネルを構築した回数がわからなければ、
現在のオペレーティングシステムのバージョンの GENERIC カーネルを入手してください。
コンピュータへの物理的なアクセスが可能であれば、インストールメディアから GENERIC
カーネルをインストールできます。

```
# mount /cdrom
# cd /cdrom/usr/freebsd-dist
# tar -C/ -xvf kernel.txz boot/kernel/kernel
```

別な方法としては、 GENERIC カーネルをソースから再構築して、 インストールしてください。

```
# cd /usr/src
# make kernel __MAKE_CONF=/dev/null SRCCONF=/dev/null
```

freebsd-update がこのカーネルを GENERIC カーネルとして認識するために、GENERIC
コンフィグレーションファイルは、とにかく変更してはいけません。
また、特別なオプションを指定しないで構築してください。

freebsd-update は、/boot/GENERIC が存在する事だけを必要とするので、GENERIC
カーネルで再起動する必要はありません。

メジャーバージョンアップグレード後の **package** のアップグレード

一般的に、マイナーバージョンアップグレードの後では、インストールされているアプリケーションは、
問題なく動作するでしょう。 メジャーバージョンが異なるとアプリケーションバイナリーインタフェース
(ABI) が異なるため、サードパーティ製のアプリケーションの多くは動作しなくなるでしょう。
メジャーバージョンアップグレード後には、インストールされているすべての packages, ports

をアップグレードする必要があります。 package は、pkg upgrade を使ってアップグレードできます。インストールされている ports をアップグレードする場合には、ports-mgmt/portmaster といったユーティリティを使ってください。

すべての package の強制的なアップグレードでは、バージョン番号が上がらない package に対しても、リポジトリから最新のバージョンで、インストールされている package を置き換えます。FreeBSD のメージャーバージョンが変わるようなアップグレードでは、ABI のバージョンも変わるため、このようなアップグレードが必要になります。
強制的なアップグレードを行うには、以下のように実行してください。

```
# pkg-static upgrade -f
```

インストールされているすべてのアプリケーションを再構築するには、以下のコマンドを実行してください。

```
# portmaster -af
```

このコマンドを実行すると、設定を変更するオプションを持つアプリケーションは、設定変更のスクリーンを表示し、ユーザからの指示待ちの状態で停止します。
この振る舞いをやめ、デフォルトのオプションを使用するには、上記のコマンドに -G を含めてください。

ソフトウェアのアップグレードが終わったら、最後にもう一度 freebsd-update を実行して、すべてのアップグレードプロセスのやり残し作業を行い、アップグレードのプロセスを完了してください。

```
# freebsd-update install
```

GENERIC カーネルを一時的に読み込んでいたのであれば、FreeBSD カーネルのコンフィグレーション に書かれている手順に従って、新しいカスタムを構築し、インストールしてください。

コンピュータを再起動し、新しい FreeBSD を立ち上げてください。
これでアップグレードのプロセスは完了です。

16.2.4. システムの状態の比較

freebsd-update を用いて、インストールされている FreeBSD の状態と、正しく動作することが分かっている状態とを比較できます。
このコマンドは、現在のシステムのユーティリティ、ライブラリ、設定ファイルを評価するので、組み込みの侵入検知システム (IDS) として使うことができます。

このコマンドは、security/snort のような本当の IDS の置き換えになるものではありません。 freebsd-update はデータをディスクに保存するので、不正な変更が行われる可能性があります。 kern.securelevel と、freebsd-update のデータを使用しないときに、読み取りのみの許可属性に設定されているファイルシステムに置くことで、不正な変更の可能性を低くできますが、よりよい解決方法は、DVD または安全に保存されている外部 USB ディスクのような安全なディスクとシステムを比較することです。

> 組み込まれているユーティリティを用いた、別の方法による IDS
> 機能については、FreeBSD バイナリによる検出 の節をご覧ください。

比較を行うには、 結果の出力先のファイル名を指定してください。

```
# freebsd-update IDS >> outfile.ids
```

システムは検査され、リリースファイルの SHA256
ハッシュ値と現在インストールされているファイルのハッシュ値がファイルの一覧と共に、指定した出力先のファイルに送られます。

これらの行は極めて長いのですが、出力形式は簡単にすぐに解析できます。
たとえば、これらのリリースで異なっているすべてのファイルを知りたいのであれば、以下のコマンドを実行してください。

```
# cat outfile.ids | awk '{ print $1 }' | more
/etc/master.passwd
/etc/motd
/etc/passwd
/etc/pf.conf
```

上の表示例では出力は切り捨てられており、実際にはもっと多くのファイルが存在します。
これらのファイルには、運用中に変更されるファイルがあります。 たとえば、/etc/passwd はユーザがシステムに追加されると変更されます。 また、カーネルモジュールは、`freebsd-update` によりアップデートされるため、変更されます。
このような特別なファイルやディレクトリを除外するには、それらを /etc/freebsd-update.conf の `IDSIgnorePaths` オプションに追加してください。

16.3. ブートコードのアップデート

ブートコードおよびブートローダのアップデートプロセスについては、 gpart(8), gptboot(8), gptzfsboot(8) および loader.efi(8) のマニュアルを参照してください。

16.4. ドキュメントのアップデート

ドキュメントは、FreeBSD オペレーティングシステムの必須要素です。 FreeBSD ドキュメントの最新バージョンは、FreeBSD ウェブサイト (Documentation Portal) から入手できますが、 FreeBSD ウェブサイト、ハンドブック、FAQ および文書の最新版をローカルに用意しておくと便利です。

この章では、ソースまたは Ports Collection を使って、ローカルの FreeBSD ドキュメントを最新に保つ方法を説明します。

ドキュメントを編集したり、ドキュメントの誤りを報告する方法については、新しい貢献者のための FreeBSD ドキュメンテーションプロジェクト入門 (FreeBSD Documentation Project Primer) をご覧ください。

16.4.1. ソースから FreeBSD ドキュメントをインストールする

ソースから FreeBSD ドキュメントを構築するのに必要なツールは、FreeBSD のベースシステムには含まれていません。 必要なツールは、新しい貢献者のための FreeBSD ドキュメンテーションプロジェクト入門で 説明されているステップ に従ってインストールしてください。

インストールしたら、git を使って、ドキュメントのソースをダウンロードしてください。

```
# git clone https://git.FreeBSD.org/doc.git /usr/doc
```

最初にドキュメントのソースをダウンロードするには少し時間がかかります。
ダウンロードが終わるまでお待ちください。

ダウンロードしたドキュメントのソースをアップデートするには、 以下のコマンドを実行してください。

```
# git pull
```

最新のドキュメントのソースのスナップショットを /usr/doc に用意できたら、インストールされているドキュメントをアップデートする準備はすべて整いました。

ドキュメントをアップデートするには、以下のように入力してください。

```
# cd /usr/doc
# make
```

16.5. 開発ブランチを追いかける

FreeBSD には二つの開発ブランチがあります。 それは FreeBSD-CURRENT と FreeBSD-STABLE です。

この節ではそれぞれのブランチと対象としている読者についての説明と、
どのようにしてシステムの対応するブランチを最新の状態に保つかについて説明します。

16.5.1. FreeBSD-CURRENT を使う

FreeBSD-CURRENT とは FreeBSD の開発の "最前線" なので、FreeBSD-CURRENT のユーザは高い技術力を持つことが要求されます。
そこまでの技術力を持っていないが、開発ブランチを追いかけたいと考えているユーザは、かわりに FreeBSD-STABLE を追いかけると良いでしょう。

FreeBSD-CURRENT は FreeBSD の最新のソースコードであり、中には現在開発途上のソフトウェア、実験的な変更、あるいは過渡的な機能などが含まれています。
また、この中に入っている機能がすべて、次の公式リリースに入るとは限りません。 FreeBSD-CURRENT をソースからほぼ毎日コンパイルしている人はたくさんいますが、短い期間ではコンパイルさえできない状態になっている時期もあります。 これらの問題は可能な限り迅速に解決されますが、FreeBSD-CURRENT が不幸をもたらすか、それとも新しい機能をもたらすかは、まさにソースコードを同期した瞬間によるの

です!

FreeBSD-CURRENT は、次の 3 つの重要なグループを対象としています。

1. ソースツリーのある部分に関して活発に作業している FreeBSD コミュニティのメンバ。
2. 活発にテストしている FreeBSD コミュニティのメンバ。
   彼らは、種々の問題を解決するのに時間を惜しまない人々であり、 さまざまな変更に関する提案や FreeBSD の大まかな方向付けを行ないたいと思っている人々でもあり、 パッチも提出します。
3. さまざまな事に目を向け、 参考のために最新のソースを使いたいと思っていたり、
   時々コメントやコードを寄稿したいと考えているユーザ。

FreeBSD-CURRENT
は、次のリリースの前に、最も早く新しい機能を入手する手段として、期待してはいけません。
リリース前の機能は十分にテストされていないため、バグを含んでいる可能性が大いにあるためです。
また、バグを修正するための素早い方法でもありません。
いかなるコミットは、元からあるバグを修正するのと同じく、新しいバグを生み出すおそれがあります。
FreeBSD-CURRENT には "公式のサポート" はありません。

FreeBSD-CURRENT を追いかけるには

1. FreeBSD-CURRENT メーリングリスト と Commit messages for the main branch of the src repository メーリングリストに加わってください。
   さまざまな人がシステムの現在の状態について述べているコメントを見たり、FreeBSD-CURRENT の現在の状態に関する重要な情報を見逃さないために、 必須の ことです。

   Commit messages for the main branch of the src repository メーリングリストでは、それぞれの変更についての commit ログが記録されています。
   また、それに関して起こり得る副作用の情報を得ることができますので、参加する価値のあるメーリングリストです。

   これらのメーリングリストに入るには、 FreeBSD リストサーバ をたどって参加したいメーリングリストをクリックし、手順の説明にしたがってください。 FreeBSD-CURRENT だけでなく、ソースツリー全体の変更点を追いかけるのであれば、 Commit messages for all branches of the src repository メーリングリストを購読してください。

2. FreeBSD-CURRENT のソースを同期してください。 通常は `git` を使って FreeBSD Git リポジトリの `main` ブランチから -CURRENT コードをチェックアウトしてください (「Git を使う」を参照してください)。
3. リポジトリのサイズが大きいため、興味のある部分や、パッチを当てる部分のソースのみを同期するユーザもいます。
   しかしながら、ソースからオペレーティングシステムをコンパイルしようと思っているユーザは、一部分だけではなく、FreeBSD-CURRENT の すべて をダウンロードする必要があります。

   FreeBSD-CURRENT をコンパイルする前に /usr/src/Makefile を注意深く読み、ソースを用いた FreeBSD のアップデート に書かれている手順に従ってください。 FreeBSD-CURRENT メーリングリスト と /usr/src/UPDATING を読めば、次のリリースへ向けて移ってゆくに当たって、ときどき必要となる既存システムからの新システムの構築手順についての最新情報が得られるでしょう。

4. アクティブになってください! FreeBSD-CURRENT のユーザには、

拡張やバグ潰しに関して提案することが勧められています。
コードを伴う提案はいつでも歓迎されます!

16.5.2. FreeBSD-STABLE を使う

FreeBSD-STABLE とは定期的に公開されるリリースを作成するための開発ブランチです。このブランチに加えられる変更は FreeBSD-CURRENT よりゆっくりで、原則として、事前に FreeBSD-CURRENT で試験ずみであるという特徴があります。ただそうであっても、これは開発用ブランチの一つであり、ある時点における FreeBSD-STABLE のソースがどんな場合にも使えるものであるとは限りません。
このブランチはもう一つの開発の流れというだけであって、エンドユーザ向けのものではありません。
もし試験をする資源的な余裕がない場合は、代わりに最新の FreeBSD リリースを使ってください。

FreeBSD の開発プロセスに興味があったり、それに対する貢献を考えていて、特にそれが次回の FreeBSD のリリースに関係するものであるなら FreeBSD-STABLE を追うことを考えると良いでしょう。

FreeBSD-STABLE ブランチはいつもコンパイルができ、安定に動作すべきですが、それが保証されているというわけではありません。FreeBSD-STABLE のユーザは FreeBSD-CURRENT よりも多いため、FreeBSD-CURRENT で発見されなかったバグが FreeBSD-STABLE で発見され、ときどきそれが問題となることがあるのは避けることができません。
このような理由から、盲目的に FreeBSD-STABLE を追いかけるべきではありません。
特に、開発環境もしくはテスト環境でコードを十分に試験せずに、プロダクション品質が要求されるサーバを FreeBSD-STABLE にアップグレードしてはいけません。

FreeBSD-STABLE を追いかけるには

1. FreeBSD-STABLE の構築に関連する事柄や、その他の注意すべき点に関する情報を得るために、FreeBSD-STABLE; メーリングリスト メーリングリストに加わってください。
   また開発者は議論の余地がある修正や変更を考えている場合に、このメーリングリストで公表し、提案された変更に関して問題が生じるかどうかを返答する機会をユーザに与えます。

   追いかけているブランチに関連する git メーリングリストに参加してください。たとえば、14-STABLE ブランチを追いかけているユーザは Commit messages for the stable branches of the src repository メーリングリストに参加してください。
   このリストでは、変更がなされるごとに作成される commit log やそれに伴う起こりうる副作用についての情報が記録されています。

   これらのメーリングリストに入るには、FreeBSD リストサーバをたどって参加したいメーリングリストをクリックし、手順の説明にしたがってください。
   ソースツリー全体の変更点を追いかけるには、Commit messages for all branches of the src repository メーリングリストを購読してください。

2. 新しい FreeBSD-STABLE システムをインストールするには、ミラーサイト から最近の FreeBSD-STABLE リリースをインストールするか、毎月公開されている FreeBSD-STABLE からビルドされたスナップショットを使ってください。
   スナップショットの詳細については、www.freebsd.org/ja/snapshots をご覧ください。

   既に FreeBSD が動いているシステムを FreeBSD-STABLE にアップグレードするには、`git` を使って、希望する開発ブランチのソースをチェックアウトしてください。`stable/9`

といったブランチ名は、www.freebsd.org/releng で説明されています。

3. FreeBSD-STABLE をコンパイルしたり FreeBSD-STABLE へとアップグレードする前に、/usr/src/Makefile を注意深く読み、ソースを用いた FreeBSD のアップデートに書かれている手順に従ってください。FreeBSD-STABLE; メーリングリスト と /usr/src/UPDATING を読んで、次のリリースへ向けて移ってゆくに当たって、ときどき必要となる既存システムからの新システムの構築手順についての最新情報を得てください。

16.5.3. n-番号

バグを追跡する際は、問題が発生したシステムの構築に用いられたソースコードのバージョンを把握することが重要となります。 FreeBSD は、バージョン情報をカーネルのコンパイル時に埋め込みます。uname(1) を使ってこの情報を調べることができます。以下はその例です。

```
% 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
```

最後のフィールドから、カーネル名、ビルドを行ったユーザ、およびコンパイルを行った場所がわかります。 また、4 番目のフィールドは、いくつかの要素から構成されていることがわかります。

```
main-n247514-031260d64c18

main        ①
n247514     ②
031260d64c18    ③
④
```

① Git ブランチ名。 注意: n-番号の比較は、FreeBSD プロジェクトで作成されたブランチ (`main`, `stable/XX` および `releng/XX`) でのみ有効です。 ローカルブランチでは、親ブランチのコミットと n-番号が重複してしまいます。

② n-番号は、ハッシュ値が含まれるようになった git リポジトリの使用開始からのコミットを数えたものです。

③ チェックアウトしたツリーのハッシュ値。

④ `-dirty` が表示されることがあります。変更点がコミットされていないツリーでカーネルが構築された場合に表示されます。この例では、チェックアウトから変更なく FRED カーネルが構築されたため、出力されていません。

`git rev-list` コマンドを使って、ハッシュ値に対応する n-番号を調べることができます。以下はその例です。

```
% git rev-list --first-parent --count 031260d64c18 ①
247514 ②
```

① 変換する git ハッシュ値 (ここでは先の例のハッシュ値を使用しています)

② n-番号

この数字は通常それほど重要ではありません。
しかしながら、バグの修正がコミットされた時には、この数字を使うことで、使用しているシステムでバグが修正されているかどうかを簡単に調べることができます。
ハッシュ値は簡単に目にする識別子である一方で n-番号はそうではありません。 そのため、開発者は通常 n-番号ではなくコミットのハッシュ値 (または、ハッシュ値を含む URL) を参照します。
セキュリティ勧告および errata 情報では n-番号が示されており、使用しているシステムの番号と直接比較できます。 `git rev-list` コマンドは、レポジトリのリビジョンをすべてカウントしますが、git の shallow clone はその情報を取得しないため、shallow clone を使用しなければならない場合には、n-番号は信頼できません。

16.6. ソースを用いた FreeBSD のアップデート

ソースをコンパイルしてFreeBSD をアップデートする方法は、バイナリを用いたアップデートに比べ、いくつもの利点があります。
特定のハードウェアをうまく利用するためのオプションを設定してコードを構築できます。
ベースシステムの特定の箇所の設定をデフォルトの設定から変更したり、必要がない部分を完全に削除して構築することもできます。
システムを構築することによるアップデートは、バイナリアップデートをインストールするだけのアップデートに比べ時間がかかりますが、利用環境に合わせた FreeBSD を作成するような完全なカスタマイズが可能です。

16.6.1. クィックスタート

以下は FreeBSD をソースから構築してアップデートする典型的な方法についてのクイックリファレンスです。
その後の節では、各プロセスをより詳細に説明します。

mergemaster(8) から etcupdate(8) に移行する際に、初めて etcupdate(8) を実行すると、変更点が不適切にマージされ、衝突が起きてしまうことがあります。
これを避けるには、ソースを更新して新しく buildworld を行う前に以下のステップを行ってください。

```
# etcupdate extract ①
# etcupdate diff ②
```

① /etc ファイルを保存するデータベースをブートストラップしてください。
詳細については、 etcupdate(8) を参照してください。

② ブートストラップ後、差分を確認してください。
不必要なローカルでの変更点をなくし、将来的なアップデートにおいて、衝突が起きる可能性が低くなるようにしてください。

- アップデートおよびビルド

```
# git pull /usr/src  ①
/usr/src/UPDATING の確認 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 最新版のソースを入手してください。 ソースの入手およびアップデートに関する情報については ソースコードのアップデート をご覧ください。

② ソースの構築の前後で必要となる手動の作業について、 /usr/src/UPDATING を確認してください。

③ ソースが置かれているディレクトリに移動してください。

④ world (カーネルを除くすべて) をコンパイルしてください。

⑤ カーネルをコンパイルしてインストールしてください。 ここに書かれているコマンドは、`make buildkernel installkernel` と同じです。

⑥ 新しいカーネルを使うため、 システムを再起動してください。

⑦ installworld を行う前に、/etc/ に置かれている設定ファイルのアップデートとマージを行ってください。

⑧ ソースが置かれているディレクトリに移動してください。

⑨ world をインストールしてください。

⑩ /etc/ に置かれている設定ファイルのアップデートとマージを行ってください。

⑪ 新しく構築された world およびカーネルを利用するため、 システムを再起動してください。

16.6.2. ソースを用いたアップデートのための準備

/usr/src/UPDATING を読んでください。 このファイルには、アップデートの前後で必要となる手動の作業について書かれています。

16.6.3. ソースコードのアップデート

FreeBSD のソースコードは /usr/src/ に置かれています。 このソースコードのアップデートには、Git バージョン管理システムを利用する方法が推奨されています。
まず、ソースコードがバージョン管理下にあることを確認してください。

```
# cd /usr/src
# git remote --v
origin  https://git.freebsd.org/src.git (fetch)
origin  https://git.freebsd.org/src.git (push)
```

この結果は、/usr/src/ がバージョン管理下にあり、git(1) を使ってアップデートできることを示しています。

```
# git pull /usr/src
```

このディレクトリをアップデートしていない期間が長いと、アップデートのプロセスには時間がかかります。
このプロセスが終わると、ソースコードは最新となり、次節以降で説明する構築のプロセスを実行できます。

ソースコードの入手

`fatal: not a git repository` と出力された場合には、ファイルがなかったり、別な方法によりインストールされているので、新しくソースコードをチェックアウトする必要があります。

表 *17. FreeBSD* のバージョンおよびリポジトリブランチ

uname -r の出力	リポジトリパス	説明
`X.Y-RELEASE`	`releng/X.Y`	このリリースバージョンに対する重大なセキュルティへの対応およびバグの修正パッチのみが適用されています。 このブランチは、ほとんどのユーザに推奨されます。
`X.Y-STABLE`	`stable/X`	リリースバージョンに対し、そのブランチにおけるすべての開発の成果が反映されたものです。 *STABLE* では、Applications Binary Interface (ABI) は変更されないため、このブランチのシステムであれば、以前のバージョンでコンパイルされたソフトウェアを実行できます。 たとえば、FreeBSD 10.1 で実行するようにコンパイルされたソフトウェアは、その後構築された FreeBSD 10-STABLE 上でも実行できます。 STABLE ブランチは、時期によってはユーザに影響するようなバグや非互換性を持つことがあります。これらは通常すぐに修正されます。

uname -r の出力	リポジトリパス	説明
X-CURRENT	main	リリースが行われていない最新の FreeBSD の開発バージョンです。CURRENT ブランチは大きなバグや非互換があることもあるので、高度な知識を持ったユーザのみ使用が推奨されます。

uname(1) を使って FreeBSD のバージョンを確認してください。

```
# uname -r
10.3-RELEASE
```

FreeBSD のバージョンおよびリポジトリブランチ から分かるように、10.3-RELEASE のアップデートのためのソースコードのパスは、releng/10.3 です。
このパスは、ソースコードをチェックアウトする時に使います。

```
# mv /usr/src /usr/src.bak ①
# git clone --branch releng/10.3 https://git.FreeBSD.org/src.git
/usr/src ②
```

① この古いディレクトリを、邪魔にならないように移動してください。
このディレクトリ以下に対して変更を行ってなければ、
削除しても構わないでしょう。

② リポジトリの URL に FreeBSD のバージョンおよびリポジトリブランチ に記載されているパスを追加します。3 番目のパラメータには、
ローカルシステム上でソースコードが置かれるディレクトリを指定します。

16.6.4. ソースからの構築

まず最初に *world* (カーネルを除くオペレーティングシステムのすべて) をコンパイルします。
このステップを最初に実行するのは、カーネルの構築を最新のツールを使って行うようにするためです。
このステップが終わったら、カーネルそのものを構築します。

```
# cd /usr/src
# make buildworld
# make buildkernel
```

コンパイルされたコードは /usr/obj に書き出されます。

これは基本のステップです。構築をコントロールする追加のオプションについては、
以下で説明します。

クリーンビルドの実行

FreeBSD ビルドシステムのいくつかのバージョンは、オブジェクトが一時的に置かれるディレクトリ /usr/obj に前回のコンパイルされたコードを残します。
これにより、変更されていないコードを再コンパイルせずにすむので、その後の構築時間を短縮できます。 すべてを再構築するには、構築を開始する前に、`cleanworld` を実行してください。

```
# make cleanworld
```

ジョブの数の設定

マルチコアプロセッサを搭載するシステムでは、構築のためのジョブの数を増やすことで、構築にかかる時間を短縮できます。 `sysctl hw.ncpu` を使って、コアの数を確認してください。
ジョブの数がどのように構築の速さに影響するかを確実に知るには、プロセッサにより異なりますし、FreeBSD
のバージョンにより使用されるビルドシステムも変わるため、実際に試してみるしか方法はありません。
試してみる最初のジョブの数の候補としては、コアの数の半分から倍の数の間で検討してみてください。
ジョブの数は、`-j` を使って指定します。

例 39. 構築のジョブの数を増やす

以下は 4 つのジョブで world とカーネルを構築する例です。

```
# make -j4 buildworld buildkernel
```

カーネルのみを構築する

ソースコードが変更された場合には、`buildworld` を完了しなければいけません。 その後、いつでも `buildkernel` でカーネルを構築できます。
カーネルだけを構築するには、以下のように実行してください。

```
# cd /usr/src
# make buildkernel
```

カスタムカーネルの構築

FreeBSD 標準のカーネルは、GENERIC と呼ばれる カーネルコンフィグレーションファイルに基づいています。 GENERIC
カーネルには、最も良く使われるデバイスドライバやオプションが含まれています。
しかしながら、特定の目的に合わせてデバイスドライバやオプションを削除したり追加するためには、カスタムカーネルを構築することが有用であったり、必要となることがあります。

たとえば、極端に RAM
が制限されているような小さな組み込みのコンピュータを開発しているユーザであれば、
必要のないデバイスドライバやオプションを削除することで、
カーネルを少しでも小さくできるでしょう。

カーネルのコンフィグレーションファイルは、 /usr/src/sys/arch/conf/ に置かれています。ここで、 *arch*

は uname -m の出力です。 ほとんどのコンピュータは amd64 であり、コンフィグレーションファイルが置かれているディレクトリは /usr/src/sys/amd64/conf/ です。

/usr/src は、削除されたり作り直されたりする可能性があるため、カスタムカーネルのコンフィグレーションファイルは、/root のような別のディレクトリで管理することが好ましいです。カーネルコンフィグレーションファイルは、conf ディレクトリにリンクします。このディレクトリが削除されたり、上書きされた場合には、カーネルコンフィグレーションファイルを新しいディレクトリにもう一度リンクしてください。

カスタムコンフィグレーションファイルは、GENERIC コンフィグレーションファイルをコピーして作成できます。 たとえば、ストレージサーバ用の STORAGESERVER という名前の新しいカスタムカーネルは、以下のようにして作成できます。

```
# cp /usr/src/sys/amd64/conf/GENERIC /root/STORAGESERVER
# cd /usr/src/sys/amd64/conf
# ln -s /root/STORAGESERVER .
```

その後 /root/STORAGESERVER を編集し、 config(5) で示されるデバイスやオプションを追加したり削除してください。

コマンドラインからカーネルコンフィグレーションファイルを KERNCONF に指定することで、カスタムカーネルを構築できます。

```
# make buildkernel KERNCONF=STORAGESERVER
```

16.6.5. コンパイルされたコードのインストール

buildworld および buildkernel が完了したら、 新しいカーネルと world をインストールしてください。

```
# cd /usr/src
# make installkernel
# shutdown -r now
# cd /usr/src
# make installworld
# shutdown -r now
```

カスタムカーネルを構築した場合は、 新しいカスタムカーネルを KERNCONF に設定して実行してください。

```
# cd /usr/src
# make installkernel KERNCONF=STORAGESERVER
# shutdown -r now
# cd /usr/src
# make installworld
```

```
# shutdown -r now
```

16.6.6. アップデートの完了

アップデートの完了までに、いくつかの最終作業が残されています。
デフォルトから変更した設定ファイルを新しいバージョンのファイルにマージし、古くなったライブラリを見つけて削除した後に、システムを再起動します。

etcupdate(8) を用いた設定ファイルのマージ

etcupdate(8) は、/etc/ 以下のファイルのように installworld のプロセスで更新されないファイルをアップデートするツールです。
このツールは、ローカルにあるファイルに対する変更点を 3-way マージでアップデートします。
mergemaster(8)
の対話的なプロンプトと対照的に、このツールはユーザによる操作を最小限になるように設計されています。

一般的に、etcupdate(8) は、実行する際に特定の引数を必要としません。
しかしながら、etcupdate(8)
を最初に使用した際に、どのようなアップデートが行われたかの健全性をチェックする便利なコマンドがあります。

```
# etcupdate diff
```

このコマンドにより、ユーザは設定の変更を検証できます。

etcupdate(8) が自動的にファイルをマージできない場合には、以下を実行することで、手動の操作により衝突を解決できます。

```
# etcupdate resolve
```

mergemaster(8) から etcupdate(8) に移行する際に、最初に etcupdate(8) を実行すると、不適切に変更点がマージされ、誤った衝突が起こる可能性があります。
これを避けるには、ソースを更新して新しく buildworld を行う前に以下のステップを行ってください。

```
# etcupdate extract ①
# etcupdate diff ②
```

① /etc ファイルを保存するデータベースをブートストラップしてください。
詳細については、 etcupdate(8) を参照してください。

② ブートストラップ後、差分を確認してください。
不必要なローカルでの変更点をなくし、将来的なアップデートにおいて、衝突が起きる可能性が低くなるようにしてください。

mergemaster(8) を用いた設定ファイルのマージ

mergemaster(8)
を用いることで、システムの設定ファイルに行われている変更を、これらのファイルの新しいバージョンにマージできます。 mergemaster(8) は、設定ファイルのアップデートで推奨されている etcupdate(8) の代価のツールです。 -Ui オプションを使って mergemaster(8) を実行すると、ユーザが手を加えていないファイルのアップデートおよび新しく追加されたファイルのインストールを自動的に行います。

```
# mergemaster -Ui
```

ファイルのマージを手動で行う必要がある時は、ファイルの中で残す箇所の選択を対話的におこなうようなインタフェースが表示さます。 詳細については、mergemaster(8) をご覧ください。

使われなくなったファイルやライブラリの確認

アップデート後に、使われなくなったファイルやディレクトリが残ることがあります。
これらのファイルは、

```
# make check-old
```

で確認でき、以下のようにして削除できます。

```
# make delete-old
```

同様に使われなくなったライブラリが残ることもあります。 これらのライブラリは、

```
# make check-old-libs
```

で確認でき、以下のようにして削除できます。

```
# make delete-old-libs
```

これらの古いライブラリを利用しているプログラムは、ライブラリが削除されると動かなくなります。
これらのプログラムは、古いライブラリを削除した後に、再構築もしくは置き換える必要があります。

古いファイルとディレクトリのすべてを削除しても問題ないことを確認したら、コマンドに BATCH_DELETE_OLD_FILES を設定することで、各ファイルを削除する際に y および Enter を押さなくても済むようにできます。 以下はその例です。

```
# make BATCH_DELETE_OLD_FILES=yes delete-old-libs
```

アップデート後の再起動

コンピュータを再起動して、すべての変更を反映させることが、
アップデートの最後におこなう作業です。

```
# shutdown -r now
```

16.7. 複数のマシンで追いかける

複数のコンピュータで同じソースツリーを追いかけていて、全部のマシンにソースをダウンロードして全部を再構築するのは、ディスクスペース、ネットワーク帯域、そして CPU サイクルの無駄使いです。解決策は 1 つのマシンに仕事のほとんどをさせ、残りのマシンは NFS 経由でそれをマウントする、というものです。 このセクションではそのやり方を概観します。NFS の使い方の詳細については、「NFS」をご覧下さい。

まず初めに、同じバイナリで動かそうとするマシンたちを決めます。
このマシンたちのことをビルドセットと呼びます。
それぞれのマシンはカスタムカーネルを持っているかもしれませんが、同じユーザランドバイナリを動かそうというのです。 このビルドセットから、 ビルドマシンとなるマシンを 1 台選びます。ベースシステムとカーネルを構築するのはこのマシンになります。 理想的には、このマシンは `make buildworld` と `make buildkernel` を実行するのに十分な CPU を持った速いマシンであるべきです。

テストマシン となるべきマシンも選んでください。
更新されたソフトウェアを使う前にそのマシンでテストするのです。
テストマシンはかなり長い時間落ちていても だいじょうぶなマシンであったほうがいいでしょう。ビルドマシンでもかまいませんが、ビルドマシンである必要はありません。

このビルドセットのマシンはすべて /usr/obj と /usr/src をビルドマシンから FTP 経由でマウントする必要があります。 ビルドセット自体が複数ある場合は、/usr/src はひとつのビルドマシン上にあるべきです。 他のマシンからはそれを NFS マウントするようにしましょう。

ビルドセットのすべてのマシン上の /etc/make.conf と /etc/src.conf がビルドマシンと一致していることを確認してください。
つまり、ビルドマシンはビルドセットのどのマシンもインストールしようとしているベースシステムを全部ビルドしなければならないということです。 また、各ビルドマシンは /etc/make.conf にそれぞれのビルドマシンのカーネル名を `KERNCONF` で指定し、ビルドマシンは自分自身のカーネルから順に全部のカーネル名を `KERNCONF` にリストアップしてください。 ビルドマシンは各マシンのカーネル設定ファイルを /usr/src/sys/arch/conf に持っていなければなりません。

ビルドマシンにて、ソースを用いた FreeBSD のアップデートに書いてあるようにカーネルとベースシステムを構築してください。
でも、まだビルドマシンにはインストールしないでください。
そのかわり、ビルドしたカーネルをテストマシンにインストールしてください。 FTP 経由で /usr/src および /usr/obj をテストマシンにマウントしてください。 その後、`shutdown now` を実行してシングルユーザモードに移行し、新しいカーネルとベースシステムをインストールし、いつもするように `mergemaster` を実行してください。
終わったら、再起動して通常のマルチユーザ動作に戻します。

テストマシンにあるものすべてがちゃんと動いている確信が得られたら、同じ手順でビルドセットの他のマシンにも新しいソフトウェアをインストールします。

ports ツリーにも同じ方法が使えます。 最初のステップは、ビルドセットのすべてのマシンが NFS 経由で /usr/ports をマウントすることです。 そして、distfiles を共有するように /etc/make.conf を設定します。

NFS マウントによってマップされる root ユーザが何であれ、DISTDIR はそのユーザが書き込める共通の共有ディレクトリに設定する必要があります。 ports をローカルでビルドする場合には、各マシンは WRKDIRPREFIX を自分のマシンのビルドディレクトリに設定しなければなりません。 また、ビルドシステムが packages をビルドしてビルドセットのコンピュータに配布するのであれば、DISTDIR と同じようにビルドシステム上の PACKAGES ディレクトリも設定してください。

Part IV: ネットワーク通信

FreeBSD は、 高性能なネットワークサーバとして最も広く使用されているオペレーティングシステムの 1 つです。 各章の内容は以下の通りです。

- シリアル通信
- PPP と PPP オーバイーサネット (PPPoE)
- 電子メール
- ネットワークサーバの運用
- ファイアウォール
- その他の高度なネットワークに関する話題

各章は、必要になった時に個別に参照できるように構成されています。
どの順番で読んでも構いませんし、ネットワーク環境で FreeBSD を使うのに、すべてを読み通す必要がある、というわけでもありません。

Chapter 17. シリアル通信

17.1. この章では

Unix は現在に至るまで、常にシリアル通信機能をサポートしていました。 実際、本当に初期の Unix マシンは、ユーザとの入出力にシリアル通信を使っていました。 10 文字毎秒のシリアルプリンタ、キーボードから構成された "端末(terminal)" が広く使われていた当時とは、何もかもがすっかり変わっています。この章では、FreeBSD でシリアル通信を行なういくつかの方法について説明しています。

この章を読むと、以下のことがわかります。

- FreeBSD システムへの端末の接続方法
- リモートホストへダイヤルするためのモデムの使い方
- リモートのユーザがモデムでシステムにログインできるようにする方法
- シリアルコンソールからのシステム起動方法

この章を読む前に、以下のことを行っておくべきです。

- 新しいカーネルを構成してインストールする方法を覚える (FreeBSD カーネルのコンフィグレーション)。
- Unix のパーミッションとプロセスについて理解する (UNIX の基礎知識)。
- FreeBSD で使おうとしているシリアルハードウェア (モデムまたはマルチポートカード) のテクニカルマニュアルを読めるようにする。

17.2. はじめに

17.2.1. 用語解説

bps

Bits per Second の略で、 データの転送速度を表す単位。

DTE

Data Terminal Equipment の略。 たとえばコンピュータ本体のこと

DCE

Data Communications Equipment の略で、具体的にはモデムのこと。

RS-232

EIA (米電気産業協会) のハードウェアシリアル通信の標準規格

通信におけるデータ転送速度に関して、 このセクションでは "ボー" (baud) という用語は使いません。ボーというのは一定時間に生じうる電気的状態の変化の数を表すにすぎず、 "bps" (bits per second) という単位の方が正しいからです (少なくとも、こういう表現をしておけば、意地の悪い人に怒られることもないのではないかと思います)。

17.2.2. ケーブルとポート

モデムまたはシリアル端末を FreeBSD システムに接続するためには、
コンピュータ上のシリアルポートと、 シリアルデバイスに接続する適切なケーブルが必要です。
ハードウェアとそれが必要とするケーブルについてよく理解しているなら、
この節は飛ばしても問題ありません。

ケーブル

シリアルケーブルにはさまざまな種類があります。 我々の目的にあうもっとも一般的な 2 種類は、
ヌルモデムケーブル と、スタンダード (ストレート) RS-232 ケーブルです
ハードウェアの説明文書に必要なケーブルの種類が記載されているはずです。

ヌルモデムケーブル

ヌルモデムケーブル (またはリバースケーブルあるいはクロ スケーブル) は、たとえば "signal ground"
信号のように、いくつかの信 号はそのまま通しますが、
他の信号は途中で入れ替えて通します。たとえば、"send data" 信号のピンは、反対側のコネクタの
"receive data" 信号の ピンと繋がっています。

自分で使うケーブルは自分で作りたいということであれば、
端末で使うヌルモデムケーブルを作成できます。この表では、 RS-232C の信号線の名前と、DB-25
コネクタ上のピンの番 号を示しています。

Signal	Pin #		Pin #	Signal
TxD	2	connects to	3	RxD
RxD	3	connects to	2	TxD
DTR	20	connects to	6	DSR
DSR	6	connects to	20	DTR
SG	7	connects to	7	SG
DCD	8	connects to	4	RTS
RTS	4		5	CTS
CTS	5	connects to	8	DCD

DCD と RST では、コネクタ内部でピン4を5に接続し、 そして逆側のコネクタのピン
8と接続します。

スタンダード **RS-232C** ケーブル

スタンダードシリアルケーブル (またはストレートケーブル) の場合は、すべての RS-232C
信号をそのまま通します。つまり、片方の "send data" 信号のピンは、逆側の "send data"
信号のピンと繋がっています。モデムを FreeBSD
に接続するときや、一部の端末を接続するときにこのタイプの ケーブルを使用します。

ポート

シリアルポートは、FreeBSDが動作しているホスト コンピュータと端

末の間でデータのやりとりを行うために用いるデバイスです。　ここでは、現在存在するポートの種類と FreeBSD でのポートのアクセス方法について解 説します。

ポートの種類

シリアルポートには何種類かのものがあります。　ケーブルを購　入したり自作したりする前に、そのケーブルのコネクタの形状が端末および　FreeBSD　システムのポートの形状と一致していることを確認してください。

ほとんどの端末は DB25 ポートを搭載しています。 FreeBSDが動作しているも のを含めて、PCは DB25 または　DB9　ポートを搭載しています。マルチポート　のシリアルカードの場合は、RJ-12　や　RJ-45 のポートを搭載しているかもし れません。

利用されているポートの種類に関しては、 ハードウェアについてきたドキュメントを参照してください。また、多くの場合、ポートの形状から判断することもできるでしょう。

ポートの名前

FreeBSDでは、/dev　ディレクトリ内のエントリを介してシリアルポートへのアクセスがおこなわれます。 2種類の異なったエン トリがあります。

- 着信用のポートの名前は、　/dev/ttydN　(*N*　は　0から始まるポート番号)　となっています。一般に端末の接続には　着信用ポートを用います。着信用のポートでは、シリアルラインのデータ キャリア検出 (DCD) 信号がオンになっている必要があります。
- 発信用のポートの名前は、　/dev/cuaaN　となっています。発信用のポートは普通モデムの接続に用い、端末の接続には　利用しません。ただ、ケーブルまたは端末がキャリア検出信号を使えない　タイプのものの場合は、発信用のポートを使うとよいでしょう。

たとえば、端末を一つ目のシリアルポート　(MS-DOS　でいうところの　COM1)　に接続したとすると、/dev/ttyd0 がこの端末を指すことになります。また、 二つ目のシリアルポート (COM2) ならば /dev/ttyd1 となり、 以下この形式のデバイスエントリを使います。

17.2.3. カーネルの設定

デフォルトでは、FreeBSD は 4 つのシリアルポートに対応しています。MS-DOS の世界では、 COM1, COM2, COM3 および COM4 と呼ばれています。 FreeBSD では、現在のところ BocaBoard の 1008 や 2016 などの、 "単純な"マルチポートシリアルインタフェースや、 Digiboard や Stallion Technologies が製造しているよりインテリジェントなマルチポートカードにも対応しています。
しかしながら、デフォルトのカーネルは、標準の COM ポートしか見ません。

搭載されているシリアルポートのいずれかを、　カーネルが認識しているかどうか確認したい場合は、カーネルの起動時のメッセージを注意深く見るか、あるいは　/sbin/dmesg　コマンドを使って、起動時の出力メッセージを確認してください。特に、 sio で始まるメッセージをよく見てください。

以下のコマンドで sio という文字列を含むメッセージだけを表示できます。

```
# /sbin/dmesg | grep 'sio'
```

たとえば、シリアルポートを四つ持つシステムの場合は、
以下のようなシリアルポートに関するメッセージがカーネルによって表示されます。

```
sio0 at 0x3f8-0x3ff irq 4 on isa
sio0: type 16550A
sio1 at 0x2f8-0x2ff irq 3 on isa
sio1: type 16550A
sio2 at 0x3e8-0x3ef irq 5 on isa
sio2: type 16550A
sio3 at 0x2e8-0x2ef irq 9 on isa
sio3: type 16550A
```

もし、カーネルに正常に認識されないポートがある場合は、 おそらくカスタマイズした FreeBSD カーネルを構築する必要があるでしょう。 カーネルコンフィグレーションの詳細については FreeBSD カーネルのコンフィグレーション をご覧ください。

カーネルコンフィグレーションの該当するデバイス行は、 次のようになります。

```
device      sio0    at isa? port "IO_COM1" tty irq 4 vector siointr
device      sio1    at isa? port "IO_COM2" tty irq 3 vector siointr
device      sio2    at isa? port "IO_COM3" tty irq 5 vector siointr
device      sio3    at isa? port "IO_COM4" tty irq 9 vector siointr
```

システムに搭載されていないデバイスに関する記述は、
コメントアウトまたは削除してしまってかまいません。 sio(4) のマニュアルを見て、
マルチポートのボードのためのコンフィグレーションファイルの記述の仕方を確認してください。
デバイスのフラグの指定方法がバージョンによって異なりますので、 別のバージョンの FreeBSD で利用していたコンフィグレーションファイルを流用する場合には十分注意してください。

なお、port "IO_COM1", IO_COM2, IO_COM3 および IO_COM4 は、それぞれのポートの一般的なアドレスである 0x3f8, 0x2f8, 0x3e8 および 0x2e8 を表します。また、割り込み番号 4, 3, 5 と 9 は、それぞれ COM1: から COM4: のポートで一般的に使用される IRQ です。また、ISA バスのコンピュータの場合、一般的なシリアルポートは複数のポートで一つの IRQ を共有することができませんので注意が必要です (マルチポートのシリアルボードの場合は、複数の 16550A ベースのポートで一つまたは二つの IRQ を共有するための機構を備えています)。

17.2.4. デバイススペシャルファイル

カーネルに組み込まれているほとんどのデバイスは、 /dev ディレクトリにある、"デバイススペシャルファイル"を介してアクセスされます。 sio デバイスの場合は、着信用の /dev/ttydN および、発信用の /dev/cuaaN が利用されます。さらに、FreeBSD は、初期化デバイス (/dev/ttyidN と /dev/cuai0N) およびロッキングデバイス (/dev/ttyldN と /dev/cual0N) も用意しています。
初期化デバイスは、通信ポートがオープンされる度に、
そのポートの初期設定を行うために使われます。たとえば、 RTS/CTS によるフロー制御を行うモデムが接続されている場合の crtscts

などのパラメータの初期化が行われます。 ロッキングデバイスは、ポートの設定をロックし、他のユーザやプログラムにこれらを変更されることのないようにするために利用されます。
通信ポートの設定、デバイスのロックと初期化および設定の変更に関しては、 それぞれ termios(4), sio(4) と stty(1) のマニュアルをご覧ください。

デバイススペシャルファイルの作成

FreeBSD 5.0 には、 必要に応じてデバイスノードを自動的に作成する devfs ファイルシステムがあります。 devfs が有効になっているバージョンの FreeBSD を動かしているなら、 この節は飛ばしてかまいません。

デバイススペシャルファイルの管理は、ディレクトリ /dev にあるシェルスクリプト MAKEDEV で行います。 MAKEDEV を使って、 COM1 (ポート 0) をダイアルアップのポートとして利用するためのデバイススペシャルファイルを作るには、 /dev に cd してから、 MAKEDEV ttyd0 と実行してください。 同様に、MAKEDEV ttyd1 とすることで、 COM2 (ポート 1) 用のデバイススペシャルファイルを作成できます。

MAKEDEV は、 /dev/ttydN のデバイススペシャルファイルだけでなく、 /dev/cuaaN, /dev/cuaiaN, /dev/cualaN, /dev/ttyldN および /dev/ttyidN ノードも作成します。

デバイススペシャルファイルの作成後、 これらのファイルの許可属性が適切に設定されていて、これらのデバイスを利用してもよいユーザのみが読み書きできるようになっていることを確認してください (特に /dev/cua* の許可属性には注意を払ってください)。 この確認を怠ると、一般のユーザがあなたのモデムを使うことができるようなことになりかねません。 デフォルトの /dev/cua* の許可属性は、以下のようになっていて、 たいていの場合適切なものだと思います。

```
crw-rw----    1 uucp     dialer    28, 129 Feb 15 14:38 /dev/cuaa1
crw-rw----    1 uucp     dialer    28, 161 Feb 15 14:38 /dev/cuaia1
crw-rw----    1 uucp     dialer    28, 193 Feb 15 14:38 /dev/cuala1
```

上の設定では、ユーザ uucp と、グループ dialer に属するユーザが発信用のデバイスを利用できます。

17.2.5. シリアルポートの設定

ttydN (または cuaaN) デバイスは、 アプリケーション上でシリアルポートをオープンする時に使用する、標準的なデバイスです。プロセスがデバイスをオープンする際、端末 I/O 設定のデフォルトセットが適用されます。これらの設定内容は、
次のコマンドで確認することができます。

```
# stty -a -f /dev/ttyd1
```

このデバイスの設定を変更した場合、 その設定はデバイスがクローズされるまで有効です。デバイスが再びオープンされる時、デフォルトの設定値に戻ります。
デフォルトの設定を変更するためには、"初期状態" を設定したいデバイスをオープンして調節できます。たとえば、ttyd5 というデバイスに対して、デフォルトで CLOCAL モード, 8 bits, XON/XOFF フロー制御を設定したい場合は、 次のように入力してください。

```
# stty -f /dev/ttyid5 clocal cs8 ixon ixoff
```

システム全体のシリアルデバイス初期化は /etc/rc.serial で制御されています。
このファイルは、シリアルデバイスのデフォルトの設定を決めます。

また、"ロック状態"のデバイスに調節を加えることで、
アプリケーションがある種の設定を変更してしまうことを防げます。 たとえば、ttyd5 の速度を 57600 bps に固定したい場合には、次のように入力してください。

```
# stty -f /dev/ttyld5 57600
```

これで、ttyd5 をオープンして、 シリアルポートの転送スピードを変更しようとするアプリケーションは 57600 bps で頭打ちになります。

本来、初期状態やロックされているデバイスに書き込めるのは `root` アカウントだけにすべきです。

17.3. シリアル端末

シリアル端末を利用することで、 コンピュータのコンソールのそばにいないときや、
手近にネットワーク接続されているコンピュータがないときでも、 FreeBSD
の機能を便利に、かつ安価に利用することができます。 ここでは、FreeBSD
にシリアル端末を接続する方法を解説します。

17.3.1. 端末の種類と利用方法

もともと Unix システムにはコンソールがありませんでした。
ユーザはコンピュータのシリアルポートに接続された端末からログインしてプログラムを利用していました。 ちょうどモデムと通信ソフトを使ってリモートのコンピュータにログインし、
テキストベースのプログラムを利用するのとよく似ています。

最近の PC は、 高品質の画像を表示できるコンソールを搭載していますが、 ほとんどすべての Unix 系 OS には未だにシリアルポートを使ってログインするための機能があり、 FreeBSD
でもこの機能がサポートされています。
現在使用されていないシリアルポートに端末を接続することでシステムにログインし、
通常はコンソールや X ウィンドウシステムの `xterm`
のウィンドウ上で起動しているテキストベースのプログラムであれば何でも利用できます。

職場での利用ということで考えるならば、FreeBSD
が動作しているコンピュータに接続された何台ものシリアル端末を各社員の机に配置するというようなことが可能です。 また、家庭での利用方法としては、余っている古い IBM PC や Macintosh を FreeBSD が動いているパワフルなコンピュータの端末として利用できます。
普通ならシングルユーザのコンピュータを、
パワフルなマルチユーザのシステムに変えることができるのです。

FreeBSD では、以下に挙げる 3 種類の端末が利用できます。

- ダム (dumb) 端末
- PCを利用した端末

- X 端末

以下は、それぞれについての解説です。

ダム端末

ダム端末は、シリアルライン経由でのコンピュータとの接続専用のハードウェアです。ダム端末は、テキストの送受信および表示ができる程度の計算能力しかもっていないので、"dumb" (間抜け) というように呼ばれています。この端末上でプログラムを実行することはできません。テキストエディタ、コンパイラ、E-mail、ゲームなどなどのプログラムを実行するのは、ダム端末を接続しているコンピュータの方です。

Digital Equipment 社の VT-100 や、Wyse 社の WY-75 を初めとして、多くのメーカが何百種類ものダム端末を作っています。ほとんどどんな種類のダム端末でも FreeBSD に接続して使用できます。さらに、高性能の端末の中には画像を取り扱えるものもありますが、限られた数のソフトウェアパッケージしかこういった機能には対応していません。

ダム端末は、X ウィンドウシステムで提供されるようなグラフィックアプリケーションを必要としない職場で広く用いられています。

PC を端末として利用する

ダム端末 がテキストの表示および送受信の機能をそなえただけのものならば、言うまでもなく、どんな PC もダム端末になり得ます。必要なものは適切なケーブルと、その PC の上で動作する端末エミュレーション を行うソフトウェアのみです。

このような環境は、家庭においてよく利用されます。たとえば、あなたの同居人が FreeBSD のコンソールを専有している時などに、あまりパワーのないコンピュータを FreeBSD システムにシリアル端末として接続し、その端末上でテキストだけを用いる作業をおこなうことができます。

X 端末

X 端末は、既存のものの中で最も洗練された種類の端末といえます。X 端末は、たいていの場合シリアルポートではなく、イーサネットのようなネットワークを利用した接続をおこないます。また、アプリケーションの利用においても、テキストベースのものだけでなく、X アプリケーションの利用が可能です。

ここでは、参考までに端末について紹介しただけで、X 端末の設定や利用についての解説はおこないません。

17.3.2. 設定

ここでは、端末からのログインを可能にするために必要な FreeBSD 側の設定について解説します。既に端末を接続するポートが利用できるように kernel の設定をおこない、端末が接続されているものと考えて、解説を進め ます。

FreeBSD の起動のプロセス で述べたように init プロセスは、システム起動時にすべてのプロセス管理や初期化をおこなっています。init が行っている仕事の一つは、/etc/ttys ファイルを読んで、利用可能な端末上で getty

プロセスを起動することです。 getty プロセスは、 ログイン名を読み込み login プログラムを起動します。

したがって、FreeBSD の端末を設定するには、 root で次の手順を踏まなければなりません。

1. 端末を接続するポートの /dev のエントリが含まれている行がまだ存在しなければ、これを /etc/ttys に追加してください。
2. /usr/libexec/getty が対象となるポートに対して実行されるように指定してください。また、/etc/gettytab ファイル内の適切な *getty* タイプのエントリを指定してください。
3. デフォルトのターミナルタイプを指定してください。
4. 対象となるポートを "on" に設定してください。
5. そのポートが "secure" であるかどうかを指定してください。
6. init に /etc/ttys を読み込みなおさせてください。

また、必要に応じて /etc/gettytab を変更し、上の 2で使用する *getty* のエントリを追加してください。この章ではこの方法については特に解説しませんので、gettytab(5) および getty(8) のマニュアルをご覧ください。

/etc/ttys へのエントリの追加

/etc/ttys には、FreeBSDシステム上のログインを許可するすべてのポートを記述します。たとえば、一つ目の仮想コンソール ttyv0 のエントリもこのファイルにあります。このエントリのおかげで、コンソールからのログインが可能になっています。このファイルには、他の仮想コンソール、シリアルポートおよび仮想端末のエントリも含まれています。端末を接続する場合は、そのポートの /dev のエントリを、/dev の部分を省略して記述します (たとえば /dev/ttyv0 については、ttyv0 として記述します)。

FreeBSD のデフォルトのインストール状態では、ttyd0 から ttyd3 までの、初めの 4 つのシリアルポートに対応した /etc/ttys ファイルが置かれています。これらのポートのいずれかに端末を接続する場合は、新たにエントリを追加する必要はありません。

例 *40.* 端末の項目を */etc/ttys* に追加する

システムに 2 台の端末、Wyse-50 と、VT-100 端末をエミュレートしている Procomm 端末ソフトウェアを動かしている古い 286 IBM PC をシステムに接続しようとしていると考えてください。Wyse は 2 番目のシリアルポートに、286 は 6 番目のシリアルポート (マルチポートシリアルカード上のポート) に接続します。/etc/ttys 内の対応する項目は次のようになります。

```
ttyd1  "/usr/libexec/getty std.38400"  wy50  on  insecure
ttyd5   "/usr/libexec/getty std.19200"  vt100  on  insecure
```

- 最初のフィールドには、通常 /dev にある端末のスペシャルファイル名を指定します。
- 2 番目のフィールドは、この回線に対して実行するコマンドで、通常は getty(8) です。getty

は、回線を初期化して開き、速度を設定して、 ユーザ名を入力するプロンプトを出して login(1) プログラムを実行します。getty プログラムは、 コマンドラインから (省略可能な) パラメータ *getty* タイプを受け取ります。 *getty* タイプは、 bps レートやパリティのような端末回線の特性を示します。 getty プログラムは、 これらの特性を /etc/gettytab ファイルから読み込みます。/etc/gettytab ファイルには、 端末回線について新旧多くの項目があります。 ほとんどの場合、std で始まる項目は、ケーブルで接続された端末に働きます。 これらの項目はパリティを無視します。110 から 115200 までの間の bps レートそれぞれに対して一つ std 項目があります。もちろん、 このファイルに独自の項目を加えてもかまいません。 gettytab(5) のマニュアルに詳しい情報が載っています。/etc/ttys ファイルに *getty* タイプを設定する時は、 端末の通信設定が対応していることを確かめましょう。たとえば、Wyse-50 はパリティなしで、38400 bps で接続します。286 PC はパリティなしで、19200 bps で接続します。

- 第 3 フィールドは、その tty 回線に通常つながる端末の種別です。 ダイアルアップポートでは、実際、 ユーザがどんな種類の端末やソフトウェアで接続してくることもありうるので、 このフィールドには unknown または dialup がよく使われています。 ケーブルで配線された端末については、 端末種別は変わりませんので、termcap(5) データベースファイルから実際の端末種別を、 このフィールドに記入できます。我々の例では、Wyse-50 には実際の端末種別を使っていますが、 Procomm を動かしている 286 PC は、VT-100 をエミュレートするように設定します。
- 4 番目のフィールドは、 ポートを有効にすべきかどうかを指定します。 ここに on と記入すると、 init プロセスが 2 番目のフィールドに記載されているプログラム、 getty を起動します。 このフィールドを off にすると、 getty は動かず、 そのポートからはログインできません。
- 最後のフィールドは、 そのポートが安全かどうか指定します。 あるポートが安全だということは、そのポートから root (またはその他の UID が 0 の) アカウントのログインを許可してよいと信頼しているということです。 安全でないポートからは、root のログインは許可されません。安全でないポートでは、 ユーザは特権を持たないアカウントでログインした後に、 su(1) や類似の仕組みを使ってスーパユーザ特権を獲得します。鍵のかかる部屋にある端末であっても、"insecure" にしておくことが強く推奨されます。 スーパユーザ特権が必要なら、ログインしてから su を使うのは十分簡単です。

init にファイル /etc/ttys の再読み込みをさせる

必要な変更を /etc/ttys ファイルに加えたら、SIGHUP (ハングアップ) シグナルを init プロセスに送って設定ファイルを強制的に再読み込みさせます。 たとえば

```
# kill -HUP 1
```

init は、システムで最初に起動するプロセスなので、 PID は常に 1 です。

すべての設定が正しくおこなわれ、 すべてのケーブルがただしく接続されていて、かつ端末の電源が入っていれば、この時点で各端末で getty プロセスが動いていて、ログインプロンプトが表示されているはずです。

17.3.3. 接続のトラブルシューティング

細心の注意を払って設定をおこなっても、 ときには端末の接続がうまくいかない場合があるでしょう。以下に、 よく見られる問題とその解決方法 を示します。

ログインプロンプトが表示されない

端末の電源が接続され、 スイッチが入っていることを確認してください。もし、PC を端末として利用している場合は、 通信ソフトが適切なシリアルポートを利用する設定になっているかどうか確かめてください。

ケーブルがしっかりと端末と FreeBSDが動作しているコンピュータの両方に接続されていることを確認してください。また、 正しい種類のケーブルを利用している か確かめてください。

端末と FreeBSD の間の通信速度とパリティの設定が一致していることを確認 してください。出力をモニタに表示するタイプの端末の場合は、モニタのコントラストと明るさの設定を確認してください。また、 出力が印刷 されるタイプの端末の場合は、紙とインクが十分にあるかどうかを確かめてく ださい。

getty が動いていて、 端末を認識していることを確認してください。 たとえば、動作中の getty プロセスの一覧を ps で取得するには、以下のように入力してください。

```
# ps -axww|grep getty
```

その端末に対応する項目が表示されるはずです。 たとえば、以下の表示例は、getty は 2 番目のシリアルポート (ttyd1) に対して /etc/gettytab 中の std.38400 エントリを使って動作しているということを示しています。

```
22189  d1  Is+    0:00.03 /usr/libexec/getty std.38400 ttyd1
```

もし、getty プロセスが一つも動いていないようであれば、 /etc/ttys の中で、そのポートを利用可能にする設定をしたかどうか確かめてください。 また、ttys ファイルを変更したら、kill -HUP 1 を実行するのを忘れないでください。

ログインプロンプトの代わりにゴミが表示される

端末と FreeBSD の間の通信速度およびパリティの設定が一致していることを確かめてください。また、getty プロセスの情報を調べて、適切な *getty* のタイプが使用されていることを確認してください。間違った *getty* タイプが使用されている場合は、/etc/ttys を修正し てから、 kill -HUP 1 を実行してください。

文字が重複して表示される、入力したパスワードが表示される

端末または通信ソフトの設定で、"半二重 (half duplex)" あるいは "ローカ ルエコー" となっているところを、"全二重 (full duplex)" に変更してください。

17.4. ダイアルインサービス

訳*: . 6 September 1996.*

FreeBSD システムをダイアルインサービス用に設定することは、端末の代わりにモデムを扱うこと以外は、 端末の接続によく似ています。

17.4.1. 外づけモデムと内蔵モデムについて

ダイアルアップのサービスに関していえば、 外づけのモデムの方が適している ようです。これは、多くの外づけのモデムは設定を不揮発ラムに書き込んで半 永久的に保存することができますし、また RS-232 に関する重要な情報を知る ための点滅するライトによるインディケータが搭載されているからです。点滅 するライトは、システムを見に来た訪問者に強い印象を与えるという効果だけ
でなく、モデムが適切に動作しているかどうかを知るためにも 有効です。

一方、たいていの内蔵型のモデムには 不揮発性ラムが搭載されていないため、ディップスイッチの変更以外に設定を保存する方法がありません。また、もしインディケータがついていても、おそらくコンピュータのケース カバーが外されていなければその状態を確認するのは 難しいでしょう。

モデムとケーブル

外付けモデムを使用しているなら、 それにあったケーブルが必要です。通常の信号が全て接続されている限り、標準的な RS-232C ケーブルで十分でしょう。

- Transmitted Data (SD)
- Received Data (RD)
- Request to Send (RTS)
- Clear to Send (CTS)
- Data Set Ready (DSR)
- Data Terminal Ready (DTR)
- Carrier Detect (CD)
- Signal Ground (SG)

FreeBSD で 2400bps 以上の転送速度を利用する場合には、 フロー制御のため に RTS 信号と CTS 信号が必要です。また、 接続の確立と回線の切 断を検出するために CD 信号を利用します。さらに、DTR 信号を使っ て回線切断後のモデムのリセットを行います。ケーブルの中には、 総ての必要な信号線が接続されていないものもありますので、 たとえば、回線切断後でも ログインセッションが残ってしまうといった問題が発生した場合などには、
ケーブルに問題がある可能性もあります。

FreeBSD も他の Unix 系 OS と同様、回線の接続およ び切断の検出や回線の切断および回線切断後のモデムの初期化にハードウェア シグナルを利用します。FreeBSD は、モデムに対するコマンドの送信やモデ ムの状態の監視を行いません。パソコンで運用されている BBS への接続に慣 れている方にとっては、 ちょっとめんどうかもしれませんね。

17.4.2. シリアル インタフェースについて

FreeBSD では、NS8250-、NS16450-、NS16550- および NS16550A- に基づ いた EIA RS-232C (CCITT V.24) 規格のシリアル インタフェースをサポート しています。8250 および 16450 ベースのディバイスには1文字のキャラクタ バッファが搭載されています。また、16550 系のディバイスには、 16文字分 のバッファが搭載されていて、はるかによいパフォーマンスを得られます (ただし、無印の 16550 では、バグがあって 16 文字バッファが利用できませ んので、可能であれば 16550A 系のディバイスを利用してください)。1文字のバッファの物は、 16550 系のものと比べて OS にかける負荷が大きいので、16550A 系ディバイスの利用を強く推奨します。多数のシリアル ポートを利用する場合や、負荷の高いシステムにおいては、 16550A 系ディバイスを使う ことで、エラー発生率を低く押さえることができます。

17.4.3. 概要

端末に関しては、 ダイアルイン接続に割り当てられたそれぞれのシリアルポートに対して、 `init` が `getty` を起動します。たとえば、モデムが /dev/ttyd0 に割り当てられていたら、`ps ax` コマンドを実行すると、以下のような出力が得られるはずです。

```
4850 ??  I      0:00.09 /usr/libexec/getty V19200 ttyd0
```

ユーザがモデムに電話をかけ、モデム同士が接続されると、 モデムの CD (Carrier Detect) が検出されます。その結果、 kernel がキャリア信号を検出して、`getty` によるポートのオープンの処理が終了します。 `getty` は、`login:` プロンプトを指定されている初期回線速度で送信します。 `getty` は、正常に文字列を受信できるかどうか監視し、通常の設定では、 もし異常な文字列を検出した場合 (理由としては、 `getty` の速度とモデ ムの接続速度が異なっているような場合が考えられます)、正常に文字列が 受信できるまで、`getty` は速度を変え続けます。

ユーザがログイン名を入力すると、 `getty` は /usr/bin/login を起動して、パスワードの入力を要求し、その 後ユーザのシェルを起動します。

17.4.4. 設定ファイル

FreeBSD
のシステムへのダイアルアップによるアクセスを実現するために編集が必要と思われる設定ファイルが、/etc ディレクトリに 3 つあります。まず、 /etc/gettytab には、 /usr/libexec/getty デーモンの設定を記述します。つぎに、 /etc/ttys に保存されている情報から、 /sbin/init はどの tty デバイスに対して `getty` のプロセスを実行するべきか判断します。 最後に、/etc/rc.serial スクリプトに、シリアルポートの初期化のためのコマンドを記述することができます。

Unix にダイアルアップモデムを接続する方法には、 二つの考え方があります。一つの方法は、ダイアルインしてくるユーザの接続速度に関係なく、 常にモデムとローカルのコンピュータの RS-232 インタフェースの接続速度を一定に保つように設定する方法です。
この設定の長所は、ユーザがダイアルインして接続されると、
即座にシステムからのログインプロンプトが送信されるということです。
短所は、システムが実際のモデム間の速度を知ることができないために、 Emacs のようなフルスクリーンのプログラムが、 端末との接続速度が遅い場合でも、

そのような場合に効果的な方法で画面出力を行わない点です。

もう一つは、モデムの RS-232 インタフェースとコンピュータの接続速度を、モデム間の接続速度に応じて変化させるような設定です。たとえば、 モデム間 の接続が V.32bis (14.4 Kbps) ならば、 モデムとコンピュータの間の接続を 19.2 Kbps とし、 モデム間の接続が 2400 bps の時には、モデムとコンピュータ間も 2400 bps で接続するような設定をします。この場合、 getty は、モデムが返すリザルトコードからモデムとコンピュータの接続速度を認識することができませんので、 getty は、まず初期速度で login: という文字列を送信して、それに対する応答の文字列を監視します。
ここで、ユーザ側の端末に無意味な文字列が表示された場合、 ユーザは意味のある文字列を受信するまで Enter キーを繰り返し押さなければならないということを知っていると仮定しています。もし接続速度が間違っている場合、getty は、 ユーザから送られた文字を無意味な文字列として扱い、次の速度を試します。そして、ここで再度 login: プロンプトを送信します。この一連の動作が異常な回数繰り返されることも考えられますが、 普通は 1 度か 2 度のキー入力があれば、 ユーザはまともなプロンプトを受信できます。
このログインの動作が前者の固定速度による方法に比べて美しくないのは明らかですが、
この方法では、低速度で接続しているユーザに対するフルスクリーンのプログラムからのレスポンスが改善されます。

このセクションでは、両方の設定方法について解説しますが、 どちらかというとモデム間の速度に応じて RS-232 インタフェースの速度が変化するような 設定の方に偏った説明になってしまうと思います。

/etc/gettytab

/etc/gettytab は、getty(8) の設定ファイルで、termcap(5) と同様の形式で記述されます。ファイルのフォーマットや定
義できる機能についての詳細については、gettytab(5) のマニュアルを ご覧ください。

固定速度の設定

モデムとコンピュータ間の通信速度を固定して使う場合、 おそらく /etc/gettytab に特に変更を加える必要はないはずです。

可変速度の設定

getty が利用するモデムとコンピュータの接続速度に関する情報を /etc/gettytab に記述する必要があります。もし、2400 bps のモ デムをお使いになるのであれば、既存の D2400 のエントリがそのまま利 用できるでしょう。

```
#
# Fast dialup terminals, 2400/1200/300 rotary (can start either way)
#
D2400|d2400|Fast-Dial-2400:\
        :nx=D1200:tc=2400-baud:
3|D1200|Fast-Dial-1200:\
        :nx=D300:tc=1200-baud:
5|D300|Fast-Dial-300:\
        :nx=D2400:tc=300-baud:
```

高速モデムをお使いの場合は、おそらく /etc/gettytab に新たなエントリを追加する必要があります。以下の例は、14.4 Kbps のモデムを、 最大インタフェース速度を 19.2 Kbps

として利用するためのエントリです。

```
#
# 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:
```

上記の例を利用した場合、 パリティなし、8ビットの接続が行われます。

上記の例では、まず 19.2 Kbps (V.32bis) によるモデムとコンピュータ間の接続を試み、続いて 9600 bps (V.32)、2400 bps、1200 bps、300 bpsと順に試み、再び 19.2 Kbps による接続を試みるという循環に入ります。 この接続速度の循環は、`nx=`("next table") の機能で実現されています。また、 各行はそれぞれ `tc=`("table continuation") の機能を使って、その他の接続速度に依存した "標準的な" 設定を取り込んでいます。

もし、お使いのモデムが 28.8 Kbps であったり、14.4 Kbps の圧縮転送の機能を有効に利用したい場合は、19.2 Kbps よりも速い速度を利用するように設定する必要があります。 以下に 57.6 Kbps から接続を試みる gettytab の設定例を示しておきます。

```
#
# 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:
```

もし、お使いの CPU が低速のものであったり、CPU に対する負荷が高い場合で、16550A 系のシリアルポートをお使いでない場合、 57.6 Kbps の接続において、`sio` の "silo" エラーが発生するかもしれません。

/etc/ttys

/etc/ttys ファイルの設定は、端末の項目を /etc/ttys に追加する で扱われています。
モデムの設定も似たようなものですが、getty
に異なる引数を渡して、異なる端末種別を指定しなければなりません。
固定速度および可変速度両方に共通する形式は次のようになります。

```
ttyd0   "/usr/libexec/getty xxx"   dialup on
```

1 番目の項目は、このエントリで対象とするデバイススペシャルファイルです。 上の例では ttyd0 として、 /dev/ttyd0 を getty に監視させることを表しています。2 番目の項目 "/usr/libexec/getty xxx" (*xxx* は初期段階で使われる gettytab のエントリに置き換えてください) が、init がこのディバイスに対して起動するプロセスです。3 番目の dialup は、デフォルトのターミナルタイプです。 4 番目の on は、 この行が有効であることを init に対して示しています。5 番目の項目に secure を指定することもできますが、これは、
たとえばシステムのコンソールのように、
物理的に安全な端末に対してのみ指定するようにしてください。

デフォルトのターミナルタイプ (上記の例では dialup)
は、ローカルのユーザの好みによって異なってきます。
ユーザがログインスクリプトをカスタマイズして、ターミナルタイプが dialup
の時には自動的に他のターミナルタイプを設定できるように、
ダイアルアップのポートのデフォルトのターミナルタイプには dialup が伝統的に用いられています。
しかし、筆者のサイトでは、ほとんどのユーザが VT102 エミュレーションを使っているので、
ダイアルアップのポートのデフォルトターミナルタイプとして vt102 を指定しています。

/etc/ttys の修正がすんだら、 以下のようなコマンドを使って init プロセスに HUP
シグナルを送り、/etc/ttys を読み込み直させてください。

```
# kill -HUP 1
```

ただ、もし初めてシステムを設定しているのであれば、 モデムが適切に設定されて接続されるまでは、
init に対してシグナルを送らない方がいいかもしれません。

固定速度の設定

速度を固定する設定では、/etc/ttys の中で、getty に対して固定速度のエントリを指定する必要があります。たとえば、 以下の例はポートのスピードが 19.2 Kbps に固定されたモデムのための ttys のエントリです。

```
ttyd0   "/usr/libexec/getty std.19200"   dialup on
```

モデムが異なる速度で固定されている場合は、 std.19200 のかわりに std.speed
を適切な値に置き換えたものにしてください。 /etc/gettytab
に挙がっている適切な種類を使うようにしてください。

可変速度の設定では、ttys のエントリが、/etc/gettytab の中の適切な "自動速度調整" の初期設定のエントリを参照していなければなりません。 たとえば、もし前述の 19.2 Kbps から接続を試みる可変速度の設定例 (V19200 の gettytab エントリ)をそのまま *ttys* に追加したのであれば、 ttys エントリは以下のようになります。

```
ttyd0   "/usr/libexec/getty V19200"   dialup on
```

/etc/rc.serial

V.32、V.32bis または V.34 モデムのような高速モデムを利用する場合、ハードウェア (RTS/CTS) フロー制御を行う必要があります。FreeBSD kernel のモデムポートにハードウェアフロー制御のフラグを設定するための stty コマンドを、 /etc/rc.serial に記述できます。

たとえば、シリアルポート 1 番 (COM2) のダイヤルインおよびダイヤルアウト初期化デバイスに termios フラグ crtscts を設定するには、次の行を /etc/rc.serial に追加するとよいでしょう。

```
# Serial port initial configuration
stty -f /dev/ttyid1 crtscts
stty -f /dev/cuai01 crtscts
```

17.4.5. モデムの設定

もし、あなたのモデムがパラメータを不揮発ラムに 保存できるタイプならば、MS-DOS 上の Telix や FreeBSD 上の tip などのような通信プログラム を使って、 パラメータを設定してください。getty が利用する初期速度でモデムに接続して、以下の条件を満たすよ
うに不揮発ラムの設定を変更してください。

- 接続時に CD 信号がオンになる
- 接続時に DTR がオンになり、 オフで回線を切断しモ デムをリセットする。
- 送信時フロー制御には CTS を利用。
- XON/XOFF によるフロー制御を行わない。
- 受信時のフロー制御は RTS を使用。
- Quiet mode (リザルト コードを返さない)
- コマンド エコーを返さない。

これらを実現するためのコマンドやディップ スイッチの設定に関しては、モ
デムのマニュアルを参照してください。

以下に、USRobotics Sportster の 14,400 bps の外づけモデムの設定例を示 しておきます。

```
ATZ
ATC1D2H1I0R2W
```

ことのついでに、たとえば、V42.bis や MNP5 のデータ圧縮を使用するかどうかなどのモデムの他の設定について確認、 調整しておくのもよいかもしれま せん。

さらに、USRobotics Sportster の 14,400 bps の外づけモデムでは、以下の ようなディップスイッチの設定も必要です。他のモデムをお使いの方も、以 下の例を設定の参考にしてください。

- スイッチ 1: UP - DTR 標準
- スイッチ 2: N/A (リザルトコードを単語形式にするか数値形式にするか)
- スイッチ 3: UP - リザルトコードを返さない
- スイッチ 4: DOWN - コマンドエコーを返さない
- スイッチ 5: UP - 自動着信
- スイッチ 6: UP - CD 標準
- スイッチ 7: UP - 不揮発ラムからデフォルト値をロードする
- スイッチ 8: N/A (Smart Mode/Dumb Mode)

リザルト コードを返さないように設定しておかないと、 `getty` が誤っ て `login:` プロンプトをコマンドモードのモデムに送信してしまった場 合に、モデムがこの入力をエコーしたり、この入力に対するリザルト コードを返してしまったりすることになります。この結果として、 モデムと `getty` の間で延々と無意味なやりとりが続いてしまう可能性があります。

固定速度の設定

固定速度の設定では、 モデムとコンピュータ間の通信速度をモデムとモデム間の接続速度に関係なく、常に一定に保つように、 モデムを設定する必要があり ます。USRobotics Sportster の 14,400 bps 外づけモデムの場合、以下のコ マンドで、モデムとコンピュータ間の速度が、コマンド送信時の速度に固定さ れます。

```
ATZ
ATB1W
```

可変速度の設定

可変速度の設定では、シリアル ポートの速度が、 着信速度に応じて変化するように設定しなければいけません。 USRobotics Sporster の 14,400 bps 外 づけモデムの場合、以下のコマンドで、エラー訂正機能を利用した通信の場合 は、コマンドを送信した時の通信速度にシリアル ポートの速度を固定し、エラー訂正機能を利用しない接続では、 シリアル ポートの速度が変化するよう に設定されます。

```
ATZ
ATB2W
```

モデムの設定の確認

ほとんどの高速モデムには、 現在の設定をある程度人間にも理解できる形式にして表示させるコマンドがあります。USRobotics Sporster の 14,400 bps 外づけモデムの場合は、`ATI5`

コマンドで、現在の不揮発ラムの設定を 表示することができます。 さらに、ディップスイッチの設定も含めた現在の 設定を確認するためには、ATZ コマンドを送信してから、ATI4 コマンドを送信してください。

他のメーカーのモデムをお使いの場合は、 モデムのマニュアルで設定値の確認方法を確認してください。

17.4.6. トラブルシューティング

以下の手順でダイアル アップ モデムの動作を確認することができます。

FreeBSD システムの動作確認

モデムを FreeBSD システムに接続し、 システムをブートします。あなたのモデムにモデムの状態を確認するためのインジケータがあれば、 DTR のインジケータの状態に注目してください。もし、 システムのコンソールに login: プロンプトが表示された時に、DTR のインジケータが点灯 すれば、FreeBSD が適切なポートに対して getty を起動し、モデムへ の着信を待っている状態であることを意味しています。

もし DTR のインジケータが点灯しない場合は、システムのコンソールから FreeBSD にログインして、ps ax を実行し、 FreeBSD が適切なポートに対してgetty プロセスを起動しようとしているのかどうか確認してください。 プロセスに関する情報の中に、以下のような行が表示されるはずです。

```
114 ??  I      0:00.10 /usr/libexec/getty V19200 ttyd0
115 ??  I      0:00.10 /usr/libexec/getty V19200 ttyd1
```

モデムにまだ着信がない状態の時に、 以下のように上とは異なる出力があった 場合、getty は既にモデムポートのオープンを終了したということに なります。

```
114 d0  I      0:00.10 /usr/libexec/getty V19200 ttyd0
```

getty は、CD (carrier detect) 信号がオンの状態になるまで、ポートのオープンを完了することはできませんので、 この場合は接続に問題があるか、あるいはモデムの設定に問題があることが考えられます。

もし、期待した ttydN ポートをオープンしようとしている getty が見あたらない場合は、再度 /etc/ttys の内容を確認し、 書式などに誤りがないか 調べてみてください。また、ログ ファイル /var/log/messages に init および getty から何か出力がないかどうかも確認してみてく ださい。もし何かメッセージが記録されていたら、再度 /etc/ttys、 /etc/gettytab の二つの設定ファイルと、ディバイス スペシャルファイル /dev/ttydN を確認し、記述に誤りがないか、足りないエントリがないか、 足りないディバイススペシャルファイルがないかといった 点について調べてみてください。

モデムで接続してみる

実際にモデムを使って別のコンピュータから 接続してみてください。この時、8 ビット、パリティなし、1 ストップビットで接続するようにしてください。 接続後すぐにプロンプトが返ってこない場合や、無意味な文字列が表示される 場合は、1秒に1回くらいの割合で Enter キーを押してみてください。

しばらくたって、なおも login: プロンプトが現れない場合は、BREAK 信号を送信してみてください。この時、端末側で使っているモデムが高速モデムならば、このモデムのインタフェースの接続速度を固定してから、再度ダイアルインしてみてください。(たとえば、USRobotics Sportster の場合は、ATB1)

それでもまだ login: プロンプトが表示されない場合は、/etc/gettytab の以下の点について再度確認してみてください。

- /etc/ttys の対応する行の2番目の項目で、/etc/gettytab の中で定義されているエントリが指定されているか
- 各 nx= で /etc/gettytab の中で定義されているもの が指定されているか
- 各 tc= で /etc/gettytab の中で定義されているもの が指定されているか

もしダイアルインしても、FreeBSD システム側のモデムが応答しない場合は、FreeBSD 側のモデムが DTR がオンになった時に電話にでるように設定されているかを確認してください。もしモデムの設定に問題がなさそうならば、モデムのインジケータ (がもしあれば) で、DTR がオンになっているか を確認してください。

この確認のステップを数回繰り返してもうまくいかない場合は、一度休憩して、しばらくたってから挑戦してみましょう。それでもだめなら、おそらく FreeBSD general questions メーリングリストにあなたのモデムについての情報と問題を書いたメールを送れば、メーリングリストのメンバーが問題の解決を助けるべく努力してくれる でしょう。

17.5. ダイアルアウトサービス

以下はモデムを利用して他のコンピュータと接続する方法を説明しています。これはリモートホストとターミナル接続を確立するための 適切な方法です。

これは BBS に接続するときによく使います。

この種の接続は PPP 接続に問題がある場合、Internet 上にあるファイルを転送するのに非常に役に立ちます。FTP で何らかのファイルを転送したいのに PPP 接続を確立できない場合は、ファイルを FTP 転送するためにターミナルセッションを利用します。そして ZMODEM を利用してファイルを転送します。

17.5.1. 私の Hayes モデムはサポートされていません、どうすればよいでしょう?

実際、tip の マニュアルページは古くなっています。既に Hayes ダイアラが組み込まれています。/etc/remote ファイル中で at=hayes を使ってください。

Hayes ドライバは、最近のモデムの新しい機能である BUSY、NO DIALTONE、CONNECT 115200などのメッセージを認識できるほど賢くはなく、単に混乱を起こすだけです。tipを使う場合には、(ATX0W とするなどして) これらのメッセージを表示させないようにしなくてはいけません。

また、tip のダイアルのタイムアウトは60秒です。モデムのタイムアウト設定はそれより短くすべきであり、そうしないと tip は通信に問題があると判断するでしょう。ATS7=45W を実行してください。

デフォルトの tip は、Hayes モデムに完全に対応しているわけではありません。解決方法は /usr/src/usr.bin/tip/tip の下の tipconf.h を変更することです。もちろんこれにはソース配布ファイルが必要です。

#define HAYES 0 と記述されている行を #define HAYES 1 と変更し、そして make, make install を実行します。これでうまく動作するでしょう。

17.5.2. これらの AT コマンドを入力するには?

/etc/remote ファイルの中で "direct" エントリを作ります。たとえばモデムが1番目のシリアルポートである /dev/cuaa0 に接続されている場合、次のようにします:

```
cuaa0:dv=/dev/cuaa0:br#19200:pa=none
```

モデムがサポートする最大の bps レートを br フィールドに使います。そして tip cuaa0 を実行すると、モデムが利用できるようになります。

/dev/cuaa0 がシステムに存在しない場合は、次のようにします:

```
# cd /dev
# sh MAKEDEV cuaa0
```

または root になって以下のように cu コマンドを実行します:

```
# cu -lline -sspeed
```

line にはシリアルポートを指定します (例えば /dev/cuaa0)。そして *speed* には接続する速度を指定します (例えば 57600)。その後 AT コマンドを実行したら、`~.` と入力すれば終了します。

17.5.3. pn 機能の @ 記号が使えません!

電話番号 (pn) 機能の中での @ 記号は、tip に /etc/phone にある電話番号を参照するように伝えます。しかし @ の文字は /etc/remote のような設定ファイルの中では特殊文字となります。 バックスラッシュを使ってエスケープをおこないます:

```
pn=\@
```

17.5.4. コマンドラインから電話番号を指定するには?

"generic" エントリと呼ばれるものを /etc/remote に追加します。 例えば次のようにします:

```
tip115200|Dial any phone number at 115200 bps:\
        :dv=/dev/cuaa0:br#115200:at=hayes:pa=none:du:
tip57600|Dial any phone number at 57600bps:\
```

```
        :dv=/dev/cuaa0:br#57600:at=hayes:pa=none:du:
```

そして

```
# tip -115200 5551234
```

のように利用できます。 tip より cu を使いたい場合、 cu の generic エントリを使います。

```
cu115200|Use cu to dial any number at 115200bps:\
    :dv=/dev/cuaa1:br#57600:at=hayes:pa=none:du:
```

そして

```
# cu 5551234 -s 115200
```

と実行します。

17.5.5. 毎回 bps レートを入力しなければいけませんか?

tip1200 や cu1200 用のエントリを記述し、適切な通信速度を br フィールドに設定します。tip は 1200 bps が正しいデフォルト値であるとみなすので、 tip1200 エントリを参照します。もちろん 1200 bps を使わなければならないわけではありません。

17.5.6. ターミナルサーバを経由して複数のホストへアクセスしたいです

毎回接続されるのを待って CONNECT host と入力する かわりに、tip の cm 機能を使います。例えば、/etc/remote に次のようなエントリを追加します:

```
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/cuaa2:br#38400:at=hayes:du:pa=none:pn=5551234:
```

これで、tip pain や tip muffin と実行すると pain や muffin のホストに接続することができ、 tip deep13 を実行するとターミナルサーバに接続します。

17.5.7. tip を使ってそれぞれのサイトの 複数の回線に接続できますか?

これは大学に電話回線がいくつかあって 数千人の学生が接続しようとする 場合によくある問題です。

あなたの大学のエントリを /etc/remote ファイルに作成して、pn のフィールドには @ を使います:

```
big-university:\
    :pn=\@:tc=dialout
dialout:\
    :dv=/dev/cuaa3:br#9600:at=courier:du:pa=none:
```

そして /etc/phone ファイルに大学の電話番号の一覧を書きます:

```
big-university 5551111
big-university 5551112
big-university 5551113
big-university 5551114
```

tip は一連の電話番号を試みて、最終的に接続できなければあきらめます。リトライを続けさせたい場合は、tip を while ループに入れて 実行します。

17.5.8. Ctrl + P を 1 回送るために Ctrl + P を 2 度押す必要があるのはなぜ?

Ctrl + P はデフォルトの "force (強制)" 文字であり、tip に次の文字がリテラルデータであることを伝えます。force 文字は "変数の設定" を意味する ~s エスケープによって他の文字にすることができます。

~sforce=single-char と入力して改行します。*single-char* は、任意の 1 バイト文字です。 *single-char* を省略すると NUL 文字になり、これは Ctrl + 2 や Ctrl + Space を押しても入力できます。また、*single-char* に Shift + Ctrl + 6 を割り当てる方法を使っているターミナルサーバもあります。

$HOME/.tiprc に次のように定義することで、任意の文字を force 文字として利用できます:

```
force=single-char
```

17.5.9. 打ち込んだ文字が突然すべて大文字になりました??

Ctrl + A を押してしまい、caps-lock キーが壊れている場合のために設計された tip の "raise character" モードに入ったのでしょう。 既に述べたように ~s を使って、raisechar をより適切な値に変更してください。もしこれら両方の機能を使用しないのであれば、force 文字と同じ設定にすることもできます。

以下は Ctrl + 2 や Ctrl + A などを頻繁に使う必要のある Emacs ユーザにうってつけの .tiprc ファイルのサンプルです。

```
force=^^
raisechar=^^
```

^^ は Shift + Ctrl + 6 です。

17.5.10. `tip` でファイルを転送するには?

もし他の Unix のシステムと接続しているなら、 `~p`(put) や `~t`(take) でファイルの送受信ができます。これらのコマンドは 相手のシステムの上で `cat` や `echo` を実行することで 送受信をします。 書式は以下のようになります: `~p` ローカルのファイル名 [リモートのファイル名] `~t` リモートのファイル名 [ローカルのファイル名]

この方法ではエラーチェックをおこないませんので、zmodem
などの他のプロトコルを使った方がよいでしょう。

17.5.11. `tip` から **zmodem** を実行するには?

ファイルを受信するには、 リモート側で送信プログラムを起動します。そして `~C rz` と入力すると、ローカル側へのファイルの受信が 始まります。

ファイルを送信するには、 リモート側で受信プログラムを起動します。そして `~C sz files` と入力すると、 リモート側への ファイルの送信が始まります。

17.6. シリアルコンソールの設定

17.6.1. 導入

FreeBSD は、 コンソールとしてシリアルポート上のダム端末しか持たないシステムでも起動します。この様な構成はきっと次のような二種類の人達に便利でしょう。それは、
キーボードやモニタのないマシンに FreeBSD をインストールしたいシステム管理者と、
カーネルやデバイスドライバをデバッグしたい開発者です。

FreeBSD の起動のプロセス で説明されているように、 FreeBSD は 3 ステージ構成のブートストラップを用いています。 最初の 2 つのステージは、 ブートディスクにある FreeBSD スライスの最初に格納されている、 ブートブロックのコードが行います。
それからブートブロックは、第 3 ステージのコードとしてブートローダ (/boot/loader) を読み込み、実行します。

シリアルコンソールを設定するためには、ブートブロックコード、
ブートローダコード、カーネルを設定する必要があります。

17.6.2. シリアルコンソールの設定

1. シリアルケーブルを用意してください。

   ヌルモデムケーブル、 もしくは標準シリアルケーブルとヌルモデムアダプタが必要となります。
   シリアルケーブルについては ケーブルとポート をご覧ください。

2. キーボードをはずして下さい。

   たいていの PC システムは Power-On Self-Test (POST) の間にキーボードを検出し、もし見つからなければエラーと
   なります。また、キーボードがないことを大きな音で知らせ、
   キーボードが接続されるまでは起動を中断するようなマシンもあります。

コンピュータがエラーを表示していても、 とにかく起動するなら特別な対応は必要ありません (Phoenix BIOS を搭載しているマシンには、 `Keyboard failed` と表示されても、正常に起動するものがあります)。

あなたのコンピュータがキーボードを接続していない状態で 起動しないようなら、 (もし可能ならば) エラーを無視するように BIOS を設定する必要があります。設定方法の詳細については、 マザーボードのマニュアルを調べてください。

BIOS の設定でキーボードを "Not installed" にするということは、キーボードを使えないということを 意味しているわけではありません。これは、BIOS がキーボードがなくても文句を言わないように、電源投入時には キーボードを探すな、と指示するだけです。このフラグを "Not installed" にしていてもキーボードを 接続したままにできますし、ちゃんと動作します。

あなたのシステムが PS/2 マウスを使っているなら、 おそらくマウスもキーボード同様にはずす必要があるでしょう。 というのは、PS/2 マウスは部分的にキーボードとハードウェアを 共有しており、マウスを接続したままにしていると、 キーボードも存在する、と誤って検出してしまう可能性があるからです。 AMI BIOS を持つ Gateway 2000 Pentium 90MHz システム はこれに該当すると言われています。 一般的にこれは問題ではありません。なぜなら、どっちにしても マウスはキーボードなしではたいして役に立たないからです。

3. COM1 (sio0) にダム端末を接続してください。

   ダム端末がなければ、かわりに古い PC/XT でモデム プログラムを走らせて使ったり、シリアルポートに他の Unix マシンを繋いだりできます。もしも COM1 (sio0) がなければ、作成してください。 今のところ、COM1 以外のポートを 選択するためにはブートブロックの再コンパイルが必要です。 すでに COM1 を他の装置に 使っていた場合は、一時的にその装置をはずして いったん FreeBSD がうまく動作してから、 新しいブートブロックとカーネルをインストールしてください。 (上記はとにかくファイル/演算 /端末サーバの COM1 が利用可能であると仮定して います。あなたが本当に何かのために COM1 が必要 (で、なおかつその何かを COM2 (sio1) に付け替えることができない) ならば、多分、そもそも 悩んでる場合ではありません。)

4. カーネルコンフィグファイルの COM1 (sio0) に適切なフラグを 設定していることを確認してください。

   関連するフラグ:

   `0x10`

   このポートのコンソールサポートを有効にします。 このフラグが設定されない場合、他のフラグは無視されます。 現在のところ、一つのポートしかコンソールサポートを有効に できません。(config ファイルに書かれた順番で) 最初にこのフラグを 指定されたポートが選択されます。

なお、このオプションを指定するだけでシリアルポートが
コンソールとして使えるわけではありません。
このフラグと一緒に、以下のフラグも指定するかもしくは -h オプションも使ってください。

0x20

後述される -h オプションを無視して、(他に優先度の高いコンソールがない限り)
このポートをコンソールとして指定します。 このフラグは FreeBSD バージョン 2.*X* の
COMCONSOLE オプションに対応するものです。 フラグ 0x20 は必ず フラグ 0x10
と一緒に指定されなければなりません。

0x40

(0x10 と組み合わせることで) このポートを予約し、通常のアクセスができない
ようにします。 このフラグは、シリアルコンソールとして使いたいポートに
指定すべきではありません。 唯一の使い道は、ユニットがカーネルのリモートデバッグ用
であることを指定することです。 リモートデバッグの詳細については The Developer's
Handbook を参照してください。

FreeBSD 4.0 以降では、 フラグ 0x40 の意味が若干異なり、
シリアルポートにリモートデバッグを指定するためには、
別のフラグを使います。

例:

```
device sio0 at isa? port "IO_COM1" tty flags 0x10 irq 4
```

詳細については sio(4) のマニュアルを参照してください。

もしこれらのフラグがセットされていなければ、(別のコンソールで) UserConfig
を実行するか、 カーネルを再コンパイルする必要があります。

5. ブートドライブの a パーティションの ルートディレクトリに boot.config を作成してください。

このファイルは、ブートブロックコードに対してどのように
システムを起動したいかを教えます。
シリアルコンソールを活かすためには、以下のオプションを幾つか -
複数の場合も一行で、設定する必要があります:

-h

内蔵コンソールとシリアルコンソールの切替えを行います。
これを使用してコンソールデバイスを変更できます。 例えば、内蔵 (ビデオ)
コンソールからブートした場合、 カーネルとブートローダがコンソールデバイスとして
シリアルポートを使用するようにするため、 -h を使って指示できます。
反対に、シリアルポートからブートした場合、
ブートローダとカーネルがコンソールとして代わりに
ビデオディスプレイを使用するようにするため、 -h を使用できます。

-D

シングルとデュアルのコンソール設定を切り替えます。 シングル設定では、上記の -h

オプションの状態によって、コンソールは内蔵コンソール (ビデオディスプレイ)かシリアルポートのいずれかになります。
デュアルコンソール設定では、ビデオディスプレイとシリアルポートの両方が、-h オプションの状態によらず、同時にコンソールになります。
しかし、デュアルコンソール設定は、ブートブロックが実行されている間でしか効果を持ちません。一旦ブートローダに制御が移ると、-h オプションによって指定されたコンソールが 唯一のコンソールになります。

-P

ブートブロックがキーボードを検出するようにします。
キーボードが発見できなかった場合には、 -D と -h オプションが自動的にセットされます。

現バージョンのブートブロックでは容量の制限により、-P オプションは拡張キーボードしか検出できません。キーが 101 個より少ない (そして F11 と F12 がない) キーボードは検出されない可能性があります。
この制限から、いくつかのラップトップコンピュータのキーボードは正しく検出されないでしょう。
もし、あなたのシステムがこのようなキーボードを使っているのであれば、-P オプションを外してください。
残念ながら、この問題の回避策はありません。

-P オプションを使ってコンソールを自動的に選ぶか、-h オプションを使ってシリアルコンソールを有効にしてください。

さらに boot(8) で説明されている他のオプションも使う ことができます。

-P 以外のオプションはブートローダ (/boot/loader) に渡されます。ブートローダは、-h オプションだけの状態を 調べることで内蔵ビデオとシリアルポートのどちらがコンソールになるのか決めます。つまり、/boot.config の中で -D オプションを指定して -h オプションを指定しなかった場合、
ブートブロック実行中でのみシリアルポートをコンソールとして使うことができます。ブートローダは内蔵ビデオディスプレイをコンソールとして使います。

6. マシンを起動する。

FreeBSD を起動したとき、ブートブロックは /boot.config の内容をコンソールに表示します。例えば、

```
/boot.config: -P
Keyboard: no
```

行の二番目は、/boot.config にオプション -P が指定してあるときだけ表示され、キーボードが存在するかどうかを表します。
これらのメッセージは、シリアルか内蔵のいずれか、あるいはその両方のコンソールに表示されます。どちらに表示されるかは、/boot.config の設定によって変わります。

オプション指定	メッセージの表示される場所
なし	内蔵
-h	シリアル
-D	シリアルと内蔵の両方
-Dh	シリアルと内蔵の両方
-P、キーボードが存在する場合	内蔵
-P、キーボードが存在しない場合	シリアル

このメッセージが表示された後、 ブートブロックがブートローダのロードを再開し、
他の全てのメッセージがコンソールに表示されるまで、
若干時間がかかります。通常の環境では、ブートブロックに
割り込みをかける必要はありませんが、
ちゃんとセットアップされているかどうか確かめるために、
割り込みをかけることができるようになっています。

ブートプロセスに割り込みをかけるには、 コンソールの (Enter 以外の) キーをたたいて下さい。
ブートブロックはその時、操作を指定するためのプロンプトを表示します。
こんな風に表示されるでしょう。

```
>> FreeBSD/i386 BOOT
Default: 0:wd(0,a)/boot/loader
boot:
```

上に示したメッセージが、シリアルか内蔵、 あるいはその両方といった、/boot.config
で指定したとおりのコンソールに表示されることを確認して下さい。
メッセージが正しいコンソールに表示されたら、 Enter
キーを押してブートプロセスを継続してください。

もし、シリアルコンソールを利用するように設定しているのに
シリアル端末にプロンプトが出てこない場合は、 設定のどこかに間違いがあります。
ブートブロック(とブートローダ、カーネル)に対して
シリアルポートをコンソールに使うことを伝えるため、 割り込みをかけた時に -h を入力し、
(可能ならば) Enter/Return キーを押して下さい。そして、
一度システムを起動させてから、どこが悪いのかをチェックして下さい。

ブートローダがロードされ、ブートプロセスの第三ステージに
いる時には、まだ内蔵コンソールとシリアルコンソールを切り替えることができます。
それにはブートローダの環境変数を適切に設定すれは良いのですが、 詳細については
ブートローダからコンソールを変更するには を参照してください。

17.6.3. まとめ

このセクションで扱ったさまざまな設定と、 最終的に選択されるコンソールに関するまとめです。

Case 1: sio0 の flags に 0x10 をセットした場合

```
device sio0 at isa? port "IO_COM1" tty flags 0x10 irq 4
```

/boot.config 内のオプション	ブートブロック実行中のコンソール	ブートローダ実行中のコンソール	カーネルのコンソール
なし	内蔵	内蔵	内蔵
-h	シリアル	シリアル	シリアル
-D	内蔵、シリアルの両方	内蔵	内蔵
-Dh	内蔵、シリアルの両方	シリアル	シリアル
-P、キーボードが存在する場合	内蔵	内蔵	内蔵
-P、キーボードが存在しない場合	内蔵、シリアルの両方	シリアル	シリアル

Case 2: sio0 の flags に 0x30 をセットした場合

```
device sio0 at isa? port "IO_COM1" tty flags 0x30 irq 4
```

/boot.config 内のオプション	ブートブロック実行中のコンソール	ブートローダ実行中のコンソール	カーネルのコンソール
なし	内蔵	内蔵	シリアル
-h	シリアル	シリアル	シリアル
-D	内蔵、シリアルの両方	内蔵	シリアル
-Dh	内蔵、シリアルの両方	シリアル	シリアル
-P、キーボードが存在する場合	内蔵	内蔵	シリアル
-P、キーボードが存在しない場合	内蔵、シリアルの両方	シリアル	シリアル

17.6.4. シリアルコンソールを利用する上で役に立つ情報

シリアルポートの通信速度をもっと速いものに設定するには

デフォルトのシリアルポート通信速度は、9600　ボー、　8　ビット、パリティなし、ストップビット　1　です。　通信速度を変更したい場合には、少なくとも　ブートブロックの再コンパイルが必要になります。/etc/make.conf に次のような行を追加して、 新しくブートブロックをコンパイルして下さい。

```
BOOT_COMCONSOLE_SPEED=19200
```

もし、シリアルコンソールがブート時の　　-h　　オプション以外の方法で設定されていたり、カーネルが利用するシリアルコンソールが　　ブートブロック実行中のものと異なる場合には、

カーネルコンフィグレーションファイルに次のオプションを追加して、
新しくカーネルをコンパイルしなければなりません。

```
options CONSPEED=19200
```

sio0 以外のシリアルポートを コンソールとして使うには

sio0 以外のポートをコンソールとして使うには、再コンパイルが必要です。
それがどんな理由であれ、他のポートを使用する場合には ブートブロック、ブートローダ、カーネルを
次のようにして再コンパイルして下さい。

1. カーネルソースを取得する (FreeBSD のアップデートとアップグレード をご覧ください)。
2. /etc/make.conf を編集し、 BOOT_COMCONSOLE_PORT に 使用したいポートのアドレス(0x3F8、0x2F8、0x3E8 or 0x2E8)を 設定してください。使用可能なのは sio0 から sio3 (COM1 から COM4) までで、 マルチポートシリアルカードは使えません。
   また、ここで割り込みの設定をする必要はありません。
3. 設定を変更するために新たなカーネルコンフィグレーションファイルを作成し、
   使いたいシリアルポートのフラグを適切に設定します。 例えば、sio1 (COM2)
   をコンソールにしたければ、

   ```
   device sio1 at isa? port "IO_COM2" tty flags 0x10 irq 3
   ```

   または、

   ```
   device sio1 at isa? port "IO_COM2" tty flags 0x30 irq 3
   ```

   とします。その際、 他のシリアルポートにコンソールフラグをつけてはいけません。
4. ブートブロックを再コンパイルし、インストールする。

   ```
   # cd /sys/boot/i386/boot2
   # make
   # make install
   ```

5. ブートローダを再コンパイルし、インストールする。

   ```
   # cd /sys/boot/i386/loader
   # make
   # make install
   ```

6. カーネルを再構築し、インストールする。
7. disklabel(8) を使ってブートブロックをブートディスクに書き込み、
   新しいカーネルから起動する。

シリアルポートから **DDB** デバッガを起動するには

シリアルコンソールからカーネルデバッガを起動したい(これは
リモートで診断する際に便利ですが、もしおかしな BREAK
信号がシリアルポートに送られるような場合には危険です!)
場合には、次のオプションを使ってカーネルをコンパイルして下さい。

```
options BREAK_TO_DEBUGGER
options DDB
```

シリアルコンソールにログインプロンプトを表示させるには

シリアルコンソールからブートメッセージを確認したり、
シリアルコンソールを経由してカーネルデバッグセッションに入ることが
できるので、これは必要がないかもしれませんが、 *login* プロンプトをシリアルポートに
出力するように設定することもできます。 これには、次のようにします。

エディタで /etc/ttys というファイルを開き、 次に示す行に移動して下さい。

```
ttyd0 "/usr/libexec/getty std.9600" unknown off secure
ttyd1 "/usr/libexec/getty std.9600" unknown off secure
ttyd2 "/usr/libexec/getty std.9600" unknown off secure
ttyd3 "/usr/libexec/getty std.9600" unknown off secure
```

`ttyd0` から `ttyd3` は、 COM1 から COM4 に対応しています。 設定したいポートの `off` を `on` に変更して下さい。 また、もしシリアルポートの通信速度を変更しているなら、 `std.9600` が実際の通信速度になるように、 例えば `std.19200` のように変更して下さい。

さらに、実際のシリアル端末に合わせて、 端末タイプを `unknown` から変更することも可能です。

ファイルの編集が終了したら、 変更を有効化するために `kill -HUP 1` を実行しなければなりません。

17.6.5. ブートローダからコンソールを変更するには

前セクションは、ブートブロックの設定を変更することでシリアルコンソールを
セットアップする方法について解説していました。
このセクションでは、ブートローダへのコマンド入力と環境変数設定で
コンソールの指定を行なう方法を紹介します。 ブートローダがブートブロックの後、
ブートプロセスの第三ステージで呼び出されたとき、
ブートローダの設定には、ブートブロックの設定がそのまま使われます。

シリアルコンソールをセットアップする

ブートローダとカーネルに対して シリアルコンソールを使用するように設定するには、 単に
/boot/loader.rc のファイルに、次のような一行を書くだけで実現できます。

```
set console=comconsole
```

これは、前セクションで扱ったブートブロックの設定に 全く関係なく機能します。

上に示した行は、 /boot/loader.rc の最初の行に書き込まなくてはいけません。
これはできるだけ早く、ブートメッセージをシリアルコンソールに 出力させるために必要なことです。

同様にして、次のように内蔵コンソールを指定することもできます。

```
set console=vidconsole
```

もし、ブートローダの環境変数 `console` が設定されていない場合、
ブートローダ、そしてその次に起動するカーネルは ブートブロックで指定された `-h` オプションに示されたコンソールを使用します。

3.2 以降のバージョンにおいては /boot/loader.rc ではなく、 /boot/loader.conf.local や /boot/loader.conf にコンソール指定を書き込みます。 その場合、 /boot/loader.rc は次のようになっていなければなりません。

```
include /boot/loader.4th
start
```

それから、/boot/loader.conf.local を作成して、次の行をそこに追加して下さい。

```
console=comconsole
```

か、もしくは

```
console=vidconsole
```

です。詳細については、loader.conf(5) を参照して下さい。

その際、ブートローダはオプション指定なし (ブートブロックに `-P` オプションが指定されたのと等価)になり、 キーボードの存在を調べて内蔵コンソールとシリアルコンソールを自動的に選択する機能は働きません。

sio0 以外のシリアルポートを コンソールとして使うには

sio0 以外のシリアルポートを コンソールとして使うには、ブートローダを再コンパイルする必要があります。 それには、 sio0 以外のシリアルポートを コンソールとして使うには に書かれている説明にしたがって下さい。

17.6.6. 注意

シリアルコンソールというアイデアは、
グラフィック出力用のハードウェアやキーボードが接続されていない
専用サーバのセットアップを可能にするためのものです。
ほとんどのシステムはキーボードなしで起動できますが、

不幸にも、グラフィックアダプタなしでは起動できないシステムはたくさんあります。 AMI BIOS を採用しているマシンでは、CMOS 設定の `graphics adapter` を `Not Installed` にするだけで、グラフィックアダプタがなくとも起動できるように設定することができます。

しかしながら、多くのマシンはこのようなオプションを持っていませんし、
ディスプレイハードウェアがシステムに存在しないと起動しないように
なっています。そのようなマシンでは、 モニタを接続する必要がなかったとしても、
適当なグラフィックカード(モノクロのジャンク品でも構いません)を
挿入したままにしておく必要があるでしょう。 また、AMI BIOS をインストールする、という手もあります。

Chapter 18. PPP と SLIP

18.1. この章では

もしあなたがモデムを使ってインターネットに接続したり，他の人々に FreeBSD によるインターネットへのダイヤルアップ接続を 提供しようとしているのでしたら，PPP または SLIP 接続を選択することができます.

この節では 3 種類の PPP について説明しています．それは ユーザ，カーネル，そして *PPPoE* (PPP オーバイーサネット) です. また SLIP のクライアントとサーバの設定についても記述しています.

最初に説明するのは，ユーザ PPP です．ユーザ PPP は FreeBSD に 2.0.5-RELEASE の時に，既に存在していたカーネル実装の PPP に加えて導入されました.

ユーザ PPP とカーネル PPP の主な違いは何かと疑問に思われるかも 知れませんが, その答えは簡単です. ユーザ PPP はデーモンとしては実行されず 必要に応じて実行されるのです. PPP インタフェイスを組み込んだカーネルは 必要ではなく，ユーザプロセスとして実行されカーネルとのデータの やり取りにはトンネルデバイスドライバ (tun) を使用します.

この節ではこれ以降ユーザ PPP のことは，`pppd` のような他の PPP ソフトウエアと特に区別する必要がある場合を除いて，単に ppp と記述します. またこの節に記述されているコマンドは すべて root で実行されなければなりません.

18.2. ユーザ ppp の利用

18.2.1. ユーザ PPP

前提条件

以下の情報を手に入れておく必要があるでしょう:

- PPP で接続するインターネットサービスプロバイダ (ISP) のアカウント．さらに，接続済みのモデム (またはその他のデバイス) があり, プロバイダとの接続が可能なように正しく設定されている.
- プロバイダの電話番号.
- ログイン名とパスワード．これは通常の unix 形式のログイン名とパスワードの組という場合もありますし，PPP PAP または CHAP のログイン名とパスワードの組という場合もあります.
- 一つ以上のネームサーバの IP アドレス．通常，プロバイダから IP アドレスを二つ指示されているはずです．一つすら提供されていないならば，ppp.conf ファイル中で `enable dns` コマンドを使って ppp にネームサーバを設定するよう 指示できます.

プロバイダからは以下の情報が提供されているはずですが, どうしても必要というわけではありません:

- プロバイダのゲートウェイの IP アドレス．ゲートウェイとは，あなたがそこに接続をおこなって，デフォルトルート として設定することになるマシンです. プロバイダがこのアドレスを明示していなくても，最初は 適当に設定しておいて，

接続時にプロバイダの PPP サーバから 正しいアドレスを教えてもらうことができます.

このアドレスは, ppp から HISADDRとして参照されます.

- プロバイダのネットマスク設定. プロバイダが明示していないとしても, ネットマスクとして 255.255.255.0 を使用しておけば問題ありません.
- もしプロバイダから固定の IP アドレスとホスト名の割り当てを 受けていれば, その情報を指定しておくこともできます. 割り当てを受けていなければ, 接続先から適切な IP アドレスを指定してもらいます.

もし, 必要な情報が不足していれば, プロバイダに連絡を取って 確認しておいてください.

ppp 対応カーネルの構築

説明でも述べているように, ppp はカーネルの tun デバイスを使います. 使っているカーネルがどれであっても, tun デバイスを設定しなければなりません. FreeBSDに付属しているデフォルトの GENERIC カーネルに合うように tun デバイスは前もって設定されています. しかしながら, 自分で修正したカーネルをインストールするのであれば, pppが正しく動くよう, カーネルが設定されているか確認しなくてはいけません.

これを確認するには, カーネルコンパイルディレクトリ (/sys/i386/conf または /sys/pc98/conf) に移動して, カーネルコンフィグレーションファイルを調べます. 以下の行がどこかに含まれている必要があります.

```
pseudo-device tun 1
```

この行がカーネルコンフィグレーションファイルに 含まれていない場合, この行を追加して カーネルの再コンパイルとインストールをおこなう必要があります. 元々の GENERIC カーネルは 標準でこれを含んでいますので, カスタムカーネルをインストールしているのではなかったり, /sys ディレクトリが存在しないのであれば, 何も変更する必要はありません. カーネルコンフィグレーションの詳細については, FreeBSD カーネルのコンフィグレーション を参照してください.

以下のコマンドを実行することで, 現在のカーネルにトンネルデバイスが いくつ組み込まれているかを調べることができます:

```
# ifconfig -a
tun0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1500
    inet 200.10.100.1 --> 203.10.100.24 netmask 0xffffffff
tun1: flags=8050<POINTOPOINT,RUNNING,MULTICAST> mtu 576
tun2: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1500
    inet 203.10.100.1 --> 203.10.100.20 netmask 0xffffffff
tun3: flags=8010<POINTOPOINT,MULTICAST> mtu 1500
```

FreeBSD 4.0やより最近のリリースでは, すでに使われている tun デバイスしか見つけることが できないでしょう. これは, 全く tun

> デバイスを見つけることが できないかもしれないということです. しかし, もしこうなって しまっても, 心配することはありません. そのデバイスは ppp が使おうとする時に動的に作られるはず だからです.

この例ではトンネルデバイスが四つ存在し, そのうち二つに 設定がおこなわれ, 使用中であることがわかります. 上の例で RUNNING フラグがオンになっている ものがありますが, これはそのインタフェースが何かに使用されていることを示しているだけであるということに注意してください. つまり, RUNNING になっていないインタフェースがあったとしても, それはエラーではありません.

トンネルデバイスがカーネルに組み込まれておらず, 何らかの理由でカーネルの再構築ができない場合でも, 方法がないわけではありません. 動的にデバイスをロードすることができるはずです. 詳細については modload(8) や lkm(4) など, 適切なマニュアルを参照してください.

tun デバイスの確認

ほとんどのユーザは tun デバイス (/dev/tun0) が一つあれば充分でしょう. より多くのデバイスを使う場合 (すなわち, カーネルコンフィグレーション ファイルで pseudo-device tun の行に 1 以外の数値を指定している場合), 以下で tun0 と書かれている部分をすべて, あなたが使うデバイスの番号に あわせて読みかえてください.

tun0 デバイスが正しく作成されていることを確認する最も簡単な方法は, それを作り直すことです. そのためには, 以下のコマンドを実行します:

```
# cd /dev
# ./MAKEDEV tun0
```

カーネルに 16 個のトンネルデバイスを組み込んだのであれば, tun0 だけでなく他の tun デバイスも作成しておく必要があるでしょう:

```
# cd /dev
# ./MAKEDEV tun15
```

また, カーネルが正しく設定されているかどうかを調べるために 以下のコマンドを実行して, このような出力が得られることを確認します:

```
# ifconfig tun0
tun0: flags=8050<POINTOPOINT,RUNNING,MULTICAST> mtu 1500
```

まだ RUNNING フラグがセットされていない場合もあります. その時は以下のような出力が得られるでしょう:

```
# ifconfig tun0
tun0: flags=8010<POINTOPOINT,MULTICAST> mtu 1500
```

前述したように， FreeBSD 4.0 以降のリリースでは tun デバイスは要求に応じて 作られるので，もしそのデバイスがまだ使われていなければ，
見つけられないかもしれないということを思い出してください．

名前の解決に関する設定

リゾルバ (resolver) はシステムの一部分で， IP アドレスとホスト名との 変換をおこないます． IP アドレスとホスト名を対応させるためのマップを， 二つの場所のうちの一つから探すように設定できます．一つめは /etc/hosts (man 5 hosts) と呼ばれるファイルです． 二つめはインターネットドメインネームサービス (DNS) と呼ばれる 分散データベースですが， これに関する議論はこのドキュメントで扱う範囲を 越えていますので, これについての説明はおこないません．

リゾルバは名前のマッピングを おこなうシステムコールの集合体です． ただしどこからマッピング情報を見つけるのかは， 最初に指示しておく必要があります． これは まず /etc/host.conf ファイルを編集することでおこないます． 混乱の元になりますので， このファイルを /etc/hosts.confと 呼んだりしてはいけません (余分な s がついていますね)．

/etc/host.conf ファイルの編集

このファイルには 以下の 2 行が (この順番で) 書かれているはずです:

```
hosts
bind
```

これは， 最初に /etc/hosts ファイルを調べ， そこで目的の名前が 見つけられなかった場合に DNS を引きにいくようリゾルバに指示します．

/etc/hosts(5) ファイルの編集

このファイルはローカルネットワーク上に存在するマシンの IP アドレスとホスト名を含んでいるはずです． 最低でも ppp を動作させるマシンのエントリが含まれている必要があります． そのマシンのホスト名が foo.bar.com で， IP アドレスが 10.0.0.1 であると仮定すると, /etc/hosts は 以下の行を含んでいなければいけません:

```
127.0.0.1   localhost.bar.com   localhost
127.0.0.1   localhost.bar.com.
10.0.0.1    foo.bar.com     foo
10.0.0.1    foo.bar.com.
```

一つめの行は localhost を現在のマシンの別名として定義しています． マシン固有の IP アドレスが何であっても， この行の IP アドレスは 常に 127.0.0.1 でなければいけません．二つめの行はホスト名 foo.bar.com (と, その省略形 foo) を IP アドレス 10.0.0.1 にマップします．

もしプロバイダから固定の IP アドレスとホスト名を割り当てられて いるのであれば，それを 10.0.0.1 エントリのかわりに使ってください．

/etc/resolv.conf ファイルの編集

/etc/resolv.conf はリゾルバの振舞いを指定します. もし自前の DNS サーバを走らせているのなら, このファイルは空のままに しておくこともできます. 通常は, 以下のように書いておく必要があるでしょう:

```
domain bar.com
nameserver x.x.x.x
nameserver y.y.y.y
```

`x.x.x.x` と `y.y.y.y` はプロバイダから指示されたアドレスで, 接続するプロバイダが提供しているネームサーバを すべて書いてください. `domain` に指定するのは このマシンのデフォルトのドメイン名で, おそらく 書かなくても問題は無いでしょう. このファイルの各エントリの詳細については, resolv.conf のマニュアルページを参照してください.

バージョン 2 以降の ppp を使用している場合には, `enable dns` コマンドを使用してネームサーバのアドレスを プロバイダに問い合わせるように指示することができます. 上の指定とは異なるアドレスをプロバイダが指定してきた場合 (または /etc/resolv.conf でネームサーバが指定されていない場合), ppp はプロバイダが指定したアドレスで resolv.conf を書きかえます.

`ppp` の設定

ユーザ ppp と `pppd` (カーネルレベルの PPP 実装) は どちらも /usr/shared/examples/ppp ディレクトリに置かれた設定ファイルを使います. ここには設定ファイルのサンプルが用意されていて, ユーザ ppp の設定を おこなう際に大変参考になりますので, 削除したりしないでください.

`ppp` の設定をするためには, 必要に応じていくつかのファイルを編集する必要が あります. 書き込む内容は, プロバイダが静的に IP アドレスを割り当てる (つまり, 固定の IP アドレスを一つ与えられて, 常にそれを使う) か, または動的に IP アドレスを割り当てる (つまり, PPP セッションごとに IP アドレスが変化する可能性がある) かということに ある程度依存します.

静的 IP アドレスによる PPP 接続

まず /etc/ppp/ppp.conf という設定ファイルを作成する必要があります. これは以下の例とほとんど同じようなものになるでしょう.

: で終る行は 1 カラム目から始め, その他の行はスペースまたはタブで以下の例のように 段をつける (インデントする) 必要があります.

```
1     default:
2       set device /dev/cuaa0
3       set speed 115200
4       set dial "ABORT BUSY ABORT NO\\sCARRIER TIMEOUT 5 \"\" ATE1Q0 OK-AT-OK
\\dATDT\\TTIMEOUT 40 CONNECT"
5     provider:
6       set phone "(123) 456 7890"
7       set login "TIMEOUT 10 \"\" \"\" gin:--gin: foo word: bar col: ppp"
```

```
8       set timeout 300
9       set ifaddr x.x.x.x y.y.y.y 255.255.255.0 0.0.0.0
10      add default HISADDR
11      enable dns
```

ファイルでは行番号を取り除いておいてください.
これは解説の際に参照する行を示すためにつけたものです.

Line 1

デフォルトエントリを指定します. このエントリ中のコマンドは ppp が起動された際に自動的に実行されます.

Line 2

モデムが接続されているデバイスを指定します. COM1: は /dev/cuaa0 に, COM2: は /dev/cuaa1 になります.

Line 3

通信速度 (DTE 速度) を指定します. もし 115200 が使えない (最近のモデムなら大抵使えるはずですが) 場合には, かわりに 38400 を指定してみてください.

Line 4

ダイアルスクリプトを指定します. ユーザ PPP は chat(8) 言語に似た, 受信待ち文字列と送信文字列の対からなるスクリプトを使用します. この言語の機能に関しては, マニュアルページを参照してください.

Line 5

接続するプロバイダの名前 "provider" を エントリ名として指定します.

Line 6

このプロバイダの電話番号を指定します. 複数の電話番号を : や | で区切って指定することができます. これら区切り文字の違いについては, ppp(8) に 詳しく書かれています. 要約すると, 毎回違う番号にかけたいのであれば : を使います. 常に まず先頭の番号にかけてみて, つながらない時にだけ 2 番目以降の番号に かけたいのであれば | を使います. 例に示されているように, 常に電話番号全体を引用符で くくって (クォートして) おきます.

Line 7

ダイアルスクリプトと同様に, ログインスクリプトも chat 言語風の記述をおこないます. この例は, 以下のような ログインセッションを使用する プロバイダのためのものです:

```
J. Random Provider
login: foo
password: bar
protocol: ppp
```

このスクリプトは必要に応じて 書きかえなければならないでしょう. 初めてスクリプトを書く時には, 予想した通りに 処理が進んだかどうかを確認するため, "chat" ログを とるようにしておいた方が良いでしょう.

PAP や CHAP を使用する場合には, ここでログインすることは ありませんから, ログイン文字列は空白のままにしておくべきです. 詳細については PAP および CHAP による認証を参照してください.

Line 8

デフォルトの接続タイムアウト時間を (秒数で) 指定します. この例では, 300 秒間 通信がおこなわれなければ 自動的に接続を切るように指定しています. タイムアウトさせたくない場合には, この値を 0 に設定します.

Line 9

インタフェースのアドレスを指定します. 文字列 *x.x.x.x* は プロバイダに割り当てられた IP アドレスで置きかえてください. 文字列 *y.y.y.y* はプロバイダから指示されたゲートウェイ (接続先となるマシン) の IP アドレスで置きかえてください. プロバイダがゲートウェイのアドレスを 指示していない場合は, `10.0.0.2/0` を使用しておいてください. もし "仮の" アドレスを使用する必要がある場合には, 動的 IP アドレスによる PPP 接続に関する指示に従って, /etc/ppp/ppp.linkup にエントリを作成していることを 確認してください. この行が省略されている場合, ppp を `-auto` モードで動作させることはできません.

Line 10

プロバイダのゲートウェイへの経路を デフォルトルートとして 追加します. 特殊文字列 `HISADDR` は, 9 行目で指定された ゲートウェイのアドレスで置きかえられます. `HISADDR` は 9 行目までは初期化されていませんので, その行よりも後でしか使えないことに 注意してください.

Line 11

ネームサーバのアドレスが正しいか どうかを確認するため, プロバイダに問い合わせをおこなうよう ppp に指示します. プロバイダがこの機能をサポートしていれば, ppp は /etc/resolv.conf のネームサーバエントリを 正しいアドレスに更新することができます.

静的な IP アドレスを持っていて, 接続が完了する前にルーティングテーブルの エントリが正しく設定されているのであれば, ppp.linkup に エントリを追加する必要はありません. しかし, この場合でもエントリを追加して, 接続が完了した時点で プログラムを呼び出したいことがあるかもしれません. これについては後ほど sendmail を例として説明します.

これらの設定ファイルのサンプルが /usr/shared/examples/ppp ディレクトリに 置かれています.

動的 IP アドレスによる PPP 接続

プロバイダが静的な IP アドレスの割り当てをおこなっていない場合, `ppp` が相手側のホスト (ゲートウェイ) と交渉して, こちら側と相手側のアドレスを 決めるように設定することができます. これは, 起動時には"仮の"アドレスを使っておいて, 接続後に IP コンフィグレーション プロトコル (IPCP) を使用して `ppp` が IP アドレスを正しく設定できるようにすることで実現されます. 静的 IP アドレスによる PPP 接続に 以下の変更を加える以外は, ppp.conf の設定は同じです:

```
9       set ifaddr 10.0.0.1/0 10.0.0.2/0 255.255.255.0
```

繰り返しますが, 行番号は取り除いておいてください. これは解説の際に参照する行を示すためにつけたものです. なお, 少なくともスペース 1 個分の段づけ

(インデント) が必要です.

Line 9

/ 文字の後ろの数字は, アドレス交渉の際に固定しておきたい ビットの数です. 場合によっては, もっと適切な IP アドレスを 指定しておきたいこともあるかもしれませんが, ほとんどの場合には上の例の通りで問題ありません.

最後の引数 (0.0.0.0) は, アドレスの交渉の際に 10.0.0.1 ではなく 0.0.0.0 を使用するよう ppp に指示するためのものです. set ifaddr コマンドの最初の引数として 0.0.0.0 を指定してはいけません. さもないと, -auto モードで動作させる際に 初期経路を設定することができなくなります.

バージョン 1.X の ppp を使用する場合, /etc/ppp/ppp.linkup にもエントリを作成しておく必要があります. ppp.linkup は接続が確立された後に使用されます. この時点では, ppp は実際にどの IP アドレスを使うべきなのか わかっているはずです. 以下のエントリは存在する仮の経路を削除し, 正しい経路を作成します:

```
1     provider:
2       delete ALL
3       add default HISADDR
```

Line 1

接続を確立する際に, ppp は以下のルールに従って ppp.linkup のエントリを検索します: まず ppp.conf で使用されたのと同じラベルを探します. もし見つからなければ, ゲートウェイの IP アドレスのエントリを 探します. このエントリは 4 オクテットの IP アドレス形式の ラベルです. それでも まだエントリが見つからなければ, MYADDR エントリを探します.

Line 2

この行は, 使用する tun インタフェースに関する既存の経路を (ダイレクトルートのエントリを除き) すべて削除するよう ppp に指示します.

Line 3

この行は HISADDR への経路をデフォルトルートとして 追加するように ppp に指示します. HISADDR は IPCP で 決定されたゲートウェイの IP アドレスで置きかえられます.

詳細なサンプルについては, /usr/shared/examples/ppp/ppp.conf.sample ファイル中のpmdemand エントリと /usr/shared/examples/ppp/ppp.linkup.sample を参照してください.

バージョン 2 の ppp から "sticky routes" が導入されました. MYADDR や HISADDR を含む add コマンドと delete コマンドを記憶して, MYADDR や HISADDR の アドレスが変化した際には経路の再設定をおこないます. したがって, これらのコマンドを ppp.linkup に繰り返し記述する必要は無くなりました.

かかってきた電話を ppp で受けるには

かかってきた電話を ppp が受けるように設定する際に, そのマシンが LAN に接続されているのであれば, パケットを LAN に転送するかどうかを決定する必要があります. 転送をおこなう場合には, その LAN のサブネットから IP アドレスを ppp クライアントに割り当て, 以下のコマンドを指定するのが良いでしょう.

```
gateway_enable=YES
```

どの **getty** を使いますか**?**

getty でダイアルアップサービスをおこなう場合の優れた解説が FreeBSD でダイアルアップサービスをおこなうための設定にあります.

getty に代わるものとしては, mgetty があります. これは getty をより柔軟にしたもので, ダイアルアップ回線での使用を意図して 設計されています.

mgetty を使う場合の利点は, mgetty が積極的にモデムと通信する ということです. つまり, もし /etc/ttys でポートを閉じている場合, モデムは電話をとらなくなります.

最近のバージョンの mgetty (0.99beta 以降) では, PPP ストリームの 自動検出もサポートされています. これにより, クライアント側で スクリプトを準備しなくてもサーバに アクセスすることができます.

mgetty に関する, より詳細な情報については Mgetty と AutoPPP を参照してください.

ppp の実行許可

ppp は通常, ID 0 のユーザ (root) として動作しなければいけませんが, 以下で説明するように, ppp を通常のユーザとしてサーバモードで実行させたい 場合には, そのユーザを /etc/group の network グループに 追加して, ppp を実行する許可を与えておかなければいけません.

また, そのユーザが設定ファイル内の目的のエントリに アクセスできるように, 以下のように allow コマンドで許可を与えておく必要があります:

```
allow users fred mary
```

このコマンドがデフォルトエントリに 書かれている場合には, 指定されたユーザは すべてのエントリをアクセスできるようになります.

動的 **IP** ユーザのための **ppp** シェルの設定

/etc/ppp/ppp-shell という名前で, 以下のような内容のファイルを 作成します:

```
#!/bin/sh
IDENT=`echo $0 | sed -e 's/^.*-\(.*\)$/\1/'`
CALLEDAS="$IDENT"
TTY=`tty`

if [ x$IDENT = xdialup ]; then
    IDENT=`basename $TTY`
fi

echo "PPP for $CALLEDAS on $TTY"
echo "Starting PPP for $IDENT"
```

```
exec /usr/sbin/ppp -direct $IDENT
```

このスクリプトには実行可能属性をつけておきます. 次に, 以下のコマンドを実行し, ppp-dialup という名前で このスクリプトへのリンクを作成します:

```
# ln -s ppp-shell /etc/ppp/ppp-dialup
```

すべてのダイアルアップ ppp ユーザのログインシェルとして このスクリプトを使用します. 以下は pchilds というユーザ名の ダイアルアップユーザを /etc/password へ登録した場合の例です. (パスワードファイルを直接エディタで編集したりせず, vipw を使ってください)

```
pchilds:*:1011:300:Peter Childs PPP:/home/ppp:/etc/ppp/ppp-dialup
```

任意のユーザが読むことのできる, /home/ppp ディレクトリを 作成します. /etc/motd が表示されないようにするため, このディレクトリには以下のように大きさが 0 バイトのファイルを作成しておきます.

```
-r--r--r--   1 root     wheel           0 May 27 02:23 .hushlogin
-r--r--r--   1 root     wheel           0 May 27 02:22 .rhosts
```

静的 **IP** ユーザのための **PPP** シェルの設定

上記と同じように ppp-shell ファイルを作成し, 静的な IP アドレスを割り当てるアカウントそれぞれについて ppp-shell へのシンボリックリンクを作成します.

例えば, クラス C ネットワークの経路制御を必要とする, 三人のダイアルアップユーザ fred, sam, mary がいるとすると, 以下のコマンドを実行することになります:

```
# ln -s /etc/ppp/ppp-shell /etc/ppp/ppp-fred
# ln -s /etc/ppp/ppp-shell /etc/ppp/ppp-sam
# ln -s /etc/ppp/ppp-shell /etc/ppp/ppp-mary
```

これらのユーザのダイアルアップアカウントでは, 上で作成した それぞれのシンボリックリンクをログインシェルとして設定しておきます. (つまり, ユーザ mary のログインシェルは /etc/ppp/ppp-mary になります).

動的 **IP** ユーザのための **ppp.conf** の設定

/etc/ppp/ppp.conf ファイルは, 大体以下のような内容になるでしょう:

```
default:
  set debug phase lcp chat
  set timeout 0

ttyd0:
```

```
  set ifaddr 203.14.100.1 203.14.100.20 255.255.255.255
  enable proxy

ttyd1:
  set ifaddr 203.14.100.1 203.14.100.21 255.255.255.255
  enable proxy
```

上の例のように段をつける (インデントする) 必要があることに注意してください.

default: エントリはセッションごとにロードされます. /etc/ttys で有効にしてある各ダイアルアップ回線ごとに一つ, 上記の ttyd0: のようなエントリを作成します. 各行の相手側アドレスとして, それぞれ別の IP アドレスを 動的 IP ユーザのための IP アドレスのプールから割り当てておく必要があります.

静的 **IP** ユーザのための **ppp.conf** の設定

上のサンプルの /usr/shared/examples/ppp/ppp.conf の内容に加えて, 静的に IP を割り当てられたダイアルアップユーザ それぞれのためのエントリを追加する必要があります. ここでも fred, sam, mary の例を使うことにしましょう.

```
fred:
  set ifaddr 203.14.100.1 203.14.101.1 255.255.255.255

sam:
  set ifaddr 203.14.100.1 203.14.102.1 255.255.255.255

mary:
  set ifaddr 203.14.100.1 203.14.103.1 255.255.255.255
```

必要であれば, それぞれの静的 IP ユーザに対する経路制御情報も /etc/ppp/ppp.linkup ファイルに書いておくべきでしょう. 以下の例ではクライアントの PPP リンクを経由する, クラス C の 203.14.101.0 ネットワークへの経路を追加しています.

```
fred:
  add 203.14.101.0 netmask 255.255.255.0 HISADDR

sam:
  add 203.14.102.0 netmask 255.255.255.0 HISADDR

mary:
  add 203.14.103.0 netmask 255.255.255.0 HISADDR
```

mgetty, **AutoPPP**, マイクロソフト拡張の詳細

mgetty と AutoPPP

AUTO_PPP オプションつきでコンパイルした mgetty を使えば, mgetty が PPP 接続の LCP

フェーズを検出して, 自動的に PPP シェルを起動するように 設定することができます. しかし この場合, デフォルトの login/password シーケンスは発生しないので, ユーザの認証は PAP または CHAP を使っておこなう必要があります.

このセクションでは, ユーザ (あなた) が問題なく AUTO_PPP オプションつきの mgetty (v0.99beta またはそれ以降) の設定, コンパイル, インストールができているものと仮定しています.

/usr/local/etc/mgetty+sendfax/login.config ファイルが 以下の行を含んでいることを確認してください:

```
/AutoPPP/ -     -       /etc/ppp/ppp-pap-dialup
```

これにより, PPP 接続を検出したら mgetty が ppp-pap-dialup スクリプトを実行するようになります.

/etc/ppp/ppp-pap-dialup という名前で, 以下のような内容のファイルを 作成します (このファイルには実行可能属性を つけておく必要があります):

```
#!/bin/sh
exec /usr/sbin/ppp -direct pap
```

さらに, かかってきた電話すべてを自分で扱うエントリを /etc/ppp/ppp.conf に作成します.

```
pap:
  enable pap
  set ifaddr 203.14.100.1 203.14.100.20-203.14.100.40
  enable proxy
```

この方法でログインする それぞれのユーザは, PAP によるユーザ認証を おこなうために /etc/ppp/ppp.secret ファイルにユーザ名とパスワードを 書いておくか, または /etc/password ファイルを使うように,

```
enable passwdauth
```

ユーザに静的な IP アドレスを割り当てる場合には, そのアドレスを /etc/ppp/ppp.secret の第三引数として指定することができます. サンプルについては, /usr/shared/examples/ppp/ppp.secret.sample を参照してください.

マイクロソフト拡張

クライアントからの要求に応じて, ppp が DNS や NetBIOS ネームサーバの アドレスを通知するように設定をおこなうこともできます.

バージョン 1.X の ppp で これらの拡張機能を有効にするには, 以下の行を /etc/ppp/ppp.conf の適切なセクションに追加する必要があるでしょう.

```
enable msext
set ns 203.14.100.1 203.14.100.2
```

```
set nbns 203.14.100.5
```

バージョン 2 以降の ppp では, 以下のようになります:

```
accept dns
set dns 203.14.100.1 203.14.100.2
set nbns 203.14.100.5
```

これにより, クライアントはプライマリと セカンダリのネームサーバアドレス および NetBIOS ネームサーバホストを知ることができます.

バージョン 2 以降の ppp では, `set dns` の行を省略した場合には /etc/resolv.conf に書かれているネームサーバのアドレスを使用します.

PAP および CHAP による認証

いくつかのプロバイダでは, PAP または CHAP のいずれかの認証メカニズムを使用して接続時の認証をおこなうように システムを設定しています. この場合, プロバイダは接続の際に `login:` プロンプトを送信せず, 最初から PPP で通信を始めようとするでしょう.

PAP ではパスワードがそのまま送られてしまうため, CHAP に比べると安全性が 低くなりますが, このパスワードはシリアル回線のみを通して送られます. そのため, クラッカーが "盗み聞き" する余地は多くないので, 通常ここの セキュリティは問題にはなりません.

静的 IP アドレスによる PPP 接続または 動的 IP アドレスによる PPP 接続の セクションに戻って, 以下の変更をおこないます:

```
7       set login
...
12      set authname MyUserName
13      set authkey MyPassword
```

これまでと同様に, 行番号は取り除いておいてください. これは解説の際に参照する行を示すためにつけたものです. なお, 少なくともスペース 1 個分の段づけ (インデント) が必要です.

Line 7

PAP または CHAP を使用する場合, 通常 プロバイダはサーバへの ログインを必要としません. そのため, "set login" 文字列を 無効にしておかなければいけません.

Line 12

この行は PAP/CHAP ユーザ名を指定します. *MyUserName* に 正しい値を入れておく必要があります.

Line 13

この行は PAP/CHAP パスワードを指定します. *MyPassword* に 正しい値を入れておく必要があります. PAP と CHAP はデフォルトで両方とも 受け付けられるようになって いますが, PAP や CHAP を使用するという 意思を明示するために,

```
15      accept PAP
```

または

```
15      accept CHAP
```

という行を追加しておくのも良いでしょう.

動作中の **ppp** の設定変更

適切な診断ポートが設定されている場合には, バックグラウンドで動作中の ppp プログラムと通信することができます. この設定をおこなうためには, 以下の行を設定ファイルに追加しておきます:

```
set server /var/run/ppp-tun%d DiagnosticPassword 0177
```

これにより, ppp は指定された unix ドメインの ソケットをモニタして, クライアントから正しいパスワードを受け取った後に アクセスを許可します. このソケット名に含まれる %d は, この ppp が使用している tun デバイスの デバイス番号で置きかえられます.

一旦ソケットの設定が終了したら, スクリプト中で pppctl(8) を 使用して, 動作中の ppp を操作することができるでしょう.

システムの最終設定

これで ppp の設定は終りました. しかし ppp を動かす前に, まだ少し必要なことがあります. それらの設定は, すべて /etc/rc.conf ファイルを 編集することでおこないます. (このファイルは以前には /etc/sysconfig と呼ばれていました)

このファイルを上から順に設定していきます. まずは hostname= の行が設定されていることを確認します. 例えば以下のように:

```
hostname="foo.bar.com"
```

もしプロバイダが静的な IP アドレスとホスト名を割り当てているのなら, ホスト名としてそれを使うのがおそらくベストでしょう.

次に network_interfaces 変数を調べます. 必要に応じて (on demand) プロバイダにダイアルするようにシステムを設定したい場合には, tun0 デバイスがこのリストに追加されていることを確認しておきます. それ以外の場合には, tun0 デバイスをリストから削除しておきます.

```
network_interfaces="lo0 tun0" ifconfig_tun0=
```

ifconfig_tun0 変数が空で, /etc/start_if.tun0 という名前のファイルが作成されていなければなりません.
このファイルの内容は以下のようになります.

```
ppp -auto mysystem
```

このスクリプトはネットワークの設定時に実行され, ppp デーモンを自動モードで立ち上げます. このマシンがもし LAN のゲートウェイであれば, -alias スイッチも使用したいと思うかもしれません. 詳細に関しては, マニュアルページを参照してください.

以下のようにルータプログラムを NO に設定します.

```
router_enable="NO"
```

routed は, ppp が作成したデフォルトのルーティングテーブルエントリを削除してしまう場合がありますので, (初期設定では起動されるようになっている) routed デーモンが 起動されないようにしておくことが重要です.

sendmail_flags 行が -q オプションを含まないように 設定しておいた方がよいでしょう. さもないと, sendmail が アドレスを調べようとして発信をおこなってしまう場合があります. 以下のような設定で良いでしょう:

```
sendmail_flags="-bd"
```

この結果, PPP リンクを立ち上げた時には いつでも以下のコマンドを実行して, キューにたまっているメールを sendmail に送信させる作業が必要になるでしょう.

```
# /usr/sbin/sendmail -q
```

ppp.linkup 中で !bg コマンドを使用することで, これを自動的に おこなうこともできます:

```
1     provider:
2       delete ALL
3       add 0 0 HISADDR
4       !bg sendmail -bd -q30m
```

こうするのが嫌であれば, SMTP トラフィックをブロックするように "dfilter" を設定しておくこともできます. 詳細についてはサンプルファイルを参照してください.

後はマシンをリブートするだけです.

リブートが終ったら,

```
# ppp
```

コマンドを実行し, 続いて PPP セッションを開始させるために dial provider と入力することもできますし, (start_if.tun0 スクリプトを作成していない場合に), 外部へのトラフィックが発生した時に, ppp が自動的にセッションを確立してくれるようにしたいのであれば, 以下のコマンドを実行することもできます.

```
# ppp -auto provider
```

まとめ

要約すると, 初めて ppp を設定する際には, 以下のステップが不可欠です:

クライアント側:

1. カーネルに tun デバイスが組み込まれていることを確認.
2. /dev ディレクトリに tunX デバイスファイルが 存在することを確認.
3. /etc/ppp/ppp.conf にエントリを作成. ほとんどのプロバイダでは, pmdemand の例で充分でしょう.
4. 動的 IP アドレスを使用するなら, /etc/ppp/ppp.linkup に エントリを作成.
5. /etc/rc.conf (または sysconfig) ファイルを更新.
6. 必要に応じてダイヤル (demand dialing) したいのであれば, start_if.tun0 スクリプトを作成.

サーバ側:

1. カーネルに tun デバイスが組み込まれていることを確認.
2. /dev ディレクトリに tunX デバイスファイルが 存在することを確認.
3. (vipw(8) コマンドを使って) /etc/passwd にエントリを作成.
4. このユーザのホームディレクトリに ppp -direct direct-server か何かを実行するプロファイルを作成.
5. /etc/ppp/ppp.conf にエントリを作成. direct-server の例で充分でしょう.
6. /etc/ppp/ppp.linkup にエントリを作成.
7. /etc/rc.confファイルを更新.

18.3. カーネル PPP の利用

18.3.1. カーネル PPP の設定

PPP の設定を始める前に, pppd が /usr/sbin にあり, また /etc/ppp という

ディレクトリが存在することを確認してください.

pppd はふたつのモードで動作します.

1. "クライアント" モード. シリアル接続やモデムを利用して, そのマシンを 外部のネットワークに PPP 接続したい場合に用います.
2. "サーバ" モード. そのマシンがネットワーク上にあるときに, PPP を使って ほかのコンピュータを接続する際に用います.

どちらの場合でも, オプションファイルを設定する必要があります (/etc/ppp/options または, そのマシン上で PPP を使用する人が 複数いる場合には ~/.ppprc).

また, ダイヤルとリモートホストへの接続をおこなうために, シリアル接続やモデムを 操作する, なんらかのソフトウェアが必要です (kermit が適しているでしょう).

18.3.2. PPP クライアントとしての動作

わたしは, CISCO ターミナルサーバの PPP 回線に接続するために, 下記のような /etc/ppp/options を使用しています.

```
crtscts         # enable hardware flow control
modem           # modem control line
noipdefault     # remote PPP server must supply your IP address.
                # if the remote host doesn't send your IP during IPCP
                # negotiation , remove this option
passive         # wait for LCP packets
domain ppp.foo.com      # put your domain name here

:<remote_ip>    # put the IP of remote PPP host here
                # it will be used to route packets via PPP link
                # if you didn't specified the noipdefault option
                # change this line to <local_ip>:<remote_ip>

defaultroute    # put this if you want that PPP server will be your
                # default router
```

接続方法:

1. kermit (またはその他のモデム操作プログラム) を使ってリモートホストに ダイヤルし, 接続してください. そして, あなたのユーザ名とパスワード (必要 であれば, その他にもリモートホストで PPP を有効にするための操作) を入力 します.
2. kermit を抜けてください. (回線を切断せずに)
3. 下記のように入力します:

```
# /usr/src/usr.sbin/pppd.new/pppd /dev/tty01 19200
```

(通信速度とデバイス名には, あなたの環境に適したものを入れてください)

これでこのコンピュータは PPP で接続されました. もし, なんらかの理由で 接続に失敗したならば, /etc/ppp/options ファイルに debug オプションを追加して, 問題点を突き止めるために, コンソールに表示される メッセージを調べてください.

下記の /etc/ppp/pppup スクリプトは, 上記の作業を すべて自動的におこないます:

```
#!/bin/sh
ps ax |grep pppd |grep -v grep
pid=`ps ax |grep pppd |grep -v grep|awk '{print $1;}'`
if [ "X${pid}" != "X" ] ; then
        echo 'killing pppd, PID=' ${pid}
        kill ${pid}
fi
ps ax |grep kermit |grep -v grep
pid=`ps ax |grep kermit |grep -v grep|awk '{print $1;}'`
if [ "X${pid}" != "X" ] ; then
        echo 'killing kermit, PID=' ${pid}
        kill -9 ${pid}
fi

ifconfig ppp0 down
ifconfig ppp0 delete

kermit -y /etc/ppp/kermit.dial
pppd /dev/tty01 19200
```

/etc/ppp/kermit.dial は kermit 用のスクリプトで, ダイヤルして, リモートホストでの認証に必要なすべての処理をおこないます. (そのようなスクリプトの例は この文書の終わりに添付してあります)

PPP 接続を切断するには, 下記のような /etc/ppp/pppdown スクリプトを 使用します:

```
#!/bin/sh
pid=`ps ax |grep pppd |grep -v grep|awk '{print $1;}'`
if [ X${pid} != "X" ] ; then
        echo 'killing pppd, PID=' ${pid}
        kill -TERM ${pid}
fi

ps ax |grep kermit |grep -v grep
pid=`ps ax |grep kermit |grep -v grep|awk '{print $1;}'`
if [ "X${pid}" != "X" ] ; then
        echo 'killing kermit, PID=' ${pid}
        kill -9 ${pid}
fi
```

```
/sbin/ifconfig ppp0 down
/sbin/ifconfig ppp0 delete
kermit -y /etc/ppp/kermit.hup
/etc/ppp/ppptest
```

PPP が動作中かどうかを調べます (/usr/etc/ppp/ppptest):

```
#!/bin/sh
pid=`ps ax| grep pppd |grep -v grep|awk '{print $1;}'`
if [ X${pid} != "X" ] ; then
        echo 'pppd running: PID=' ${pid-NONE}
else
        echo 'No pppd running.'
fi
set -x
netstat -n -I ppp0
ifconfig ppp0
```

モデム回線を切断します (/etc/ppp/kermit.hup):

```
set line /dev/tty01 ; put your modem device here
set speed 19200
set file type binary
set file names literal
set win 8
set rec pack 1024
set send pack 1024
set block 3
set term bytesize 8
set command bytesize 8
set flow none

pau 1
out +++
inp 5 OK
out ATH0\13
echo \13
exit
```

次は `kermit` の代わりに `chat` を使う方法です.

pppd 接続を確立するためには, 次の二つのファイルの設定だけで十分です.

/etc/ppp/options:

```
/dev/cuaa1 115200

crtscts     # enable hardware flow control
```

```
modem       # modem control line
connect "/usr/bin/chat -f /etc/ppp/login.chat.script"
noipdefault # remote PPP serve must supply your IP address.
            # if the remote host doesn't send your IP during
                # IPCP negotiation, remove this option
passive         # wait for LCP packets
domain <your.domain>  # put your domain name here

:       # put the IP of remote PPP host here
            # it will be used to route packets via PPP link
                # if you didn't specified the noipdefault option
                # change this line to <local_ip>:<remote_ip>

defaultroute    # put this if you want that PPP server will be
            # your default router
```

/etc/ppp/login.chat.script:

(実際には一行になります.)

```
ABORT BUSY ABORT 'NO CARRIER' "" AT OK ATDT<phone.number>
 CONNECT "" TIMEOUT 10 ogin:-\\r-ogin: <login-id>
 TIMEOUT 5 sword: <password>
```

正しくインストールし編集した後は, 必要な事はこれだけです

```
# pppd
```

このサンプルは主に Trev Roydhouse <Trev.Roydhouse@f401.n711.z3.fidonet.org> から寄せられた情報に基づいており, 承諾を得て使用しています.

18.3.3. PPP サーバとしての動作

/etc/ppp/options:

```
crtscts                         # Hardware flow control
netmask 255.255.255.0           # netmask ( not required )
192.114.208.20:192.114.208.165  # ip's of local and remote hosts
                                # local ip must be different from one
                                # you assigned to the ethernet ( or other )
                                # interface on your machine.
                                # remote IP is ip address that will be
                                # assigned to the remote machine
domain ppp.foo.com              # your domain
passive                         # wait for LCP
modem                           # modem line
```

下記のような /etc/ppp/pppserv スクリプトで, そのマシンを PPP サーバにすることができます.

```
#!/bin/sh
ps ax |grep pppd |grep -v grep
pid=`ps ax |grep pppd |grep -v grep|awk '{print $1;}'`
if [ "X${pid}" != "X" ] ; then
        echo 'killing pppd, PID=' ${pid}
        kill ${pid}
fi
ps ax |grep kermit |grep -v grep
pid=`ps ax |grep kermit |grep -v grep|awk '{print $1;}'`
if [ "X${pid}" != "X" ] ; then
        echo 'killing kermit, PID=' ${pid}
        kill -9 ${pid}
fi

# reset ppp interface
ifconfig ppp0 down
ifconfig ppp0 delete

# enable autoanswer mode
kermit -y /etc/ppp/kermit.ans

# run ppp
pppd /dev/tty01 19200
```

PPP サーバを終了するには, この /etc/ppp/pppservdown スクリプト を使用します:

```
#!/bin/sh
ps ax |grep pppd |grep -v grep
pid=`ps ax |grep pppd |grep -v grep|awk '{print $1;}'`
if [ "X${pid}" != "X" ] ; then
        echo 'killing pppd, PID=' ${pid}
        kill ${pid}
fi
ps ax |grep kermit |grep -v grep
pid=`ps ax |grep kermit |grep -v grep|awk '{print $1;}'`
if [ "X${pid}" != "X" ] ; then
        echo 'killing kermit, PID=' ${pid}
        kill -9 ${pid}
fi
ifconfig ppp0 down
ifconfig ppp0 delete

kermit -y /etc/ppp/kermit.noans
```

下記の kermit スクリプトは, モデムの自動応答機能を有効, または無効にします (/etc/ppp/kermit.ans):

```
set line /dev/tty01
set speed 19200
set file type binary
set file names literal
set win 8
set rec pack 1024
set send pack 1024
set block 3
set term bytesize 8
set command bytesize 8
set flow none

pau 1
out +++
inp 5 OK
out ATH0\13
inp 5 OK
echo \13
out ATS0=1\13   ; change this to out ATS0=0\13 if you want to disable
                ; autoanswer mod
inp 5 OK
echo \13
exit
```

この /etc/ppp/kermit.dial スクリプトは, リモートホストに ダイヤルし, 認証手続きをするのに使用します. あなたは必要に応じて, これを 変更しないといけないでしょう. あなたのユーザ名とパスワードをこのスクリプトに書かなければいけませんし, モデムやリモートホストからの 応答によっては, 入力待ちの文を変更する必要もあります.

```
;
; put the com line attached to the modem here:
;
set line /dev/tty01
;
; put the modem speed here:
;
set speed 19200
set file type binary            ; full 8 bit file xfer
set file names literal
set win 8
set rec pack 1024
set send pack 1024
set block 3
set term bytesize 8
set command bytesize 8
set flow none
set modem hayes
set dial hangup off
set carrier auto                ; Then SET CARRIER if necessary,
```

```
set dial display on             ; Then SET DIAL if necessary,
set input echo on
set input timeout proceed
set input case ignore
def \%x 0                       ; login prompt counter
goto slhup

:slcmd                          ; put the modem in command mode
echo Put the modem in command mode.
clear                           ; Clear unread characters from input buffer
pause 1
output +++                      ; hayes escape sequence
input 1 OK\13\10                ; wait for OK
if success goto slhup
output \13
pause 1
output at\13
input 1 OK\13\10
if fail goto slcmd              ; if modem doesn't answer OK, try again

:slhup                          ; hang up the phone
clear                           ; Clear unread characters from input buffer
pause 1
echo Hanging up the phone.
output ath0\13                  ; hayes command for on hook
input 2 OK\13\10
if fail goto slcmd              ; if no OK answer, put modem in command mode

:sldial                         ; dial the number
pause 1
echo Dialing.
output atdt9,550311\13\10               ; put phone number here
assign \%x 0                    ; zero the time counter

:look
clear                           ; Clear unread characters from input buffer
increment \%x                   ; Count the seconds
input 1 {CONNECT }
if success goto sllogin
reinput 1 {NO CARRIER\13\10}
if success goto sldial
reinput 1 {NO DIALTONE\13\10}
if success goto slnodial
reinput 1 {\255}
if success goto slhup
reinput 1 {\127}
if success goto slhup
if < \%x 60 goto look
else goto slhup

:sllogin                        ; login
```

```
assign \%x 0                    ; zero the time counter
pause 1
echo Looking for login prompt.

:slloop
increment \%x                   ; Count the seconds
clear                           ; Clear unread characters from input buffer
output \13
;
; put your expected login prompt here:
;
input 1 {Username: }
if success goto sluid
reinput 1 {\255}
if success goto slhup
reinput 1 {\127}
if success goto slhup
if < \%x 10 goto slloop         ; try 10 times to get a login prompt
else goto slhup                 ; hang up and start again if 10 failures

:sluid
;
; put your userid here:
;
output ppp-login\13
input 1 {Password: }
;
; put your password here:
;
output ppp-password\13
input 1 {Entering SLIP mode.}
echo
quit

:slnodial
echo \7No dialtone.  Check the telephone line!\7
exit 1

; local variables:
; mode: csh
; comment-start: "; "
; comment-start-skip: "; "
; end:
```

18.4. PPP オーバイーサネット (PPPoE) の利用

以下の解説は, PPPoE として知られる, PPP オーバイーサネットの設定法です.

18.4.1. 必要なもの

あなたのシステムで PPPoE を適切に機能させるためには, 以下のものが必要です.

- FreeBSD 3.4やそれより新しいバージョンのカーネルソース
- FreeBSD 3.4やそれより新しいバージョンのppp

18.4.2. カーネルコンフィギュレーション

以下に示すオプションをカーネルコンフィギュレーションファイルに 追加して, その後 新しいカーネルをコンパイルする必要があります.

- options NETGRAPH

以下は任意

- options NETGRAPH_PPPOE
- options NETGRAPH_SOCKET

この機能は実行時には有効ではありませんが, 要求に応じて ppp は関係のあるモジュールを 読み込みます.

18.4.3. ppp.conf の設定

これは動作している ppp.conf の 例です:

```
default: # or name_of_service_provider
  set device PPPoE:xl1 # replace xl1 with your ethernet device
  set mru 1492
  set mtu 1492
  set authname YOURLOGINNAME
  set authkey YOURPASSWORD
  set log Phase tun command # you can add more detailed logging if you wish
  set dial
  set login
  set ifaddr 10.0.0.1/0 10.0.0.2/0
  add default HISADDR
  nat enable yes # if you want to enable nat for your local net

papchap:
  set authname YOURLOGINNAME
  set authkey YOURPASSWORD
```

-nat オプションを付けてPPPoEを起動する際には注意するべきです.

18.4.4. PPP の起動

以下を root 権限において実行することで, 起動させることができます:

```
# ppp -ddial name_of_service_provider
```

18.4.5. システム起動時に PPP を立ち上げる

/etc/rc.conf ファイルに以下の行を追加 してください:

```
ppp_enable="YES"
ppp_mode="ddial"
ppp_nat="YES"
ppp_profile="default" # or your provider
```

18.5. SLIP の利用

18.5.1. SLIPクライアントのセットアップ

ここには FreeBSD マシンを静的アドレスのネットワークにつなげる場合のSLIPのセットアップの一つの方法を書いてあります. ホスト名を動的に割り当てる(つまり, ダイヤルアップするたびにアドレスが かわる)ためには, おそらくもっと凝ったことが必要です.

まず, モデムがどのシリアルポートにつながっているか決めましょう. 私は /dev/cuaa1 から /dev/modemへというシンボリックリンクを張り,
コンフィグレーションではその名前だけを使っています. /etc や.kermrc など, システム全体に散らばっているファイルを修正する 必要がでるとまったく煩わしいのです!

ここで, /dev/cuaa0は COM1であり, cuaa1はCOM2です.

カーネルのコンフィグレーションファイルに

```
pseudo-device   sl      1
```

という記述があるのを確認してください. これは GENERIC カーネルに含まれているので削除していない限り大丈夫でしょう.

最初の設定

1. /etc/hosts ファイルにあなたのマシンのゲートウェイとネームサーバ を加えてください. 私のは以下のようになっています.

```
127.0.0.1               localhost loghost
136.152.64.181          silvia.HIP.Berkeley.EDU silvia.HIP silvia
136.152.64.1            inr-3.Berkeley.EDU inr-3 slip-gateway
128.32.136.9            ns1.Berkeley.edu ns1
128.32.136.12           ns2.Berkeley.edu ns2
```

2. /etc/host.conf ファイル中で hostsがbind よりも前にあること を確認してください. さもないとヘンなことが起こるかもしれません.
3. /etc/rc.conf ファイルを編集してください. なお, お使いの FreeBSD が 2.2.2 よりも前のバージョンのものの場合は, /etc/sysconfig を編集してください.
   a. 行

      ```
      hostname=myname.my.domain
      ```

      を編集してホスト名をセットしてください. 完全なInternetホスト名を与えるべきです.

   b. 行

      ```
      network_interfaces="lo0"
      ```

      を

      ```
      network_interfaces="lo0 sl0"
      ```

      へ変更することにより ネットワークインタフェースのリストに sl0 を加えてください.

   c. 行

      ```
      ifconfig_sl0="inet ${hostname} slip-gateway netmask 0xffffff00 up"
      ```

      を加えて sl0 のスタートアップフラグをセットしてください.

   d. 行

      ```
      defaultrouter=NO
      ```

      を

      ```
      defaultrouter=slip-gateway
      ```

      へ変更してデフォルトのルータを 指定してください.

4. 次の

   ```
   domain HIP.Berkeley.EDU
   nameserver 128.32.136.9
   nameserver 128.32.136.12
   ```

という内容を含むファイル /etc/resolv.conf を作ってください. 見ればわかるように, これらはネームサーバホストを設定しています. もちろん, 実際のドメイン名やアドレスはあなたの環境に依存します.

5. root と toor (及びパスワードを持っていない他のアカウントすべて) のパスワードを設定してください. passwdコマンドを使いましょう. /etc/passwd や /etc/master.passwd といったファイルを編集してはいけません!
6. マシンを再起動して正しいホスト名で 立ち上がることを確認してください.

SLIP接続をおこなう

1. モデムを起動, つながったらプロンプトで slipとタイプし, マシン名とパスワードを入力してください. 入力する必要があるものは環境に よって異なります. 私は次のようなスクリプトでkermitを使っています.

```
# kermit setup
set modem hayes
set line /dev/modem
set speed 115200
set parity none
set flow rts/cts
set terminal bytesize 8
set file type binary
# The next macro will dial up and login
define slip dial 643-9600, input 10 =>, if failure stop, -
output slip\x0d, input 10 Username:, if failure stop, -
output silvia\x0d, input 10 Password:, if failure stop, -
output ***\x0d, echo \x0aCONNECTED\x0a
```

(もちろん, ホスト名とパスワードは変える必要があります). 接続するためには kermit のプロンプトで slipとタイプするだけです.

ファイルシステムのどんなところにもプレインテキスト にパスワードを書いておくのは一般的にはよくありません. 覚悟の上で やってください. 私は単に不精なだけです.

2. ここでkermitから抜け出し (zでkermitをサスペンドできます), root で

```
# slattach -h -c -s 115200 /dev/modem
```

と入力しましょう. もしルータの向う側のホストへ ping できるなら接続成功です! もしうまくいかなければslattachへの引数として -c の代わりに-aとやってみてください.

接続の切り方

slattachを殺すためにrootで

```
# kill -INT `cat /var/run/slattach.modem.pid`
```

とタイプしてください. そして kermit に戻り (もしkermitをサスペンドしていたなら fg), kermitから抜けてください (q).

slattachのマニュアルページにはインタフェースを落すために ifconfig sl0 downをしなければいけないと書いていますが, 私には差がないように見えます. (ifconfig sl0とやっても同じ結果が得られる.)

時にはモデムがキャリアを落すのを 拒絶するかもしれません(私のは よくそうなります). その時は単にkermitをスタートしてまた終了 してください. 普通は2回目で落ちます.

トラブルシューティング

もし動かなければ自由に私に質問してください. 今までいろんな人がつまずいた のは次のようなことです.

- slattach で -c や -a を使わなかった(私はなぜこれが致命的になり得るのか わかりませんが, このフラグを付けることで少なくとも一人の 問題は解決しました.)
- sl0 の代わりに s10 を使った(いくつかのフォントでは見分けるのは難しい かもしれません).
- インタフェースの状態を見るために ifconfig sl0 をやってみてください. 私は,

  ```
  # ifconfig sl0
  sl0: flags=10<POINTOPOINT>
          inet 136.152.64.181 --> 136.152.64.1 netmask ffffff00
  ```

  となります.

- また, pingが "no route to host" というメッセージを返す時には netstat -rでルーティングテーブルを確認しましょう. 私のは,

  ```
  # netstat -r
  Routing tables
  Destination      Gateway            Flags     Refs     Use  IfaceMTU    Rtt
  Netmasks:
  (root node)
  (root node)

  Route Tree for Protocol Family inet:
  (root node) =>
  default          inr-3.Berkeley.EDU UG          8   224515  sl0 -      -
  localhost.Berkel localhost.Berkeley UH          5    42127  lo0 -       0.438
  inr-3.Berkeley.E silvia.HIP.Berkele UH          1        0  sl0 -      -
  silvia.HIP.Berke localhost.Berkeley UGH        34 47641234  lo0 -       0.438
  ```

```
(root node)
```

となります. (これはたくさんのファイルを転送した後でのもので, あなたの見る数字はもっと小さいかも しれません).

18.5.2. SLIPサーバのセットアップ方法

この文書の目的は, SLIPサーバ機能を FreeBSDシステムのもとで設定するため の助言を提供することです. SLIPサーバ機能を設定するということは, リモー トの SLIPクライアントがログインできるようにするために, 自動的に接続処 理をおこなうようにすることです. この文書は著者の経験に基づいておりますが, 実際のシステム構成や要望は異なりますから, すべての疑問にこの文書が答え ることはできません. なお, ここでの助言を試みた結果, あなたのシステムへ の悪影響やデータの損失が生じたとしても, 著者が責任を持つことはできません のでご了解をお願いします.

前提

この文書の内容はテクニカルなものなので, 前提知識が必要です. すなわち, TCP/IPネットワークプロトコルについての知識, 特に, ネットワークとノード のアドレス指定をはじめ, ネットワークアドレスマスク, サブネット化, ルー ティング, および RIPなどのルーティングプロトコルなどに関する知識を前提 としています. ダイヤルアップサーバで SLIP機能を設定するためには, これ らの概念についての知識が必要ですから, もし不案内であると思われる方は, O'Reilly & Associates, Inc.から出版されている Craig Hunt氏の *TCP/IP Network Administration* (ISBN 0-937175-82-X)か, または Douglas Comer氏の TCP/IPプロトコルに関する一連の書籍をお読みください.

前提知識に加え, さらに, モデムの設定が完了しており, そのモデムを経由し てログインできるように, システムファイル群が適切に記述できているものと 仮定しています. もしモデムの準備ができていないときには, あらかじめダイヤ ルアップ機能の設定についてのチュートリアルをお読みください. Webブラ ウザが使えるのであれば http://www.FreeBSD.org/ におけるチュー トリアルの一覧を調べてください. あるいは, この文書を見つけた場所を調べ て, dialup.txt やそれに類似した名前の文書をお読みください. 関連するマニュアルページとしては, シリアルポート向けデバイスドライバについて の sio(4) をはじめ, モデムからのログインを 受理できるようにシステ ムを設定するための ttys(5), gettytab(5), getty(8), init(8) など, さらには, シリアルポート関連パラメータ (たと えば直接接続シリアルインタフェースの `clocal`) についての stty(1) なども助けになるかもしれません.

概要

一般的な設定内容で FreeBSDを SLIPサーバとして利用すると, その動作は次 のようになります. まず, SLIPユーザが FreeBSD による SLIPサーバへ電話し て, SLIP専用IDでログインします. なお, この IDを持ったユーザはシェルとし て /usr/sbin/sliplogin を使います. この `sliplogin` は, ファイル /etc/sliphome/slip.hosts の中から, ログインIDと一致する 記述行を探します. もし一致する行があれば, ログインしたシリアル回線を, 利用可能な SLIPインタフェースへ接続し, その後にシェルスクリプト /etc/sliphome/slip.login で SLIPインタフェースを設定します.

SLIPサーバへのログイン例

仮に SLIPユーザIDが `Shelmerg` とします. すると, /etc/master.passwd における `Shelmerg`

のエントリは次のよ うなものになります (実際には一つの行に続いている) .

```
Shelmerg:password:1964:89::0:0:Guy Helmer -
SLIP:/usr/users/Shelmerg:/usr/sbin/sliplogin
```

Shelmerg がログインすると, sliplogin は, ファイル /etc/sliphome/slip.hosts からユーザ IDと一致する行を探しま す. いま仮に, /etc/sliphome/slip.hosts に次のような記述がなされていたとします.

```
Shelmerg        dc-slip sl-helmer       0xfffffc00      autocomp
```

sliplogin が上記のエントリを見つけると, *Shelmerg* が使用して いるシリアル回線を, 利用可能な SLIPインタフェースのなかの最初のものへ 接続し, 次の内容の /etc/sliphome/slip.login を実行します.

```
/etc/sliphome/slip.login 0 19200 Shelmerg dc-slip sl-helmer 0xfffffc00 autocomp
```

もし上記の手順が正常に処理されると, /etc/sliphome/slip.login は, sliplogin が割り当てた SLIPインタフェース (この例では slip.login で与えられたパラメータのうちで最初の値である SLIP インタフェース0である) に対して ifconfig を実行し, ローカル IPアドレス (dc-slip)をはじめ, リモート IPアドレス (sl-helmer), SLIPインタフェースへのネットワークマスク (0xfffffc00), およびその他のフラグ (autocomp)を設定 します. 逆に, さきほどの手順が正常に終了しなかった場合, 通常は sliplogin は十分な情報を syslog の daemon 機能経由で /var/log/messages へ記録します (syslogd(8) や syslog.conf(5) のマニュアルページを参照のうえ, さらに /etc/syslog.conf を調べて syslogd がどのファイルへ記 録するかを確認のこと) .

例はこのくらいにして, さっそくシステムのセットアップを始めてみましょう.

カーネルのコンフィグレーション

FreeBSD のデフォルトのカーネルには, 通常, 二つの SLIPインタフェースが 準備されています (sl0 と sl1) . これらのインタフェー スが使用中のカーネルに準備されているかどうかを調べるには, netstat -i を実行してください.

netstat -i の出力例

```
Name  Mtu   Network     Address              Ipkts Ierrs    Opkts Oerrs  Coll
ed0   1500  <Link>0.0.c0.2c.5f.4a        291311     0   174209     0   133
ed0   1500  138.247.224 ivory            291311     0   174209     0   133
lo0   65535 <Link>                          79     0       79     0     0
lo0   65535 loop        localhost           79     0       79     0     0
sl0*  296   <Link>                         0     0        0     0     0
sl1*  296   <Link>                         0     0        0     0     0
```

netstat -i の出力に sl0 と sl1 のインタフェー スが含まれているということから, カーネルには二つの SLIPインタフェー スが組み込まれているということを示しています. (sl0 と sl1 に付いたアスタリスクは, netstat -i の実行時点で はインタフェースが "ダウン" していることを表しています.)

なお，パケットのフォワード機能は FreeBSD のデフォルトのカーネルでは設定 されていません (すなわちルータとしては動作しない)．もしインターネット 接続ホストについての RFC要件 (RFC 1009 [Requirements for Internet Gateways] と 1122 [Requirements for Internet Hosts - Communication Layers], おそらく 1127 [A Perspective on the Host Requirements RFCs] も) に準拠して，FreeBSDによる SLIPサー バをルータとして動作させたいときには，/etc/rc.conf (バージョ ン 2.2.2 より前の FreeBSD では /etc/sysconfig) ファイル の `gateway_enable` 変数を `YES` としてください．もし古いシステ ムで /etc/sysconfig ファイルすらないときには，次のコマン ドを /etc/rc.local へ追加してください．

```
sysctl -w net.inet.ip.forwarding = 1
```

この新しい設定を有効とするには，リブートする必要があります．

デフォルトのカーネルコンフィグレーションファイル (/sys/i386/conf/GENERIC) の最後の部分に，次のような行があります．

```
pseudo-device sl 2
```

この行によって，使用可能な SLIPデバイスの総数が決まります．すなわち，行 末の数値が，同時に動作可能な SLIP接続の最大数となります．

カーネルの再構築については，FreeBSDカー ネルのコンフィグレーション を参照ください．

Sliploginのコンフィグレーション

すでにご説明したように，/usr/sbin/sliplogin のコンフィグレー ションのために，3 種類のファイルが/etc/sliphome ディレクトリに あります (`sliplogin` についての実際のマニュアルページとしては sliplogin(8) を参照のこと) ． ファイル slip.hosts は SLIPユーザおよびその IPアドレスを決めます． 通常，ファイル slip.login は，SLIPインタフェースを設定することだけに使 用します．slip.logout はオプションのファイルで，slip.login で設定した内容を，シリアル接続が終了した時点で解除 するときに使用します．

slip.hosts のコンフィグレーション

/etc/sliphome/slip.hosts には，少なくとも 4 つの項目をホワイ トスペース (スペースやタブ) で区切って指定します．

- SLIPユーザのログインID
- SLIPリンクのローカル (SLIPサーバ側) アドレス
- SLIPリンクのリモートアドレス
- ネットワークマスク

ホスト名をローカルおよびリモートのアドレスとして 記述できます (IPアドレ スの決定は，/etc/host.conf の指定内容に応じて，/etc/hosts か DNSのいずれかによって決定される) ．また，ネット ワークマスクも /etc/networks ファイルに記述された名前を参照す ることで， 指定することもできると思います．これまでの例としてあげたシス テムでの /etc/sliphome/slip.hosts は次のようになります．

```
#
# login local-addr      remote-addr     mask            opt1    opt2
#                                                (normal,compress,noicmp)
#
Shelmerg  dc-slip       sl-helmerg      0xfffffc00      autocomp
```

それぞれの行の最後には, 次に示すオプションを一つ以上指定できます.

- normal - ヘッダを圧縮しない
- compress - ヘッダを圧縮する
- autocomp - リモートの設定に応じて, ヘッダを圧縮する
- noicmp - ICMPパケットを禁止する ("ping" パケットは送出されず, バンド幅を占有しない)

なお, FreeBSDバージョン2の初期リリースの sliplogin は, 旧 FreeBSD 1.xでは有効であった上記のオプションを無視していましたので, normal, compress, autocomp, そして noicmp などのオプションは FreeBSD 2.2でサポートされるまでは効果がありませんでした (ただしこれらのフラグを使うためには slip.login スクリプトへ記述する 必要がある) .

SLIPリンクでのローカルとリモート向けのアドレスの 選び方は, TCP/IPサブネッ トを専用に割り当てるか, または "プロキシ ARP" を SLIPサーバへ用いるかによって違います ("プロキシ ARP" という用語のここでの使い方は本来のものではないが, 説明のためにこの用語を使う) . もし, どちらの方式を選ぶべきか判らなかったり, IPアドレスの割り当て方が不明のときには, 上述の 前提 の節で紹介した TCP/IP関連書籍を参考になさるか, またはあなたの IPネットワークを管理している方に相談なさると よいでしょう.

独立したサブネットを SLIPクライアントへ適用するときには, すでに割り当てられている IPネットワーク番号の範囲からサブネット番号を割り当て, 同 時にそのサブネットの範囲内で有効な IPアドレスを SLIPクライアントの IP 番号として割り当てる必要があります. さらに, この SLIPサブネットから SLIPサーバを経由して最も近い IPルータへの経路を静的に設定するか, または gated を FreeBSDによる SLIPサーバへインストールして, 適当 なルーティングプロトコルを使って, SLIPサーバ経由のサブネットへの経路情 報をルータ群へ通知できるように設定するか, のいずれかをおこなう必要があります.

"プロキシ ARP" 方式を採用するときには, SLIPクライアント向けの IPアドレス として, SLIPサーバのサブネットの範囲から 選んで割り当てるとともに, arp(8) コマンドを使うために /etc/sliphome/slip.login と/etc/sliphome/slip.logout のスクリプトを修正して, SLIPサー バにおける ARPテーブル内のプロキシ ARPエントリへ 反映させる必要がありま す.

slip.login のコンフィグレーション

ファイル /etc/sliphome/slip.login の一般的な内容は次にようになります.

```
#!/bin/sh -
#
#   @(#)slip.login  5.1 (Berkeley) 7/1/90

#
```

```
# generic login file for a slip line.  sliplogin invokes this with
# the parameters:
#      1        2         3        4          5         6     7-n
#   slipunit ttyspeed loginname local-addr remote-addr mask opt-args
#
/sbin/ifconfig sl$1 inet $4 $5 netmask $6
```

この slip.login ファイルの役目は単に，SLIPインタフェースにつ いてのローカルとリモートのアドレス，およびそのネットワークマスクを ifconfig コマンドで設定することです.

もし "プロキシ ARP" 方式を採用する (SLIPクライアントへ独立したサブネットを使わない) ときには，ファイル /etc/sliphome/slip.login は次のような内容になります.

```
#!/bin/sh -
#
#   @(#)slip.login  5.1 (Berkeley) 7/1/90

#
# generic login file for a slip line.  sliplogin invokes this with
# the parameters:
#      1        2         3        4          5         6     7-n
#   slipunit ttyspeed loginname local-addr remote-addr mask opt-args
#
/sbin/ifconfig sl$1 inet $4 $5 netmask $6
# Answer ARP requests for the SLIP client with our Ethernet addr
/usr/sbin/arp -s $5 00:11:22:33:44:55 pub
```

この slip.login で追加された行 arp -s $5 00:11:22:33:44:55 pub は，SLIPサーバにおけるARPテーブルへ新たなエントリを作ります. SLIPサーバ は，この ARPエントリが作られると，SLIPクライアントの IPアドレスと話し たい他の IPノードが要求してきたときにはいつも，SLIPサーバ の Ethernet MACアドレスを返すようになります.

上記の例を実際に流用なさるときには，例にある Ethernet MACアドレス (00:11:22:33:44:55) を，あなたのシステムの実際のEthernetカー ドの MACアドレスと置き換えなければ "プロキシ ARP" はうまく動作しません! SLIPサーバの Ethernet MACアドレスを調べるには netstat -i コマンドを利用してください. 実行結果の第2行は次のようなものになるはずです.

```
ed0   1500  <Link>0.2.c1.28.5f.4a         191923     0   129457     0   116
```

この例での Ethernet MACアドレスは 00:02:c1:28:5f:4a であると 読みます. なお arp(8) における MACアドレスの指定に際しては，コマンド netstat -i が付けた Ethernet MACアドレスのピリオド記号をコロン記号と置き換え，かつ単一桁の 16 進数にはゼロを先頭に加える必 要があります. この指定についての正確な情報は arp(8) を参照く ださい.

/etc/sliphome/slip.login と /etc/sliphome/slip.logout を作成したならば，ファイル属性の "実行" ビット (すなわち chmod 755 /etc/sliphome/slip.login /etc/sliphome/slip.logout) を 設定しなければなりません. さもなければ sliplogin が

> うまく実行されません.

slip.logout のコンフィグレーション

ファイル /etc/sliphome/slip.logout は必ずしも必要なものではあ りません (ただし "プロキシ ARP" を利用する場合を除く) . もしこのファイルを 作成するときには, 次に示す標準的な slip.logout スクリプト例を 参考にしてください.

```
#!/bin/sh -
#
#   slip.logout

#
# logout file for a slip line.  sliplogin invokes this with
# the parameters:
#      1        2         3        4          5         6     7-n
#   slipunit ttyspeed loginname local-addr remote-addr mask opt-args
#
/sbin/ifconfig sl$1 down
```

"プロキシ ARP" を利用する場合, この /etc/sliphome/slip.logout を 使って, 特定の SLIPクライアント向けの ARPエントリを削除したくなるようなときがあります.

```
#!/bin/sh -
#
#       @(#)slip.logout

#
# logout file for a slip line.  sliplogin invokes this with
# the parameters:
#      1        2         3        4          5         6     7-n
#   slipunit ttyspeed loginname local-addr remote-addr mask opt-args
#
/sbin/ifconfig sl$1 down
# Quit answering ARP requests for the SLIP client
/usr/sbin/arp -d $5
```

コマンド `arp -d $5` は, SLIPクライアントがログインした 際に, "プロキシ ARP" を使った slip.login によって追加され た ARPエントリを削除します.

これによって, 繰り返して利用することができるわけです. 必ず, /etc/sliphome/slip.logout を作成した後に, 実行ビットを設定し てください (`chmod 755 /etc/sliphome/slip.logout`) .

ルーティングについての考慮点

"プロキシ ARP" 方式を利用せずに SLIPクライアントとその他のネットワーク (Internetも含む) の構成要素との間でパケットをルーティングするときには, SLIPサーバ経由で SLIPクライアントが属するサブネットまでの経路を, 最も

近いデフォルトのルータ群へ静的な経路情報として 追加しなければならないか, または gated を FreeBSDによる SLIPサーバへインストールして, SLIP サブネットについての経路情報を, 適当なルーティングプロトコルでルー タ群へ通知できるように設定するか, のどちらかをおこなわなければなりません.

静的な経路

静的な経路を最も近いデフォルトの ルータ群へ追加することが困難なことがあ ります (経路情報を追加できる権限がなければそもそも不可能となる). もし あなたの組織に複数のルータで構成された ネットワークがあるならば, ある種 のルータ (たとえば CiscoやProteonなど) は, 静的な経路を SLIPサブネッ トへ使うようにルータを設定しなければならないだけでなく, その静的経路を 他のどのルータへ知らせるのかもあらかじめ 指定しておく必要がありますから, 静的経路に基づくルーティングを軌道に乗せるには それなりの専門的技術やト ラブルシューティングやコツが必要だと思います.

gatedの稼働

静的経路についての頭痛への代替手段は, gated を FreeBSDによる SLIPサー バへインストールして, 適切なルーティングプロトコル (RIP/OSPF/BGP/EGP) を使って SLIPサブネットについての経路情報を他のルータへ知らせるように 設定することです. ports コレクションから gated を用いることもできますし, GateD 匿名 FTP サイト から探して自分自身で構築することもで きます. この文章を執筆時点の最新バージョンは gated-R3_5Alpha_8.tar.Z であり, このファイル "だけで" FreeBSDで 動作させることができます. gated についてのすべての情報と文書 は Merit GateD コンソーシアム からはじまる Web 上で入手でき ます. gated のコンパイルとインストールを行ったならば, 独自の 設定のために /etc/gated.conf ファイルを記述してください. 次の 例は, 筆者が FreeBSDによる SLIP サーバで使っている内容と類似のものです.

```
#
# gated configuration file for dc.dsu.edu; for gated version 3.5alpha5
# Only broadcast RIP information for xxx.xxx.yy out the ed Ethernet interface
#
#
# tracing options
#
traceoptions "/var/tmp/gated.output" replace size 100k files 2 general ;

rip yes {
 interface sl noripout noripin ;
 interface ed ripin ripout version 1 ;
 traceoptions route ;
} ;

#
# Turn on a bunch of tracing info for the interface to the kernel:
kernel {
 traceoptions remnants request routes info interface ;
} ;
```

```
#
# Propagate the route to xxx.xxx.yy out the Ethernet interface via RIP
#

export proto rip interface ed {
        proto direct {
                xxx.xxx.yy mask 255.255.252.0 metric 1; # SLIP connections
        } ;
} ;

#
# Accept routes from RIP via ed Ethernet interfaces

import proto rip interface ed {
        all ;
} ;
```

この gated.conf ファイルの例では，SLIPのサブネット *xxx.xxx.yy* についての経路情報を RIPを使ってEthernetへブロー ドキャストしています．もし ed ドライバ以外の Ethernetドライバを使うのであれば，ed インタフェースの記述を適切なものに置き換えてくだ さい． またこの例では，gatedの動作をデバッグするために， /var/tmp/gated.output へトレース情報を出力するように指示しています．gated が希望通りに動作したならば，このトレースオプショ ンを止めることができます．なお，例における *xxx.xxx.yy* を，あ なた自身の SLIPサブネットのネットワークアドレスに換えてください (また proto direct 部分のネットワークマスクも換えることを忘れないこ と) .

gated のコンパイルとインストールが終了し， コンフィグレーショ ンファイルの作成も完了したら，FreeBSDシステムではデフォルトの routedに代わって gated を起動してください． そのため には，/etc/netstart の routed/gated 起動パラメータを 適切な値に設定してください． gated のコマンドラインパラメータにつ いての情報は，gated のマニュアルページを参照してください．

Chapter 19. 電子メール

19.1. この章では

"電子メール"、email としてのほうが知られているでしょう、は現代で最も広く利用されているコミュニケーション手段の一つです。 この章では FreeBSD 上でメールサーバを実行するための基本的な導入を説明します。
しかし、この文書は完璧な参考文献ではなく、
実際のところ考慮すべき重要な点の多くが省略されています。
この件について、より網羅したものについては 参考図書
に掲載されている多くの優れた書籍を参照してください。

この章では、以下の分野について説明します。

- 電子メールの送受信に関係しているソフトウェアの構成要素
- FreeBSD における sendmail の基本的な設定ファイルのある場所
- スパマーがあなたのメールサーバを踏台として不正に使用することを防ぐ方法
- あなたのシステムに sendmail の置き換えとなる代替の MTA をインストールして設定する方法
- メールサーバにまつわる共通の問題の解決法
- UUCP とともに SMTP を使う方法
- ダイアルアップ接続でメールを使う方法
- セキュリティを向上するために SMTP 認証を設定する方法

この章を読む前に、以下のことを理解しておく必要があります。

- ネットワーク接続の適切な設定方法 (高度なネットワーク)
- あなたのメールホストに対する DNS 情報の適切な設定方法 (高度なネットワーク)
- サードパーティ製ソフトウェアのインストール方法 (アプリケーションのインストール - packages と ports)

19.2. 電子メールを使う

email の交換には 5 つの主要な部分があります。 それらは ユーザープログラム、 サーバーデーモン、 DNS、 POP もしくは IMAP のデーモン、 そしてもちろん メールホストです。

19.2.1. ユーザープログラム

いくつか名前を挙げれば、 mutt, pine, elm そして mail といったコマンドラインプログラムや balsa, xfmail のような GUI プログラム、WWW ブラウザーのようにさらに "洗練された" ものまであります。 これらのプログラムは、email の処理を server daemons を呼び出したり TCP 経由で渡したり、といった手段でローカルの "メールホスト" に任せるだけです。

19.2.2. メールホストサーバデーモン

通常、これは sendmail (FreeBSD のデフォルト) や qmail, postfix もしくは exim

といった他のメールサーバーデーモンの一つです。
他にもあるのですが、以上のものが広く使われています。

サーバーデーモンは通常 2 つの機能 - やってくるメールを受け取るのと出ていくメールを配送する、を持っています。メールを読むために POP や IMAP で接続する、ということはできません。そのためにはもう一つデーモンが必要なのです。

いくつかの古いバージョンの sendmail には深刻なセキュリティ問題がありますが、現在のバージョンを使っていれば特に問題ないことに注意してください。例のごとく、どんなソフトウェアを利用する時にも最新の状態にしておくのが大事なのです。

19.2.3. Email と DNS

Domain Name System (DNS) とそのデーモンである `named` は email の配送において大変重要な役割を担ってます。
あなたのサイトからもう一つのサイトへメールを配送するためには、サーバーデーモンは DNS からそのサイトを探し、メールの受け取り先のホストを決定します。

メールがあなたに送られた場合にも同じような仕組みになっています。DNS にはホスト名と IP アドレス、ホスト名とメールホストをマッピングするデータベースがあります。IP アドレスは A レコードで指定されます。MX (Mail eXchanger) レコードはあなた宛のメールを受け取るホストを指定します。あなたのホスト名に対する MX レコードがない場合には、メールは直接あなたのホストに配送されます。

19.2.4. メールの受け取り

メールはメールホストが受け取ります。このホストは送られてきたメールを集め、(ユーザーが) 読んだりピックアップしたりするために保存します。
保存されているメールをピックアップするにはメールホストに接続する必要があります。これは POP や IMAP を用いて行なわれます。メールホスト上で直接メールを読みたい時は POP や IMAP のサーバーは必要ありません。

POP や IMAP のサーバーを走らせるためには 2 つのことをやらなければいけません。

1. POP や IMAP のデーモンを ports コレクション からインストールします。
2. /etc/inetd.conf を修正して POP や IMAP のサーバーが起動されるように設定します。

19.2.5. メールホスト

メールホストとは責任をもってメールを配送したり、
あなたのホストや、もしかするネットワークも、に宛てたメールを受け取ったりするホストに与えられる名前です。

19.3. sendmail の設定

sendmail(8) は FreeBSD のデフォルトの メールトランスファエージェント (MTA) です。sendmail の仕事はメールユーザエージェント (MUA) からのメールを受け取り、それを設定ファイルで定義された適当なメーラに届けることです。sendmail はネットワーク接続を受け入れて、ローカルのメールボックスにメールを届けたり

別のプログラムにメールを渡したりもできます。

sendmail は次の設定ファイルを使用します。

ファイル名	機能
/etc/mail/access	sendmail アクセスデータベースファイル
/etc/mail/aliases	メールボックスエイリアス
/etc/mail/local-host-names	sendmail が受け付ける配送先ホストのリスト
/etc/mail/mailer.conf	メーラプログラムの設定
/etc/mail/mailertable	メーラ配送表
/etc/mail/sendmail.cf	sendmail の主設定ファイル
/etc/mail/virtusertable	仮想ユーザおよび仮想ドメイン表

19.3.1. /etc/mail/access

アクセスデータベースは、どのホストまたは IP アドレスがローカルメールサーバに接続できるか、そして接続の種類は何か、ということを定義します。ホストは OK, REJECT, RELAY として指定できます。または、メーラエラーを指定することで、単に sendmail の エラー処理ルーチンに渡されます。OK として指定されたホスト（これはデフォルトです）は、メールの最終宛先がローカルマシンである限り、このホストへメールを送ることを認められます。REJECT として指定されたホストは、すべてのメール接続を拒絶されます。ホスト名に対して RELAY オプションを指定されたホストは、このメールサーバを通過して任意の宛先へメールを送ることを認められます。

例 *41. sendmail* アクセスデータベースの設定

```
cyberspammer.com                550 We don't accept mail from spammers
FREE.STEALTH.MAILER@            550 We don't accept mail from spammers
another.source.of.spam          REJECT
okay.cyberspammer.com           OK
128.32                          RELAY
```

この例では五つのエントリがあります。
表の左側に当てはまるメール送信者は、表の右側の動作に支配されます。
はじめの二つの例は、エラーコードを sendmail のエラー処理ルーチンに渡します。
メールが表の左側に当てはまると、リモートホストにそのメッセージが表示されます。次のエントリは another.source.of.spam というインターネット上の特定のホストからのメールを拒絶します。
次のエントリは okay.cyberspammer.com からのメール接続を受け入れます。このエントリは上にある cyberspammer.com という行よりもさらに厳密です(厳密に一致すればするほど、そうでないものより優先されます)。最後のエントリは 128.32 から始まる IP アドレスのホストからの電子メールのリレーを認めます。
これらのホストは他のメールサーバに到達できるこのメールサーバを使ってメールを送ることができるでしょう。

このファイルを変更したら、データベースを更新するために /etc/mail/ ディレクトリで make コマンドを実行する必要があります。

19.3.2. /etc/mail/aliases

エイリアスデータベースには、 他のユーザ、ファイル、プログラムまたは他のエイリアスに展開される仮想的なメールボックスの一覧が記載されています。 /etc/mail/aliases において使用できる例をいくつかあげます。

例 *42.* メールエイリアス

```
root: localuser
ftp-bugs: joe,eric,paul
bit.bucket:  /dev/null
procmail: "|/usr/local/bin/procmail"
```

ファイル形式はシンプルです。
コロンの左側にあるメールボックス名は、右側のターゲットに展開されます。 はじめの例は単純に `root` のメールボックスを `localuser` のメールボックスに展開し、それからエイリアスデータベースをもう一度調べます。
一致するエントリがなければメッセージはローカルユーザである `localuser` に配送されます。
次の例はメーリングリストです。 `ftp-bugs` のメールボックスへのメールは `joe`, `eric` および `paul` の三つのローカルメールボックスに展開されます。 リモートメールボックスは `user@example.com` のように指定できることに注意してください。 次の例はメールをファイル、この場合 /dev/null に書き込みます。 最後の例はメールをプログラムに送ります。 この場合メールのメッセージは UNIX® パイプを通じて /usr/local/bin/procmail の標準入力に書き込まれます。

このファイルを変更したら、 データベースを更新するために/etc/mail/ ディレクトリで `make` コマンドを実行する必要があります。

19.3.3. /etc/mail/local-host-names

これは sendmail(8) がローカルホスト名として認めるホスト名のリストです。 sendmail がメールを受け取るすべてのドメインやホストにこのファイルを置いてください。
たとえば、このメールサーバは `example.com` というドメインおよび `mail.example.com` というホストへのメールを受け取るとすると、 local-host-names ファイルの内容は次のようになるでしょう。

```
example.com
mail.example.com
```

このファイルを更新したら、変更を読み込むために sendmail(8) を再起動する必要があります。

19.3.4. /etc/mail/sendmail.cf

sendmail の主設定ファイルである sendmail.cf は、電子メールアドレスの書き換えから、リモートメールサーバへ拒絶メッセージを送ることまで sendmail の全般的な動作をすべて制御します。
当然、そのようなさまざまな役割によりこの設定ファイルは大変複雑で、
その詳細についてはこの節の少し範囲外です。好運なことに、
標準的な構成のメールサーバではこのファイルをめったに変更する必要はありません。

sendmail の主設定ファイルは sendmail の機能と動作を決定する m4(1) マクロから構築できます。詳細については /usr/src/contrib/sendmail/cf/README を参照してください。

このファイルを更新したら、その変更を反映するために sendmail を再起動する必要があります。

19.3.5. /etc/mail/virtusertable

virtusertable
は仮想ドメインおよび仮想メールボックスに対するアドレスを実際のメールボックスと対応づけます。
これらのメールボックスにはローカル、リモート、 /etc/mail/aliases に定義されたエイリアス、
またはファイルを使用できます。

例 *43.* 仮想ドメインメール対応表の例

```
root@example.com                root
postmaster@example.com          postmaster@noc.example.net
@example.com                    joe
```

上の例では example.com ドメインへの対応づけをしています。
このファイルはファイルの下までファーストマッチ (訳注: 一致するルールが複数ある場合、一番最初に一致したルールが適用されること) で処理されます。 はじめの行では root@example.com をローカルの root メールボックスに対応づけています。 次のエントリでは postmaster@example.com を noc.example.net ホスト上の postmaster メールボックスに対応づけています。
最後に、今までのところでは example.com
に関して何も一致しない場合、最後のエントリと一致するでしょう。 これは example.com
の誰かに送ったすべてのメールが一致します。これは joe
のローカルメールボックスに対応づけられています。

19.4. MTA の変更

すでに述べたように、FreeBSD には MTA (Mail Transfer Agent) として、 sendmail がすでにインストールされています。
したがって、デフォルトではこれがメールの送受信を担当しています。

しかしながら、さまざまな理由によって、 システムの MTA を変更しようと考えるシステム管理者もいるかもしれません。 その理由は、単に他の MTA を試してみたいというものから
他のメーラに依存する特定の機能やパッケージが必要だといったものまで、 多岐にわたることでしょう。
幸い、理由がどんなものであれ、FreeBSD では簡単に変更できます。

19.4.1. 新しい MTA のインストール

さまざまな MTA が利用できます。 FreeBSD Ports Collection から探しはじめるのがよいでしょう。
もちろん、どんな場所からでも、あなたが利用したい MTA が FreeBSD で動作する限りすべて自由に使えます。

新しい MTA をインストールすることからはじめましょう。 新しい MTA をインストールすると、あなたの要求が実際に実現したかどうか決める機会が与えられます。さらに、 サービスを sendmail

から引き継ぐ前に 新しいソフトウェアを設定する機会が与えられます。これを行う場合、新しいソフトウェアが /usr/bin/sendmail のようなシステムバイナリを上書きしようとしないことを確認してください。
そうしないとあなたが設定する前に新しいメールソフトウェアが本格的に動作しはじめてしまいます。

あなたが選択したソフトウェアを設定する方法についての情報は、その MTA の文書を参照してください。

19.4.2. sendmail を無効にする

sendmail を起動するために使用されていた手続きは、4.5-RELEASE と 4.6-RELEASE の間で著しく変更されました。 したがって、それを無効にするための手続きは微妙に違います。

2002 年 4 月 4 日より前の FreeBSD 4.5-STABLE (4.5-RELEASE とそれ以前のバージョンが該当)

/etc/rc.conf に次の行を加えてください。

```
sendmail_enable="NO"
```

これは sendmail のメール受信機能を無効にします。 しかし /etc/mail/mailer.conf (下記参照) が変更されていなければ、sendmail はメールの送信にまだ使われるでしょう。

2002 年 4 月 4 日以降の FreeBSD 4.5-STABLE (4.6-RELEASE とそれ以降のバージョンが該当)

sendmail を完全に無効にするためには /etc/rc.conf に次の行を加えなくてはいけません。

```
sendmail_enable="NONE"
```

もしこの方法で sendmail のメール送信機能を無効にしたのなら、完全に動作する代替メール配送システムと置き換えることが重要です。
さもなければ、periodic(8) などのシステム機能は、それらの結果を通常想定しているようにメールで配送することができなくなるでしょう。 システムの多くの部分が sendmail 互換のシステムがあることを想定しているかもしれません。
もしそれらを無効にした後に、アプリケーションがメールを送ろうとするために sendmail のバイナリを使用し続ければ、メールは使われていない sendmail のキューに入り、そして決して配送されないでしょう。

もし sendmail のメール受信機能だけを無効にしたいのなら /etc/rc.conf に以下の行を追加してください。

```
sendmail_enable="NO"
```

sendmailの起動オプションに関する詳細は rc.sendmail(8) マニュアルをご覧ください。

19.4.3. 起動時に新しい MTA を起動する

起動時に新しい MTA を起動するには二つの選択肢があります。 ここでも、あなたが稼働させている FreeBSD のバージョンに依存します

2002 年 4 月 11 日より前の FreeBSD 4.5-STABLE (4.5-RELEASE とそれ以前のバージョンが該当)

/usr/local/etc/rc.d/ ディレクトリに、 ファイル名が .sh でおわり、 `root` によって実行可能なスクリプトを追加します。 このスクリプトは `start` および `stop` パラメータを引数として受け付けるようにします。
起動時にシステムスクリプトは次のコマンドを実行するでしょう。

```
/usr/local/etc/rc.d/supermailer.sh start
```

これは手動でサーバを起動するためにも使用できます。 システム終了時にはシステムスクリプトは `stop` オプションを使用して、次のコマンドを実行するでしょう。

```
/usr/local/etc/rc.d/supermailer.sh stop
```

これはシステムが稼働している間に手動でサーバを停止するためにも使えます。

2002 年 4 月 11 日以降の FreeBSD 4.5-STABLE (4.6-RELEASE とそれ以降のバージョンが該当)

より新しいバージョンの FreeBSD では、 上記の方法または次の行を /etc/rc.conf に設定できます。

```
mta_start_script="filename"
```

filename は、あなたが MTA を立ち上げるために起動時に実行するスクリプト名です。

19.4.4. システムのデフォルトメーラとして sendmail を置き換える

sendmail プログラムは UNIX® システム上の標準ソフトウェアとして本当にどこでも利用できるので、これがすでにインストールおよび設定されているとみなしている ソフトウェアもあるかもしれません。 この理由により、代替となる MTA の多くは sendmail コマンドラインインタフェースと互換性のある実装を提供しています。 これを "差し込む" ことによって、 sendmail の置き換えとして代替 MTA を使用することが容易になります。

したがって、あなたが互換メーラを使用しているときには、 /usr/bin/sendmail のような標準 sendmail バイナリを実行しようとするソフトウェアが、
実際にはその代わりにあなたの選択したメーラを実行しているということを
確かめる必要があるでしょう。 好運なことに、FreeBSD はこの仕事をする mailwrapper(8) と呼ばれるシステムを提供しています。

インストールされたまま sendmail が稼働しているときには /etc/mail/mailer.conf には以下のような記述があるでしょう。

```
sendmail    /usr/libexec/sendmail/sendmail
send-mail   /usr/libexec/sendmail/sendmail
```

```
mailq       /usr/libexec/sendmail/sendmail
newaliases  /usr/libexec/sendmail/sendmail
hoststat    /usr/libexec/sendmail/sendmail
purgestat   /usr/libexec/sendmail/sendmail
```

このことは、これらのうちどの共通コマンド (sendmail 自身のような) が実行されても、 システムは mailer.conf を確認して、 代わりに /usr/libexec/sendmail/sendmail を実行する sendmail という名前の mailwapper のコピーを呼び出すことを意味します。 このようなシステムでは、デフォルトの sendmail が呼び出されたときに、 どのバイナリが実際に実行されるかを変更するのが簡単になります。

したがって、sendmail の代わりに /usr/local/supermailer/bin/sendmail-compat を実行させたいのなら、次のように /etc/mail/mailer.conf を変更してください。

```
sendmail    /usr/local/supermailer/bin/sendmail-compat
send-mail   /usr/local/supermailer/bin/sendmail-compat
mailq       /usr/local/supermailer/bin/mailq-compat
newaliases  /usr/local/supermailer/bin/newaliases-compat
hoststat    /usr/local/supermailer/bin/hoststat-compat
purgestat   /usr/local/supermailer/bin/purgestat-compat
```

19.4.5. 完了

あなたのやりたいようにすべてを設定しおえたら、 もはや必要のない sendmail のプロセスを終了して新しいソフトウェアに関するプロセスを起動するか、 単に再起動してください。 再起動することによって、新しい MTA が起動時に正しく立ち上がるようにシステムが設定されているかどうか確認することもできるでしょう。

19.5. トラブルシュート

19.5.1. どうして自分のサイトのホストなのに FQDN を使わなければいけないのですか?

恐らく、そのホストは実際には別のドメインにあるのでしょう。 例えば `foo.bar.edu` ドメインにいて、`bar.edu` というドメイン内の `mumble` というホストにアクセスしたいとします。 この時は単に `mumble` ではなく `mumble.bar.edu` と FQDN で参照しなければなりません。

そもそも、BSD BIND のリゾルバー (resolver) ではこのようなことが可能でしたが、 FreeBSD に入っている最新版の BIND では自分のドメイン以外に対する FQDN でない省略形は許されません。 従ってホストを `mumble` と曖昧に指定した場合は `mumble.foo.bar.edu` という名前があればそれになり、そうでなければ root ドメインから検索されます。

これは、`mumble.bar.edu` と `mumble.edu` ということなったドメイン名に対してホスト名のサーチがおこなわれていた以前の振る舞いとは異なったものです。 このような事が悪い例もしくはセキュリティホールとみなされる理由については RFC 1535 を見てください。

/etc/resolv.conf で

```
domain foo.bar.edu
```

と書いてある行を

```
search foo.bar.edu bar.edu
```

と書き換えることで上のようなことができます。 しかし、RFC 1535 にあるように検索順序が "内部 (local) と外部 (public) の管理の境界" をまたがないようにしてください。

19.5.2. sendmail が mail loops back to myself というメッセージを出すのですが。

sendmail FAQ に次のように書いてあります。

```
Local configuration error というメッセージが出ます。例えば、

553 relay.domain.net config error: mail loops back to myself
554 <user@domain.net>... Local configuration error

のような感じですが、どうしたら解決できますか?

これは、例えば domain.net のようなドメイン宛てのメールを
MX レコードで特定のホスト(ここでは
relay.domain.net) に送ろうとしたのに、
そのホストでは domain.net
宛てのメールを受け取れるような設定になっていない場合です。
設定の際に
FEATURE(use_cw_file) を指定してある場合には
/etc/mail/local-host-names の中に domain.net を追加してください。
もしくは、/etc/mail/sendmail.cf の中に Cw domain.net
を追加してください。
```

sendmail FAQ は http://www.sendmail.org/faq にありますので、 メールの設定に "おかしなこと" があれば常に読んでください。

19.5.3. ダイアルアップ PPP ホストでメールサーバを実行するにはどうしたらいいの?

LAN 上にある FreeBSD マシンを、 インターネットに接続したいとします。FreeBSD マシンは、その LAN でのメールゲートウェイになります。FreeBSD マシンは専用線接続ではありません (訳注: ダイアルアップ接続など)。

これには、少なくとも二つの方法があります。 一つは UUCP を使うことです。

もう一つの方法は、あなたのドメインに対するセカンダリ MX サービスを提供する常時稼働のインターネットサーバを用意することです。
たとえば、あなたの会社のドメインが example.com で、 ISP があなたのドメインに セカンダリ MX サービスを提供するために example.net ドメインを 用意するとしたら次のようにします。

```
example.com.    MX  10  example.com.
MX  20  example.net.
```

最終的なメール受信先としては、一つのホストだけが定義されるべきです (example.com 上の /etc/mail/sendmail.cf ファイルに、Cw example.com を追加します)。

送信側の sendmail が、メールを配送しようとしている時、モデムの接続を介してあなたのところ (example.com) に接続しようとします。大抵の場合、あなたのマシンがオンラインでないために、接続はタイムアウトしてしまうでしょう。sendmail プログラムは自動的に、たとえばあなたのインターネットプロバイダなどのセカンダリの MX サイト (example.net) にメールを配送するでしょう。セカンダリ MX サイトは定期的にあなたのホストに接続し、プライマリ MX ホスト (example.com) にメールを配送しようとするでしょう。

ログインスクリプトとして、このようなものを使うとよいでしょう。

```
#!/bin/sh
# Put me in /usr/local/bin/pppmyisp
( sleep 60 ; /usr/sbin/sendmail -q ) &
/usr/sbin/ppp -direct pppmyisp
```

ユーザごとにログインスクリプトを作りたい場合には、上記のスクリプトの代わりに、sendmail -qRexample.com を使用することもできます。このようにすると、キューの中の example.com に対するすべてのメールは、すぐに強制的に処理されます。

さらに、次のような改良もできます。

以下は、FreeBSD Internet service provider's メーリングリスト から抜粋してきたメッセージです。

```
> 私たちはお客様に対して、セカンダリ MX を提供しています。
> お客様は一日に何回か私たちのサービスに接続し、メールを彼らのプライマリ MX
> に受け取ります（彼らのドメインに対するメールが到着した時には、
> 私たちは彼らのサイトを呼び出しません)。
> 私たちの sendmail は、30 分ごとにメールキューに溜っているメールを配送します。
> ちょうどその時に、すべてのメールがプライマリ MX
に送られたかどうかを確かめるためには、
> 彼らは 30 分は オンラインでいなければなりません。
>
> すべてのメールを今すぐ送るために sendmail を初期化するコマンドはあるでしょうか?
> もちろん私たちのマシン上には、ユーザはルート（root）権限を持っていません。

sendmail.cf の privacy flags セクションに、
Opgoaway,restrictqrun の定義があります。

root 以外のユーザがキューを処理できるようにするには、
restrictqrun を削除してください。また、MX の再調整が必要かもしれません。
あなたがたは、顧客のサイトに対する一番優先度の高い MX なので、
次のように定義します。
```

```
# If we are the best MX for a host, try directly instead of generating
# local config error.
OwTrue
```

このようにすると、リモートサイトからのメールが、
顧客のマシンと接続しようとせず、直接あなたがたのホストマシンに配送されるようになります
。
ホストマシンに配送されたメールは、続いて顧客のマシンに送られます。
これはホスト名にのみ有効なので、顧客のメールマシンに、
host.customer.com とは別に、customer.com も定義する必要があります。
DNS 上で、customer.com に対する A レコードを定義してください。

19.5.4. なぜ他のホストにメールを送ろうとすると、いつも Relaying Denied と怒られてしまうの？

FreeBSD がインストールされたデフォルトの状態では、sendmail は動作しているホストからのメールだけを送るように設定されています。たとえば POP3 サーバがインストールされているとすると、
ユーザは学校や職場など他のリモートの場所からメールを確認することが
できます。しかし、彼らは外部からそのホスト以外へのメールを送ることはやはりできません。
通常、メールを送ろうとしてから少しすると、5.7 Relaying Denied というエラーメッセージの書かれたメールが MAILER-DAEMON から送られてくるでしょう。

これを解決する方法はいくつかあります。一番の正攻法は /etc/mail/relay-domains リレードメインファイルにあなたの ISP のアドレスを書くことです。
これをするのに簡単な方法は次のとおりです。

```
# echo "your.isp.example.com" > /etc/mail/relay-domains
```

このファイルを作成または編集したら、sendmail を再起動してください。
もしあなたがサーバ管理者でメールをローカルに送りたくないか、ポイントを使用して他のマシン(や、さらに他の ISP) の クライアントまたはシステムへ送りたい時は、とても効果があります。
さらに、あなたが一つあるいは二つだけのメールアカウントを
設定している場合でもこれは非常に有用です。追加すべきアドレスがたくさんある場合には、
単にこのファイルをあなたの好きなテキストエディタで開いて、
そして一行に一つずつドメインを追加してください。

```
your.isp.example.com
other.isp.example.net
users-isp.example.org
www.example.org
```

これで、リストに掲載されているすべてのホスト
(ユーザがあなたのシステムにアカウントを持っていると規定する)
からあなたのシステムを通るすべてのメールは送信に成功するでしょう。これはあなたのシステムから SPAM を送ることを認めることなく、リモートであなたのシステムからメールを送ることをユーザに認めるためのとてもよい方法です。

19.6. 先進的なトピックス

これからのセクションでは、
メールの設定やドメイン全体のためのメールの設定といったさらに突込んだ話題について触れます。

19.6.1. 基本事項

あなたのマシンに FreeBSD を普通にインストールして、 /etc/resolv.conf ファイルを設定するか、またはネームサーバを走らせれば、 他のホストへ電子メールを送ることができるようになります。 あなたのホスト宛のメールをあなた自身の FreeBSD ホスト上の MTA (たとえば sendmail) に配送するようにしたい場合には、次の二つの方法があります。

- 自身でネームサーバーを実行し、 自分のドメインを持つ。例えば FreeBSD.org。
- あなたのホストへ直接メールが配送されるようにする。 これはメールがあなたのマシンの現在の DNS 名に直接配送されるようにすることにより実現できます。 たとえば example.FreeBSD.org。

上のどちらを選ぶ場合でも、自分のホストに直接メールが配送されるようにするには恒久的で 静的 な IP アドレス (ほとんどの PPP ダイアルアップ設定で用いられる動的なアドレスではなく) を持っていなければなりません。 もしファイアウォールの中にいるならば、 SMTP トラフィックが通過してくれないといけません。 もし自分のホストでメールを直接受け取りたいならば、次の二つのうちのどちらかができていることを確認してください。

- 自分のドメインでの (一番値の小さい) MX レコードが自分のホストの IP アドレスを差していることを確認する。
- 自分のドメインの中に自分のホスト用の MX エントリがないことを確認する。

上のどちらかが設定されていれば、 自分のホストでメールを受け取ることができるでしょう。

次のコマンドを実行してみてください。

```
# hostname
example.FreeBSD.org
# host example.FreeBSD.org
example.FreeBSD.org has address 204.216.27.XX
```

もしあなたのマシンが上記のメッセージだけを出力したならば、 yourlogin@example.FreeBSD.org へのメールは問題なく配送されるでしょう (sendmail が example.FreeBSD.org 上で正しく動作していると仮定します)。

上記のメッセージの代わりに、

```
# host example.FreeBSD.org
example.FreeBSD.org has address 204.216.27.XX
example.FreeBSD.org mail is handled (pri=10) by hub.FreeBSD.org
```

というメッセージが出力された場合は、 あなたのホスト (example.FreeBSD.org) に宛てたメールは全て直接配送されずに hub 上の同じユーザー名に配送されます。

上の情報は DNS サーバーが扱います。 メールルーティング情報をもつ DNS レコードは、 *M*ail e *X*change エントリーです。 MX エントリが存在しない場合には、IP アドレスにしたがって、直接宛先ホストに配送されます。

freefall.FreeBSD.org の現時点での MX エントリは、次のようになっています。

```
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 は多くの MX エントリを持っています。 一番 MX の値の小さいホストが利用可能な場合は直接メールを受け取ります。
もしなにかの理由でアクセスができない時には、 他のホスト (ときどき "バックアップ MX" と呼ばれます) が一時的にメールを受け取ります。そして、
より値の小さいホストが利用可能になったときにメールを渡し、
最終的に一番値の小さいホストに渡ります。

使い勝手をよくするためには、代替の MX サイトは、それぞれ別の経路でインターネットへ接続しているとよいでしょう。
インターネットプロバイダまたは他の関連サイトが、このサービスを 提供することができます。

19.6.2. あなたのドメインに対するメール設定

"メールホスト" (メールサーバーとしても知られています) をセットアップするためには、いろいろなワークステーションに宛てた全てのメールを受ける必要があります。
基本的には、あなたのドメイン内 (この場合だと *.FreeBSD.org) のすべてのホスト名宛てのすべてのメールを "受け取って"、
そのメールをあなたのメールサーバーに配送し、
ユーザーがマスタメールサーバ上でメールをチェックできるようにします。

話を簡単にするために、あるユーザーのアカウントはどのマシンでも同じユーザー名にすべきです。
そのためには adduser(8) を使ってください。

使用する予定のメールホストは、各ワークステーションごとにメール交換ができるように設定されていなければなりません。 これは DNS の設定で次のように行なうことができます。

```
example.FreeBSD.org A   204.216.27.XX       ; ワークステーション
MX  10 hub.FreeBSD.org  ; メールホスト
```

これは、ワークステーションの A レコードがどこを指していようともそのワークステーション宛てのメールをメールホストに転送する、というものです。

自前で DNS サーバを運用しているのでなければ、 この作業は自分では行えません。自分で DNS サーバを運用しないとかできないという場合は、 あなたの DNS を提供しているインターネットプロバイダなどに依頼して 作業を行ってもらってください。

もしバーチャル電子メールホストを運用するなら次の情報が役に立つでしょう。

例として、あなたには自分のドメイン、ここでは customer1.org、を持っている顧客がいるとしましょう。 あなたは customer1.org 宛ての全てのメールを mail.myhost.com というメールホストに集めたいとします。 DNS エントリーは次のようになるでしょう。

```
customer1.org       MX  10 mail.myhost.com
```

customer1.org に対して電子メールを送りたいだけなら、 A レコードは必要ありません。

customer1.org に対して ping を実行しても、 A レコードが存在しない限りうまくいかないことに留意しておいてください。

やらなければいけない最後のことは、メールホスト上の sendmail に対してどんなドメインやホスト宛のメールを受け取るのか、を教えることです。いくつかの方法がありますが次のどちらかでいいでしょう。

- FEATURE(use_cw_file) を使っているなら、/etc/mail/local-host-names ファイルにホストを加えます。 もし sendmail のバージョンが 8.10 より前であれば該当ファイルは /etc/sendmail.cw です。
- /etc/sendmail.cf もしくは sendmail 8.10 以降なら /etc/mail/sendmail.cf といったファイルに Cwyour.host.com という行を加えます。

19.7. UUCP とともに SMTP を使う

FreeBSD とともに出荷されている sendmail の設定は、サイトがインターネットに直接接続しているものとして設計されています。 UUCP 経由でメールを交換したいサイトは、他の sendmail 設定ファイルをインストールしなければいけません。

/etc/mail/sendmail.cf を手動で調整することは先進的なトピックです。 sendmail のバージョン 8 は設定ファイルを m4(1) プリプロセッサから生成します。これにより、高度に抽象化された設定を行うことができます。 m4(1) による設定ファイルは /usr/src/usr.sbin/sendmail/cf 以下にあります。

もしシステムをすべてのソースとともにインストールしていなければ、sendmail の設定材料は分割された個別のソース tarball を取得してください。 FreeBSD のソースコードが入った CDROM をマウントしているのなら、

```
# cd /cdrom/src
# cat scontrib.?? | tar xzf - -C /usr/src/contrib/sendmail
```

と展開してください (展開してもたった数百 KB 程度です)。 cf ディレクトリの README ファイルは m4(1) による設定の基本的な手引として役に立つでしょう。

UUCP 配送に対応するための一番よい方法は mailertable 機能を使用することです。これは経路を決定するために sendmail が使用できるデータベースを作成します。

まずはじめに .mc ファイルを作成しなければいけません。 /usr/src/usr.sbin/sendmail/cf/cf

にいくつか例があります。foo.mc という名前のファイルをあなたが作成したとすると、有効な sendmail.cf ファイルへ変換するには次のようにするだけです。

```
# cd /usr/src/usr.sbin/sendmail/cf/cf
# make foo.cf
# cp foo.cf /etc/mail/sendmail.cf
```

典型的な .mc ファイルは次のようになるでしょう。

```
VERSIONID(`Your version number') OSTYPE(bsd4.4)

FEATURE(accept_unresolvable_domains)
FEATURE(nocanonify)
FEATURE(mailertable, `hash -o /etc/mail/mailertable')

define(`UUCP_RELAY', your.uucp.relay)
define(`UUCP_MAX_SIZE', 200000)
define(`confDONT_PROBE_INTERFACES')

MAILER(local)
MAILER(smtp)
MAILER(uucp)

Cw    your.alias.host.name
Cw    youruucpnodename.UUCP
```

accept_unresolvable_domains, nocanonify および confDONT_PROBE_INTERFACES 機能を含んでいる行は、メール配送時にまったく DNS を使用しません。UUCP_RELAY の記述は UUCP 配送に対応するのに必要です。そこにインターネットホスト名を単に書くだけで .UUCP pseudo ドメインアドレスを扱うことができるようになります。大抵の場合、あなたの ISP のメールリレーをそこに入力するでしょう。

次に、/etc/mail/mailertable が必要になります。
メールを配送するリンクが外界との間に一つだけの場合は、
次のようにファイルを記述するだけで十分でしょう。

```
#
# makemap hash /etc/mail/mailertable.db < /etc/mail/mailertable
.                               uucp-dom:your.uucp.relay
```

次はさらに複雑な例です。

```
#
# makemap hash /etc/mail/mailertable.db < /etc/mail/mailertable
#
horus.interface-business.de   uucp-dom:horus
.interface-business.de        uucp-dom:if-bus
```

```
interface-business.de          uucp-dom:if-bus
.heep.sax.de                   smtp8:%1
horus.UUCP                     uucp-dom:horus
if-bus.UUCP                    uucp-dom:if-bus
.                              uucp-dom:
```

はじめの三行はドメインで宛先を指定されたメールが、配送路を"近道"するために、デフォルトルートではなく代わりにいくつかの UUCP 隣接ホストへ送られる特別な場合を扱います。次の行はメールを SMTP で配送可能なローカルイーサネットドメインへ送ります。最後に `uucp-neighbor !recipient` がデフォルトルートを上書きすることを許可するための UUCP 隣接ホストは .UUCP 仮想ドメイン記法で言及されます。最後の行は常に他のすべてが当てはまるシングルドットです。これは UUCP 隣接ホストへの UUCP 配送をすることで、世界に向けたあなたの普遍的メールゲートウェイとして役に立ちます。`uucp-dom:` キーワードの後ろにあるノード名はすべて、`uuname` コマンドを使用することで確かめられる正しい UUCP 隣接ホストである必要があります。

このファイルは、実際に使用する前に DBM データベース形式に変換する必要があることに注意してください。これを実行するコマンドラインは mailertable ファイルの先頭にコメントとして書かれています。mailertable を変更するたびにいつもこのコマンドを実行する必要があります。

最後のアドバイス: もし、いくつかのメールルーティングがうまく動いているかどうか分からないときは sendmail に `-bt` オプションをつけることを覚えておいてください。これは sendmail をアドレステストモードで起動します。あなたがテストしたいメールルーティングのアドレスを後につけて、単純に `3,0` と入力してください。最後の行は、内部で使われたメールエージェント、このエージェントが呼び出された目的地ホスト、および(もしかしたら変換された)アドレスを表示します。このモードを終了するには `Ctrl` + `D` を入力します。

```
% sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 foo@example.com
canonify           input: foo @ example . com
...
parse            returns: $# uucp-dom $@ your.uucp.relay $: foo < @ example . com . >
> ^D
```

19.8. ダイアルアップ接続でメールを使う

あなたが固定 IP アドレスを持っているのなら、デフォルトから何も変更する必要はありません。割りあてられたインターネット名をホスト名に設定すれば、sendmail が残りをやってくれます。

あなたが動的に割り当てられた IP アドレスを持っていて、インターネットに接続するのにダイアルアップ PPP を使用しているのなら、おそらく ISP のメールサーバにメールボックスがあるでしょう。ここでは、あなたの ISP のドメインが `example.net`, あなたのユーザ名が `user`, あなたのマシンは `bsd.home` と呼ばれているものとします。また、ISP から、メールリレーとして `relay.example.net` を使用してよいと通知されているとします。

(訳注: ISP 上の) メールボックスからメールを取得するためには、取得アプリケーションをインストールしないといけません。 fetchmail ユーティリティは、さまざまなプロトコルの多くに対応しているのでよい選択肢です。 通常、あなたの ISP は POP3 を提供しています。 このプログラムは、mail/fetchmail package または Ports Collection からインストールできます。 あなたが ユーザ PPP を使用しているなら、次のエントリを /etc/ppp/ppp.linkup に追加することで、インターネット接続が確立したときに自動的にメールを取得することができます。

```
MYADDR:
 !bg su user -c fetchmail
```

あなたがローカルではないアカウントへのメールを配送するために (下記のような) sendmail を使用しているなら、 インターネット接続が確立するとすぐに、 sendmail があなたのメールキューを処理して欲しいとおそらく考えるでしょう。
これを行うには、/etc/ppp/ppp.linkup ファイルの fetchmail コマンドの後に次のコマンドを追加してください。

```
 !bg su user -c "sendmail -q"
```

bsd.home 上に user というアカウントを所有しているとします。 bsd.home 上の user のホームディレクトリに .fetchmailrc ファイルを作成します。

```
poll example.net protocol pop3 fetchall pass MySecret
```

このファイルはパスワード MySecret を含んでいるので、user を除く他の誰にも読めるようになっていてはいけません。

正しい from: ヘッダでメールを送るためには、 sendmail が user@bsd.home ではなく user@example.net を使用するようにしなくてはいけません。 また、素早くメール送信をするために sendmail にすべてのメールを relay.example.net 経由で送るようにもしたいかもしれません。

次の .mc ファイルで十分でしょう。

```
VERSIONID(`bsd.home.mc version 1.0')
OSTYPE(bsd4.4)dnl
FEATURE(nouucp)dnl
MAILER(local)dnl
MAILER(smtp)dnl
Cwlocalhost
Cwbsd.home
MASQUERADE_AS(`example.net')dnl
FEATURE(allmasquerade)dnl
FEATURE(masquerade_envelope)dnl
FEATURE(nocanonify)dnl
FEATURE(nodns)dnl
define(`SMART_HOST', `relay.example.net')
Dmbsd.home
```

```
define(`confDOMAIN_NAME',`bsd.home')dnl
define(`confDELIVERY_MODE',`deferred')dnl
```

.mc ファイルを sendmail.cf ファイルに変換する方法の詳細については前の節を参照してください。
また、sendmail.cf ファイルを変更した後は、 sendmail を再起動し忘れないでください。

19.9. SMTP 認証

メールサーバ上で SMTP 認証を行うと、 多くの利益があります。 SMTP 認証は sendmail にもう一つのセキュリティ層を追加することができます。
さらに、ホストを切りかえるモバイルユーザにとっては、
その都度メールクライアントの設定を変更せずとも 同じメールサーバを利用できるようになります。

1. ports から security/cyrus-sasl をインストールします。 この port は security/cyrus-sasl にあります。 security/cyrus-sasl にはここで使用する方法に対する多くのコンパイルオプションがあり、 確実に pwcheck オプションを選択してください。
2. security/cyrus-sasl をインストールした後に /usr/local/lib/sasl/Sendmail.conf を編集して (もし無ければ作成して) 次の行を追加してください。

   ```
   pwcheck_method: passwd
   ```

   この方法は sendmail があなたの FreeBSD の passwd データベースに対して認証することを可能にします。 この方法は SMTP 認証に必要となる、それぞれのユーザに対する一組の新しいユーザ名とパスワードを
   作成する際のトラブルを減らし、 ログインパスワードとメールパスワードを同じままにします。

3. ここで /etc/make.conf 編集し、 次の行を加えます。

   ```
   SENDMAIL_CFLAGS=-I/usr/local/include/sasl1 -DSASL
   SENDMAIL_LDFLAGS=-L/usr/local/lib
   SENDMAIL_LDADD=-lsasl
   ```

   これらの行は sendmail に対して、 コンパイルするときに cyrus-sasl とリンクするための適切な設定オプションを与えるものです。 sendmail を再コンパイルする前に cyrus-sasl がインストールされていることを確かめてください。

4. 次のコマンドを入力して sendmail を再コンパイルしてください。

   ```
   # cd /usr/src/usr.sbin/sendmail
   # make cleandir
   # make obj
   # make
   # make install
   ```

sendmail のコンパイルは /usr/src が大幅に変更されていなくて、必要な共有ライブラリが利用可能であれば何の問題も起こらないでしょう。

5. sendmail をコンパイルして再インストールした後は、/etc/mail/freebsd.mc ファイル (またはあなたが .mc ファイルとして使用しているファイル。多くの管理者は唯一の名前を用いるために hostname(1) の出力を .mc として使用することを選んでいます) を編集してください。 次の行を加えてください。

```
dnl set SASL options
TRUST_AUTH_MECH(`GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN')dnl
define(`confAUTH_MECHANISMS', `GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN')dnl
define(`confDEF_AUTH_INFO', `/etc/mail/auth-info')dnl
```

これらのオプションは、ユーザを認証するために sendmail が利用可能な異なる方法を設定します。 もし pwcheck 以外の方法を使用したいのならドキュメントを参照してください。

6. 最後に /etc/mail で make(1) を実行してください。 これにより、新しい .mc ファイルから freebsd.cf という名前 (またはあなたの .mc に使用している名前) の .cf ファイルが作成されます。 それから `make install restart` コマンドを実行してください。新しい .cf ファイルが sendmail.cf にコピーされ、 sendmail が適切に再起動されるでしょう。この手続きについての詳細は /etc/mail/Makefileを参照してください。

すべてがうまくいけば、ログイン情報をメールクライアントに入力し、テストメッセージを送ることができるでしょう。 より詳細に調べるには sendmail の `LogLevel` を 13 に設定し、 すべてのエラーについて /var/log/maillog を見てください。

このサービスがシステムを起動した後にいつでも利用可能となるように、/etc/rc.conf に次の行を追加しておくとよいでしょう。

```
sasl_pwcheck_enable="YES"
sasl_pwcheck_program="/usr/local/sbin/pwcheck"
```

これにより、システムの起動時に SMTP_AUTH が確実に初期化されるでしょう。

詳細については SMTP 認証 に関する sendmail の文書を参照してください。

Chapter 20. 高度なネットワーク

20.1. この章では

この章では UNIX® システム上で良く利用されるネットワークサービスについて説明します。 FreeBSD が利用するすべてのネットワークサービスをどのように定義し、
設定し、テストし、そして保守するのかを扱います。さらに、
本章を通してあなたの役に立つ設定例が載っています。

この章を読めば以下のことが分かります。

- ゲートウェイと経路の基本
- FreeBSD をブリッジとして動作させる方法
- ネットワークファイルシステム (NFS) の設定方法
- ディスクレスマシンのネットワークブートの設定方法
- ユーザアカウントを共有するためのネットワークインフォメーションサーバ (NIS) の設定方法
- DHCP を用いて自動的にネットワーク設定を行う方法
- ドメインネームサーバ (DNS) の設定方法
- NTP プロトコルを用いて日時を同期してタイムサーバを設定する方法
- ネットワークアドレス変換 (NAT) の設定方法
- `inetd` デーモンの管理方法
- PLIP 経由で二台のコンピュータを接続する方法
- FreeBSD で IPv6 を設定する方法

この章を読む前に、以下のことを行っておくべきです。

- /etc/rc スクリプトの基本を理解していること
- 基礎的なネットワーク用語に精通していること

20.2. ゲートウェイと経路

あるマシンがネットワーク上で他のマシンをみつけることができるようにするには、
あるマシンから他のマシンへどのようにたどり着くかを記述する適切な仕組みが必要です。
この仕組みをルーティングと呼びます。 "経路" (route) は "送信先" (destination) と "ゲートウェイ" の 2 つのアドレスの組で定義します。この組合せは、この 送信先 へたどり着こうとする場合は、そのゲートウェイ を通じて通信することを示しています。 送信先には個々のホスト、サブネット、"デフォルト" の 3 つの型があります。 "デフォルトルート" は他のどの経路も適用できない場合に使われます。
デフォルトルートについてはのちほどもう少し詳しく述べます。
また、ゲートウェイには、個々のホスト、インタフェース ("リンク" とも呼ばれます)、イーサネットハードウェアアドレス (MAC アドレス) の 3 つの型があります。

20.2.1. 例

以下に示す netstat の例を使って、ルーティングのさまざまな状態を説明します。

```
% netstat -r
Routing tables

Destination      Gateway            Flags     Refs     Use     Netif Expire

default          outside-gw         UGSc       37      418      ppp0
localhost        localhost          UH          0      181       lo0
test0            0:e0:b5:36:cf:4f   UHLW        5    63288       ed0     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
```

最初の 2 行はデフォルトルート (次節で扱います) と、 localhost への経路を示しています。

localhost に割り当てるインタフェース (Netif 欄) としてこのルーティングテーブルが指定しているのは lo0 で、これはループバックデバイスともいいます。 これは結局のところ出たところに戻るだけなので、この送信先あてのトラフィックは、LAN に送られずに、すべて内部的に処理されます。

次の行では 0:e0: から始まるアドレスに注目しましょう。これはイーサネットハードウェアアドレスで、MAC アドレスともいいます。 FreeBSD はローカルなイーサネット上の任意のホスト (この例では test0) を自動的に認識し、イーサネットインタフェース ed0 にそのホストへの直接の経路をつけ加えます。この種の経路には、タイムアウト時間 (Expire 欄) も結びつけられており、指定された時間内にホストからの応答がないことを判断するのに用いられます。その場合、そのホストへの経路情報は自動的に削除されます。 これらのホストは RIP (Routing Information Protocol) という、最短パス判定に基づいてローカルなホストへの経路を決定する仕組みを利用して認識されます。

さらに FreeBSD ではローカルサブネットへの経路情報も加えることができます (10.20.30.255 は 10.20.30 というサブネットに対するブロードキャストアドレスで、 example.com はこのサブネットに結びつけられているドメイン名)。 link#1 という名称は、このマシンの一つ目のイーサネットカードのことをさします。 これらについては、何も追加インタフェースが指定されていないことがわかります。

これら 2 つのグループ (ローカルネットワークホストとローカルサブネット) は、両方とも routed というデーモンによって自動的に経路が設定されます。 routed を動かさなければ、静的に定義した (つまり明示的に設定した) 経路のみが存在することになります。

host1 の行は私たちのホストのことで、イーサネットアドレスで示されています。送信側のホストの場合、FreeBSDはイーサネットインタフェースへ送るのではなく、 ループバックインタフェース (lo0) を使います。

2 つある host2 の行は、 ifconfig(8) のエイリアスを使ったときにどのようになるかを示す例です (このようなことをする理由については Ethernet の節を参照してください)。 lo0 の後にある ⇒ は、インタフェースが (このアドレスがローカルなホストを参照しているので) ループバックを使っているというだけでなく、 エイリアスになっていることも示しています。
このような経路はエイリアスに対応しているホストにのみ現れます。
ローカルネットワーク上の他のすべてのホストでは、 それぞれの経路に対して単にlink#1 となります。

最後の行 (送信先サブネット 224) はマルチキャストで扱うものですが、これは他の節で説明します。

最後に Flags (フラグ) 欄にそれぞれの経路のさまざまな属性が表示されます。
以下にフラグの一部と、それが何を意味しているかを示します。

U	Up: この経路はアクティブです。
H	Host: 経路の送信先が単一のホストです。
G	Gateway: この送信先へ送られると、 どこへ送ればよいかを明らかにして、 そのリモートシステムへ送られます。
S	Static: この経路はシステムによって自動的に生成されたのではなく、 手動で作成されました。
C	Clone: マシンに接続したときにこの経路に基づく新しい経路が作られます。 この型の経路は通常はローカルネットワークで使われます。
W	WasCloned: ローカルエリアネットワーク (LAN) の (Clone) 経路に基づいて自動的に生成された経路であることを示します。
L	Link: イーサネットハードウェアへの参照を含む経路です。

20.2.2. デフォルトルート

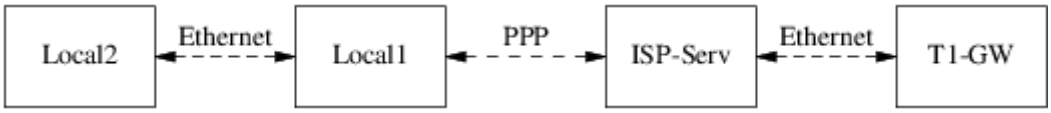
ローカルシステムからリモートホストにコネクションを張る必要がある場合、
既知の経路が存在するかどうかを確認するためにルーティングテーブルをチェックします。
到達するための経路を知っているサブネットの内部にリモートホストがある場合 (Cloned routes)、システムはそのインタフェースから接続できるかどうか確認します。

知っているパスがすべて駄目だった場合でも、 システムには最後の手段として "デフォルト" ルートがあります。このルートはゲートウェイルート (普通はシステムに 1 つしかありません) の特別なものです。そして、 フラグ欄には必ず c が表示されています。このゲートウェイは、LAN 内のホストにとって、どのマシンでも外部へ (PPP リンク、DSL、ケーブルモデム、T1、またはその他のネットワークインタフェースのいずれかを経由して)
直接接続するために設定されるものです。

外部に対するゲートウェイとして機能するマシンでデフォルトルートを設定する場合、
デフォルトルートはインターネットサービスプロバイダ (ISP)

のサイトのゲートウェイマシンになるでしょう。

それではデフォルトルートの一例を見てみましょう。 一般的な構成を示します。



ホスト Local1 とホスト Local2 はあなたのサイト内にあります。Local1 はダイアルアップ PPP 接続経由で ISP に接続されています。 この PPP サーバコンピュータは、その ISP のインターネットへの接続点に向けた外部インタフェースを備えた他のゲートウェイコンピュータへ LAN を通じて接続しています。

あなたのマシンのデフォルトルートはそれぞれ次のようになります。

ホスト	デフォルトゲートウェイ	インタフェース
Local2	Local1	Ethernet
Local1	T1-GW	PPP

"なぜ (あるいは、どうやって) デフォルトゲートウェイを、Local1 が接続されている ISP のサーバではなく、T1-GW に設定するのか" という質問がよくあります。

PPP 接続で、あなたのサイト側の PPP インタフェースは、ISP のローカルネットワーク上のアドレスを用いているため、ISP のローカルネットワーク上のすべてのマシンへの経路は自動的に生成されています。つまりあなたのマシンは、どのようにして T1-GW に到達するかという経路を既に知っていることになりますから、 ISP サーバにトラフィックを送るのに、中間的な段階を踏む必要はありません。

一般的にローカルネットワークでは X.X.X.1 というアドレスをゲートウェイアドレスとして使います。ですから (同じ例を用います)、あなたの class-C のアドレス空間が 10.20.30 で ISP が 10.9.9 を用いている場合、デフォルトルートは次のようになります。

ホスト	デフォルトルート
Local2 (10.20.30.2)	Local1 (10.20.30.1)
Local1 (10.20.30.1, 10.9.9.30)	T1-GW (10.9.9.1)

デフォルトルートは /etc/rc.conf ファイルで簡単に定義できます。この例では、 Local2 マシンで /etc/rc.conf に次の行を追加しています。

```
defaultrouter="10.20.30.1"
```

route(8) コマンドを使ってコマンドラインから直接実行することもできます。

```
# route add default 10.20.30.1
```

経路情報を手動で操作する方法について詳しいことは route(8) のマニュアルページをご覧ください。

20.2.3. デュアルホームホスト

ここで扱うべき種類の設定がもう一つあります。 それは 2 つの異なるネットワークにまたがるホストです。 技術的にはゲートウェイとして機能するマシン (上の例では PPP コネクションを用いています) はすべてデュアルホームホストです。 しかし実際にはこの言葉は、2 つの LAN 上のサイトであるマシンを指す言葉としてのみ使われます。

2 枚のイーサネットカードを持つマシンが、別のサブネット上にそれぞれアドレスを持っている場合があります。 あるいは、イーサネットカードが 1 枚しかないマシンで、 ifconfig(8) のエイリアスを使っているかもしれません。 物理的に分かれている 2 つのイーサネットのネットワークが使われているならば前者が用いられます。 後者は、物理的には 1 つのネットワークセグメントで、 論理的には 2 つのサブネットに分かれている場合に用いられます。

どちらにしても、 このマシンがお互いのサブネットへのゲートウェイ (inbound route) として定義されていることが分かるように、
おのおののサブネットでルーティングテーブルを設定します。このマシンが 2 つのサブネットの間のルータとして動作するという構成は、
パケットのフィルタリングを実装する必要がある場合や、
一方向または双方向のファイアウォールを利用したセキュリティを構築する場合によく用いられます。

このマシンが二つのインタフェース間で実際にパケットを受け渡すようにしたい場合は、 FreeBSD でこの機能を有効にしないといけません。 くわしい手順については次の節をご覧ください。

20.2.4. ルータの構築

ネットワークルータは単にあるインタフェースから別のインタフェースへパケットを転送するシステムです。 インターネット標準およびすぐれた技術的な慣習から、 FreeBSD プロジェクトは FreeBSD においてこの機能をデフォルトでは有効にしていません。 rc.conf(5) 内で次の変数を YES に変更することでこの機能を有効にできます。

```
gateway_enable=YES          # Set to YES if this host will be a gateway
```

このオプションは sysctl(8) 変数の net.inet.ip.forwarding を 1 に設定します。 一時的にルーティングを停止する必要があるときには、 この変数を一時的に 0 に設定しなおせます。

次に、トラフィックの宛先を決めるために、 そのルータには経路情報が必要になります。 ネットワークが十分簡素なら、静的経路が利用できます。 また、FreeBSD は BSD の標準ルーティングデーモンである routed(8) を備えています。これは RIP (バージョン 1 および 2) および IRDP を扱えます。 BGP バージョン 4、OSPF バージョン2、その他洗練されたルーティングプロトコルは net/zebra package を用いれば対応できます。 また、より複雑なネットワークルーティングソリューションには、 GateD® のような商用製品も利用可能です。

このように FreeBSD を設定したとしても、ルータに対するインターネット標準要求を完全に満たすわけではありません。
しかし、通常利用に関しては十分といえます。

20.2.5. 静的な経路の設定

手動による経路の設定

以下のようなネットワークが存在すると仮定します。

```
   INTERNET
      | (10.0.0.1/24) Default Router to Internet
      |
      |Interface xl0
      |10.0.0.10/24
   +------+
   |      | RouterA
   |      | (FreeBSD gateway)
   +------+
      | Interface xl1
      | 192.168.1.1/24
      |
  +--------------------------------+
   Internal Net 1      | 192.168.1.2/24
                       |
                   +------+
                   |      | RouterB
                   |      |
                   +------+
                       | 192.168.2.1/24
                       |
                     Internal Net 2
```

このシナリオでは、FreeBSD マシンの RouterA がインターネットに向けられたルータとして動作します。ルータは外側のネットワークへ接続できるように 10.0.0.1 へ向けたデフォルトルートを保持しています。 RouterB はすでに適切に設定されており、どこへ向かう必要があるか、 行き着く方法を知っていると仮定します (この例では、図のように簡単です。 192.168.1.1 をゲートウェイとして RouterB にデフォルトルートを追加するだけです)。

RouterA のルーティングテーブルを確認すると、 以下のような出力を得ます。

```
% 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
10.0.0/24          link#1             UC           0        0    xl0
192.168.1/24       link#2             UC           0        0    xl1
```

現在のルーティングテーブルでは、RouterA はまだ Internal Net 2 には到達できないでしょう。

192.168.2.0/24 の経路を保持していないからです。
解決するための一つの方法は、経路を手動で追加することです。 以下のコマンドで RouterA のルーティングテーブルに 192.168.1.2 を送り先として、Internal Net 2 ネットワークを追加します。

```
# route add -net 192.168.2.0/24 192.168.1.2
```

これにより、RouterA は、 192.168.2.0/24 ネットワーク上のホストに到達出来ます。

永続的な設定

上記の例は、 起動しているシステム上に静的な経路を設定する方法としては完全です。
しかしながら、FreeBSD
マシンを再起動した際にルーティング情報が残らないという問題が一つあります。
静的な経路を追加するには、/etc/rc.conf ファイルにルートを追加してください。

```
# Add Internal Net 2 as a static route
static_routes="internalnet2"
route_internalnet2="-net 192.168.2.0/24 192.168.1.2"
```

static_routes の設定変数は、 スペースによって分離される文字列のリストです。
それぞれの文字列は経路名として参照されます。 上記の例では static_routes は一つの文字列のみを持ちます。 その文字列は *internalnet2* です。その後、 route_internalnet2 という設定変数を追加し、 route(8) コマンドに与えるすべての設定パラメータを指定しています。
前節の例では、以下のコマンド

```
# route add -net 192.168.2.0/24 192.168.1.2
```

を用いたので、 "-net 192.168.2.0/24 192.168.1.2" が必要になります。

上記のように static_routes は一つ以上の文字列を持つことが出来るので、
多数の静的な経路を作ることができます。 以下の行は 192.168.0.0/24 および 192.168.1.0/24 ネットワークを、 仮想ルータ上に静的な経路として追加する例です。

```
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"
```

20.2.6. ルーティングの伝搬

外部との経路をどのように定義したらよいかはすでに説明しました。
しかし外部から私たちのマシンをどのようにして見つけるのかについては説明していません。

ある特定のアドレス空間 (この例では class-C のサブネット) におけるすべてのトラフィックが、到着したパケットを内部で転送するネットワーク上の特定のホストに送られるようにルーティングテーブルを設定することができるのは分かっています。

あなたのサイトにアドレス空間を割り当てる場合、

あなたのサブネットへのすべてのトラフィックがすべて PPP
リンクを通じてサイトに送ってくるようにサービスプロバイダはルーティングテーブルを設定します。
しかし、国境の向こう側のサイトはどのようにしてあなたの ISP へ送ることを知るのでしょうか?

割り当てられているすべてのアドレス空間の経路を維持する (分散している DNS 情報とよく似た)
システムがあり、 そのインターネットバックボーンへの接続点を定義しています。 "バックボーン"
とは国を越え、 世界中のインターネットのトラフィックを運ぶ主要な信用できる幹線のことです。
どのバックボーンマシンも、
あるネットワークから特定のバックボーンのマシンへ向かうトラフィックと、
そのバックボーンのマシンからあなたのネットワークに届くサービスプロバイダまでのチェーンのマスタ
テーブルのコピーを持っています。

あなたのサイトが接続 (プロバイダからみて内側にあることになります) したということを、
プロバイダからバックボーンサイトへ通知することはプロバイダの仕事です。 これが経路の伝搬です。

20.2.7. トラブルシューティング

経路の伝搬に問題が生じて、 いくつかのサイトが接続をおこなうことができなくなることがあります。
ルーティングがどこでおかしくなっているかを明らかにするのに最も有効なコマンドはおそらく
traceroute(8) コマンドでしょう。
このコマンドは、あなたがリモートマシンに対して接続をおこなうことができない (たとえば ping(8)
に失敗するような) 場合も、同じように有効です。

traceroute(8) コマンドは、 接続を試みているリモートホストを引数にして実行します。
試みている経路が経由するゲートウェイホストを表示し、 最終的には目的のホストにたどり着くか、
コネクションの欠如によって終ってしまうかのどちらかになります。

より詳しい情報は、traceroute(8) のマニュアルページをみてください。

20.2.8. マルチキャストルーティング

FreeBSD
はマルチキャストアプリケーションとマルチキャストルーティングの両方にネイティブ対応しています。
マルチキャストアプリケーションを動かすのに FreeBSD で特別な設定をする必要は一切ありません。
アプリケーションは普通はそのままで動くでしょう。 マルチキャストルーティングに対応するには、
下のオプションを追加してカーネルをコンパイルする必要があります。

```
options MROUTING
```

さらに、/etc/mrouted.conf を編集してルーティングデーモン mrouted(8) を設定し、トンネルと DVMRP
を設置する必要があります。 マルチキャスト設定についての詳細は mrouted(8)
のマニュアルページを参照してください。

20.3. 無線ネットワーク

20.3.1. はじめに

常にネットワークケーブルをつないでいるという面倒なことをせずに、
コンピュータを使用できることは、とても有用でしょう。 FreeBSD は無線のクライアントとして、

さらに "アクセスポイント" としても使えます。

20.3.2. 無線の動作モード

802.11 無線デバイスの設定には、BSS と IBSS の二つの方法があります。

BSS モード

BSS モードは一般的に使われているモードです。 BSS モードはインフラストラクチャモードとも呼ばれています。 このモードでは、多くの無線アクセスポイントが 1 つの有線ネットワークに接続されます。 それぞれのワイヤレスネットワークは固有の名称を持っています。 その名称はネットワークの SSID と呼ばれます。

無線クライアントはこれらの無線アクセスポイントに接続します。 IEEE 802.11 標準は無線ネットワークが接続するのに使用するプロトコルを規定しています。 SSID が設定されているときは、 無線クライアントを特定のネットワークに結びつけることができます。 SSID を明示的に指定しないことにより、 無線クライアントを任意のネットワークに接続することもできます。

IBSS モード

アドホックモードとも呼ばれる IBSS モードは、 一対一通信のために設計された通信方式です。 実際には二種類のアドホックモードがあります。 一つは IBSS モードで、アドホックモード、または IEEE アドホックモードとも呼ばれます。 このモードは IEEE 802.11 標準に規定されています。 もう一つはデモアドホックモードもしくは Lucent アドホックモード (そして時々、紛らわしいことに、アドホックモード) と呼ばれるモードです。 このモードは古く、802.11 が標準化する以前のアドホックモードで、 これは古い設備でのみ使用されるべきでしょう。 ここでは、どちらのアドホックモードについてもこれ以上言及しません。

20.3.3. インフラストラクチャーモード

アクセスポイント

アクセスポイントは一つ以上の無線クライアントが、
そのデバイスをセントラルハブとして利用できるようにする無線ネットワークデバイスです。
アクセスポイントを使用している間、 すべてのクライアントはアクセスポイントを介して通信します。
家屋や職場、または公園などの空間を無線ネットワークで完全にカバーするために、
複数のアクセスポイントがよく使われます。

アクセスポイントは一般的に複数のネットワーク接続 (無線カードと、その他のネットワークに接続するための一つ以上の有線イーサネットアダプタ) を持っています。

アクセスポイントは、出来合いのものを購入することもできますし、 FreeBSD と対応している無線カードを組み合わせて、 自分で構築することもできます。 いくつものメーカが、さまざまな機能をもった無線アクセスポイントおよび無線カードを製造しています。

FreeBSD のアクセスポイントの構築

要件

FreeBSD で無線アクセスポイントを設定するためには、 互換性のある無線カードが必要です。 現状では

Prism チップセットのカードのみに対応しています。 また FreeBSD に対応している有線ネットワークカードも必要になるでしょう (これを見つけるのは難しくないでしょう。 FreeBSD は多くの異なるデバイスに対応しているからです)。 この手引きでは、無線デバイスと有線ネットワークカードに接続しているネットワーク間のトラフィックを bridge(4) したいと仮定します。

FreeBSD がアクセスポイントを実装するのに使用する hostap 機能はファームウェアの特定のバージョンで一番よく性能を発揮します。 Prism 2 カードは、 1.3.4 以降のバージョンのファームウェアで使用すべきです。 Prism 2.5 および Prism 3 カードでは、バージョン 1.4.9 のバージョンのファームウェアで使用すべきです。 それより古いバージョンのファームウェアは、 正常に動くかもしれませんし、動かないかもしれません。 現時点では、カードのファームウェアを更新する唯一の方法は、 カードの製造元から入手できる Windows® 用ファームウェアアップデートユーティリティを使うものです。

設定

はじめにシステムが無線カードを認識していることを確認してください。

```
# ifconfig -a
wi0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        inet6 fe80::202:2dff:fe2d:c938%wi0 prefixlen 64 scopeid 0x7
        inet 0.0.0.0 netmask 0xff000000 broadcast 255.255.255.255
        ether 00:09:2d:2d:c9:50
        media: IEEE 802.11 Wireless Ethernet autoselect (DS/2Mbps)
        status: no carrier
        ssid ""
        stationname "FreeBSD Wireless node"
        channel 10 authmode OPEN powersavemode OFF powersavesleep 100
        wepmode OFF weptxkey 1
```

細かいことは気にせず、 無線カードがインストールされていることを示す何かが表示されていることを確かめてください。 PC カードを使用していて、無線インタフェースを認識できない場合、 詳しい情報を得るために pccardc(8) と pccardd(8) のマニュアルページを調べてみてください。

次に、アクセスポイント用に FreeBSD のブリッジ機能を担う部分を有効にするために、モジュールを読み込む必要があるでしょう。 bridge(4) モジュールを読み込むには、次のコマンドをそのまま実行します。

```
# kldload bridge
```

モジュールを読み込む時には、何もエラーはでないはずです。 もしもエラーがでたら、カーネルに bridge(4) のコードを入れてコンパイルする必要があるかもしれません。 ハンドブックのブリッジの節が、 この課題を成し遂げる手助けをになるかもしれません。

ブリッジ部分が準備できたので、 どのインタフェース間をブリッジするのかを FreeBSD カーネルに指定する必要があります。 これは、sysctl(8) を使って行います。

```
# sysctl net.link.ether.bridge=1
# sysctl net.link.ether.bridge_cfg="wi0,xl0"
# sysctl net.inet.ip.forwarding=1
```

FreeBSD 5.2-RELEASE 以降では、次のように指定しなければなりません。

```
# sysctl net.link.ether.bridge.enable=1
# sysctl net.link.ether.bridge.config="wi0,xl0"
# sysctl net.inet.ip.forwarding=1
```

さて、無線カードを設定するときです。 次のコマンドはカードをアクセスポイントとして設定します。

```
# ifconfig wi0 ssid my_net channel 11 media DS/11Mbps mediaopt hostap up stationname
"FreeBSD AP"
```

この ifconfig(8) コマンド行は wi0 インタフェースを up 状態にし、SSID を *my_net* に設定し、ステーション名を *FreeBSD AP* に設定します。 media DS/11Mbps オプションはカードを 11Mbps モードに設定し、また mediaopt を実際に有効にするのに必要です。 mediaopt hostap オプションはインタフェースをアクセスポイントモードにします。 channel 11 オプションは使用するチャネルを 802.11b に設定します。 各規制地域 (regulatory domain) で有効なチャネル番号は wicontrol(8) マニュアルページに載っています。

さて、 これで完全に機能するアクセスポイントが立ち上がり、動作しています。 より詳しい情報については、wicontrol(8), ifconfig(8) および wi(4) のマニュアルを読むとよいでしょう。

また、下記の暗号化に関する節を読むこともおすすめします。

ステータス情報

一度アクセスポイントが設定されて稼働すると、
管理者はアクセスポイントを利用しているクライアントを見たいと思うでしょう。
いつでも管理者は以下のコマンドを実行できます。

```
# wicontrol -l
1 station:
00:09:b7:7b:9d:16  asid=04c0, flags=3<ASSOC,AUTH>, caps=1<ESS>, rates
=f<1M,2M,5.5M,11M>, sig=38/15
```

これは一つの局が、 表示されているパラメータで接続していることを示します。 表示された信号は、 相対的な強さを表示しているだけのものとして扱われるべきです。 dBm やその他の単位への変換結果は、 異なるファームウェアバージョン間で異なります。

クライアント

無線クライアントはアクセスポイント、 または他のクライアントに直接アクセスするシステムです。

典型的には、 無線クライアントが有しているネットワークデバイスは、 無線ネットワークカード 1

枚だけです。

無線クライアントを設定するにはいくつか方法があります。
それぞれは異なる無線モードに依存していますが、一般的には BSS
(アクセスポイントを必要とするインフラストラクチャーモード) か、IBSS
(アドホック、またはピアツーピアモード) のどちらかです。
ここでは、アクセスポイントと通信をするのに、両者のうちで最も広まっている BSS
モードを使用します。

要件

FreeBSD を無線クライアントとして設定するのに、 本当に必要なものはたった 1 つだけです。 FreeBSD が対応している無線カードが必要です。

無線 **FreeBSD** クライアントの設定

設定をはじめる前に、
あなたが接続しようとする無線ネットワークについていくつか知っておかなければなりません。
この例では、*my_net* という名前で暗号化は無効になっているネットワークに接続しようとしています。

この例では暗号化を行っていないのですが、
これは危険な状況です。次の節で、暗号化を有効にする方法と、 なぜそれが重要で、
暗号技術によっては完全にはあなたを保護することができないのはなぜか、
ということを学ぶでしょう。

カードが FreeBSD に認識されていることを確認してください。

```
# ifconfig -a
wi0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        inet6 fe80::202:2dff:fe2d:c938%wi0 prefixlen 64 scopeid 0x7
        inet 0.0.0.0 netmask 0xff000000 broadcast 255.255.255.255
        ether 00:09:2d:2d:c9:50
        media: IEEE 802.11 Wireless Ethernet autoselect (DS/2Mbps)
        status: no carrier
        ssid ""
        stationname "FreeBSD Wireless node"
        channel 10 authmode OPEN powersavemode OFF powersavesleep 100
        wepmode OFF weptxkey 1
```

それでは、このカードをネットワークに合わせて設定しましょう。

```
# ifconfig wi0 inet 192.168.0.20 netmask 255.255.255.0 ssid my_net
```

192.168.0.20 と 255.255.255.0 を有線ネットワークで有効な IP
アドレスとネットマスクに置き換えてください。
アクセスポイントは無線ネットワークと有線ネットワークの間でデータをブリッジしているため、
ネットワーク上の他のデバイスには、このデバイスが、他と同様に、
有線ネットワーク上にあるかのように見えることに注意してください。

これを終えると、あなたは標準的な有線接続を使用しているかのように、

有線ネットワーク上のホストに ping を送ることができるでしょう。

無線接続に関する問題がある場合は、 アクセスポイントに接続されていることを確認してください。

```
# ifconfig wi0
```

いくらか情報が表示されるはずです。 その中に以下の表示があるはずです。

```
status: associated
```

もし associated と表示されなければ、 アクセスポイントの範囲外かもしれないし、
暗号化が有効になっているかもしれないし、 または設定の問題を抱えているのかもしれません。

暗号化

無線ネットワークを暗号化することが重要なのは、
十分保護された領域にネットワークを留める能力がもはやないからです。
無線データはその周辺全体にわたって放送されるので、
それを読みたいと思う人はだれでも読むことができます。 そこで暗号化が役に立ちます。
電波に載せて送られるデータを暗号化することによって、
興味を抱いた者が空中からデータを取得することをずっと難しくします。

クライアントとアクセスポイント間のデータを暗号化するもっとも一般的な方法には、 WEP と ipsec(4) の二種類があります。

WEP

WEP は Wired Equivalency Protocol（訳注: 直訳すると、有線等価プロトコル）の略語です。WEP は無線ネットワークを有線ネットワークと同程度に安全で確実なものにしようとする試みです。
残念ながら、これはすでに破られており、 破るのはそれほど苦労しません。
これは、機密データを暗号化するという場合に、 これに頼るものではないということも意味します。

なにも無いよりはましなので、 次のコマンドを使って、あなたの新しい FreeBSD アクセスポイント上で WEP を有効にしてください。

```
# ifconfig wi0 inet up ssid my_net wepmode on wepkey 0x1234567890 media DS/11Mbps
mediaopt hostap
```

クライアントについては次のコマンドで WEP を有効にできます。

```
# ifconfig wi0 inet 192.168.0.20 netmask 255.255.255.0 ssid my_net wepmode on wepkey
0x1234567890
```

0x1234567890 をより特異なキーに変更すべきであることに注意してください。

IPsec

ipsec(4) はネットワーク上で交わされるデータを暗号化するための、 はるかに頑健で強力なツールです。 これは無線ネットワーク上のデータを暗号化する明らかに好ましい方法です。 ハンドブック内の IPsec 節で ipsec(4) セキュリティ、 およびその実装方法の詳細を読むことができます。

ツール

無線ネットワークをデバッグしたり設定するのに使うツールがわずかばかりあります。
ここでその一部と、それらが何をしているか説明します。

bsd-airtools パッケージ

bsd-airtools パッケージは、 WEP キークラッキング、
アクセスポイント検知などの無線通信を監査するツールを含む完備されたツール集です。

bsd-airtools ユーティリティは net/bsd-airtools port からインストールできます。 ports のインストールに関する情報はこのハンドブックの アプリケーションのインストール - packages と ports を参照してください。

`dstumbler` プログラムは、 アクセスポイントの発見および S/N 比のグラフ化をできるようにするパッケージツールです。
アクセスポイントを立ち上げて動かすのに苦労しているなら、 `dstumbler` はうまく行く手助けになるかもしれません。

無線ネットワークの安全性をテストするのに、 "dweputils" (`dwepcrack`, `dwepdump` および `dwepkeygen`) を使用することで、 WEP があなたの無線安全性への要求に対する正しい解決策かどうか判断するのを助けられるかもしれません。

`wicontrol`, `ancontrol` および `raycontrol` ユーティリティ

これらは、無線ネットワーク上で無線カードがどのように動作するかを制御するツールです。
上記の例では、無線カードが wi0 インタフェースであるので、wicontrol(8) を使用することに決めました。 もし Cisco の無線デバイスを持っている場合は、それは an0 として動作するでしょうから、 ancontrol(8) を使うことになるでしょう。

`ifconfig` コマンド

ifconfig(8) は wicontrol(8) と同じオプションの多くを処理できますが、 いくつかのオプションを欠いています。 コマンドライン引数とオプションについて ifconfig(8) を参照してください。

対応しているカード

アクセスポイント

現在のところ (アクセスポイントとして) BSS モードに対応した唯一のカードは Prism 2, 2.5 または 3 チップセットを利用したデバイスです。 wi(4) に完全な一覧があります。

クライアント

現在、FreeBSD では、ほとんどすべての 802.11b 無線カードに対応しています。 Prism, Spectrum24, Hermes, Aironet または Raylink のチップセットを利用したほとんどのカードは、 (アドホック、ピアツーピア、そして BSS の) IBSS

モードで無線ネットワークカードとして動作するでしょう。

20.4. Bluetooth

20.4.1. はじめに

Bluetooth は免許のいらない 2.4 GHz の帯域を利用して、 10 m 程度のパーソナルネットワークを作る無線技術です。
ネットワークはたいていの場合、その場その場で、携帯電話や PDA やノートパソコンなどの携帯デバイスから形成されます。 Wi-Fi などの他の有名な無線技術とは違い、Bluetooth はより高いレベルのサービスを提供します。 たとえば、FTP のようなファイルサーバ、ファイルのプッシュ、
音声伝送、シリアル線のエミュレーションなどのサービスです。

FreeBSD 内での Bluetooth スタックは Netgraph フレームワーク (netgraph(4) 参照) を使って実現されています。 ng_ubt(4) ドライバは、 多種多様な Bluetooth USB ドングルに対応しています。 Broadcom BCM2033 チップを搭載した Bluetooth デバイスは ubtbcmfw(4) および ng_ubt(4) ドライバによって対応されています。 3Com Bluetooth PC カード 3CRWB60-A は ng_bt3c(4) ドライバによって対応されています。 シリアルおよび UART を搭載した Bluetooth デバイスは sio(4), ng_h4(4) および hcseriald(8) ドライバによって対応されています。
この節では USB Bluetooth ドングルの使用法について説明します。 Bluetooth に対応しているのは FreeBSD 5.0 以降のシステムです。

5.0, 5.1 Release ではカーネルモジュールは利用可能ですが、種々のユーティリティとマニュアルは標準でコンパイルされていません。

20.4.2. デバイスの挿入

デフォルトでは Bluetooth デバイスドライバはカーネルモジュールとして利用できます。デバイスを接続する前に、 カーネルにドライバを読み込む必要があります。

```
# kldload ng_ubt
```

Bluetooth デバイスがシステム起動時に存在している場合、 /boot/loader.conf からモジュールを読み込んでください。

```
ng_ubt_load="YES"
```

USB ドングルを挿してください。コンソールに (または syslog に) 下記のような表示が現れるでしょう。

```
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
```

/usr/shared/examples/netgraph/bluetooth/rc.bluetooth を /etc/rc.bluetooth

のようなどこか便利な場所にコピーしてください。 このスクリプトは Bluetooth スタックを開始および終了させるのに使われます。
デバイスを抜く前にスタックを終了するのはよい考えですが、 (たいていの場合) しなくても致命的ではありません。 スタックを開始するときに、下記のような出力がされます。

```
# /etc/rc.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
```

20.4.3. ホストコントローラインタフェース (HCI)

ホストコントローラインタフェース (HCI) は、ベースバンドコントローラおよびリンクマネージャへのコマンドインタフェースを提供し、ハードウェアステータスおよびコントロールレジスタへアクセスします。 このインタフェースは Bluetooth ベースバンド機能へアクセスする画一的な方法を提供します。 ホストの HCI 層は Bluetooth ハードウェア上の HCI ファームウェアと、 データとコマンドをやり取りします。 ホストコントローラトランスポート層 (つまり物理的なバス) のドライバは、 両方の HCI 層に相互に情報を交換する能力を与えます。

一つの Bluetooth デバイスにつき、*hci* タイプの Netgraph ノードが一つ作成されます。 HCI ノードは通常 Bluetooth デバイスドライバノード (下流) と L2CAP ノード (上流) に接続されます。 すべての HCI 動作はデバイスドライバノード上ではなく、 HCI ノード上で行われなくてはいけません。 HCI ノードのデフォルト名は "devicehci" です。 詳細については ng_hci(4) マニュアルを参照してください。

最も一般的なタスクの一つに、無線通信的に近傍にある Bluetooth デバイスの発見があります。 この動作は *inquiry* (問い合わせ) と呼ばれています。 Inquiry や他の HCI に関連した動作は hccontrol(8) ユーティリティによってなされます。 下記の例は、どの Bluetooth デバイスが通信圏内にあるかを知る方法を示しています。
デバイスのリストが表示されるには数秒かかります。 リモートデバイスは *discoverable* (発見可能な) モードにある場合にのみ inquiry に返答するということに注意してください。

```
% 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 は Bluetooth デバイスに固有のアドレスです。 これはネットワークカードの MAC アドレスに似ています。 このアドレスはデバイスとの通信を続けるのに必要となります。 BD_ADDR に人間が判読しやすい名前を割り当てることもできます。 /etc/bluetooth/hosts ファイルには、 既知の Bluetooth ホストに関する情報が含まれています。 次の例はリモートデバイスに割り当てられている、人間が判読しやすい名前を得る方法を示しています。

```
% hccontrol -n ubt0hci remote_name_request 00:80:37:29:19:a4
BD_ADDR: 00:80:37:29:19:a4
Name: Pav's T39
```

リモートの Bluetooth デバイス上で inquiry を実行すると、 あなたのコンピュータは "your.host.name (ubt0)" と認識されます。 ローカルデバイスに割り当てられた名前はいつでも変更できます。

Bluetooth システムは一対一接続 (二つの Bluetooth ユニットだけが関係します) または一対多接続を提供します。 一対多接続では、接続はいくつかの Bluetooth デバイス間で共有されます。
次の例は、ローカルデバイスに対するアクティブなベースバンド接続のリストを得る方法を示しています。

```
% 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
```

connection handle はベースバンド接続の終了が必要とされるときに便利です。 もっとも、通常はこれを手動で行う必要はありません。 Bluetooth スタックはアクティブでないベースバンド接続を自動的に終了します。

```
# hccontrol -n ubt0hci disconnect 41
Connection handle: 41
Reason: Connection terminated by local host [0x16]
```

利用可能な HCI コマンドの完全な一覧を得るには、 hccontrol help を参照してください。 HCI コマンドのほとんどはスーパユーザ権限を必要としません。

20.4.4. ロジカルリンクコントロールおよびアダプテーションプロトコル (L2CAP)

ロジカルリンクコントロールおよびアダプテーションプロトコル (L2CAP) は、プロトコル多重化ケーパビリティおよび分割・再編成動作を備えた、
上位層プロトコルへのコネクション指向およびコネクションレスデータサービスを提供します。 L2CAP は上位層プロトコルおよびアプリケーションが 64 KB までの長さの L2CAP データパケットを送受信することを可能にします。

L2CAP は チャネル の概念に基づいています。

チャネルはベースバンド接続の上位に位置する論理的な接続です。
それぞれのチャネルは多対一の方法で一つのプロトコルに結びつけられます。
複数のチャネルを同じプロトコルに結びつけることは可能ですが、
一つのチャネルを複数のプロトコルに結びつけることはできません。
チャネル上で受け取られたそれぞれの L2CAP パケットは、 適切なより上位のプロトコルに渡されます。
複数のチャネルは同じベースバンド接続を共有できます。

一つの Bluetooth デバイスに対して、*l2cap* タイプの Netgraph ノードが一つ作成されます。 L2CAP ノードは通常 Bluetooth HCI ノード (下流) と Bluetooth ソケットノード (上流) に接続されます。 L2CAP ノードのデフォルト名は "devicel2cap" です。 詳細については ng_l2cap(4) マニュアルを参照してください。

便利なコマンドに、他のデバイスに ping を送ることができる l2ping(8) があります。Bluetooth 実装によっては、 送られたデータすべては返さないことがあります。 したがって次の例で *0* バイトは正常です。

```
# 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
```

l2control(8) ユーティリティは L2CAP ノード上でさまざまな操作を行うのに使われます。 この例は、ローカルデバイスに対する論理的な接続 (チャネル) およびベースバンド接続の一覧を得る方法を示しています。

```
% 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
```

別の診断ツールが btsockstat(1) です。 これは netstat(1) と同様の作業を、Bluetooth ネットワークに関するデータ構造についての行います。 下記の例は上の l2control(8) と同じ論理的な接続を示します。

```
% 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
```

20.4.5. RFCOMM プロトコル

RFCOMM プロトコルは L2CAP プロトコルを介してシリアルポートのエミュレーションを提供します。このプロトコルは ETSI (訳注: 欧州電気通信標準化機構) 標準 TS 07.10 に基づいています。 RFCOMM プロトコルは、単純な伝送プロトコルに RS-232 (EIATIA-232-E) シリアルポートの 9 本の結線をエミュレートする項目を加えたものです。 RFCOMM プロトコルは、二つの Bluetooth デバイス間で、最大 60 までの同時接続 (RFCOMM チャネル) に対応しています。

RFCOMM の目的から、完全な通信経路は、異なるデバイス上 (通信の端点) で動作している二つのアプリケーションと、 その間の通信セグメントを含んでいます。RFCOMM は、それが動いているデバイスのシリアルポートを利用するアプリケーションをカバーするためのものです。 通信セグメントはあるデバイスから他のデバイスへの Bluetooth リンクです (直接接続)。

RFCOMM は直接接続している場合のデバイス間の接続、またはネットワークの場合のデバイスとモデムの間の接続にだけ関係があります。 RFCOMM は、一方が Bluetooth 無線技術で通信し、 もう一方で有線インタフェースを提供するモジュールのような、他の構成にも対応できます。

FreeBSD では RFCOMM プロトコルは Bluetooth ソケット層に実装されています。

20.4.6. デバイスのペアリング

デフォルトでは Bluetooth 通信は認証されておらず、すべてのデバイスが他のすべてのデバイスと通信できます。 Bluetooth デバイス (たとえば携帯電話) は特定のサービス (たとえばダイアルアップサービス) を提供するために、認証を要求することも選択できます。 Bluetooth 認証は通常 *PIN* コード で行われます。 PIN コードは最長 16 文字のアスキー文字列です。 ユーザは両デバイスで同じ PIN コードを入力することを要求されます。 一度 PIN コードを入力すると、 両デバイスは リンクキーを作成します。 その後、リンクキーはそのデバイス自身または、不揮発性記憶デバイス内に格納できます。
次の機会には、両デバイスは前に作成されたリンクキーを使用するでしょう。
このような手続きをペアリング *(pairing)* と呼びます。いずれかのデバイス上でリンクキーが失われたときには、ペアリングをやり直さなければならないことに注意してください。

hcsecd(8) デーモンが Bluetooth 認証要求のすべてを扱う責任を負っています。デフォルトの設定ファイルは /etc/bluetooth/hcsecd.conf です。 PIN コードが "1234" に設定された携帯電話に関する例は以下の通りです。

```
device {
        bdaddr  00:80:37:29:19:a4;
        name    "Pav's T39";
        key     nokey;
        pin     "1234";
      }
```

PIN コードには (長さを除いて) 制限はありません。 いくつかのデバイス (たとえば Bluetooth

ヘッドフォン) には固定的な PIN コードが組み込まれているかもしれません。 -d オプションは hcsecd(8) デーモンがフォアグラウンドで動作するように強制するため、
何が起きているのか確認しやすくなります。 リモートデバイスがペアリングを受け取るように設定して、リモートデバイスへの Bluetooth 接続を開始してください。
リモートデバイスはペアリングが受け入れらた、と応答して PIN コードを要求するでしょう。
hcsecd.conf 内にあるのと同じ PIN コードを入力してください。 これであなたの PC とリモートデバイスがペアとなりました。
また、リモートデバイスからペアリングを開始することもできます。 以下は hcsecd の出力例です。

```
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
```

20.4.7. サービスディスカバリプロトコル (SDP)

サービスディスカバリプロトコル (SDP) は、クライアントアプリケーションが、サーバアプリケーションが提供するサービスの存在とその属性を発見する手段を提供します。
サービスの属性には提示されているサービスのタイプまたはクラス、
および、サービスを利用するのに必要な仕組みまたはプロトコルの情報が含まれます。

SDP には SDP サーバと SDP クライアント間の通信が含まれます。 SDP サーバは、サーバに関連づけられたサービスの特性について記述しているサービスレコードの一覧を維持しています。 各サービスレコードにはそれぞれ 1 つのサービスの情報が書かれています。
クライアントは SDP リクエストを出すことによって、SDP サーバが維持しているサービスレコードから情報を検索できます。
クライアントまたはクライアントに関連づけられたアプリケーションがサービスを利用することにしたら、 サービスを利用するためには、 サービスプロバイダへの接続を別途開かなければなりません。 SDP はサービスとそれらの属性を発見するための仕組みを提供しますが、
そのサービスを利用するための仕組みは提供しません。

通常 SDP クライアントは希望するサービスの特性に基づいてサービスを検索します。しかしながら、サービスに関する事前の情報なしに、どのタイプのサービスが SDP サーバのサービスレコードに記述されているか知ることが望ましいことがあります。
この、提供されている任意のサービスを閲覧する手順を、 ブラウジング *(browsing)* と呼びます。

現在のところ Bluetooth SDP サーバおよびクライアントは、ここからダウンロードできる第三者パッケージ sdp-1.5 で実装されています。 sdptool はコマンドラインの SDP クライアントです。 次の例は SDP ブラウズの問い合わせ方法を示しています。

```
# sdptool browse 00:80:37:29:19:a4
Browsing 00:80:37:29:19:A4 ...
```

```
Service Name: Dial-up Networking
Protocol Descriptor List:
 "L2CAP" (0x0100)
 "RFCOMM" (0x0003)
   Channel: 1

Service Name: Fax
Protocol Descriptor List:
 "L2CAP" (0x0100)
 "RFCOMM" (0x0003)
   Channel: 2

Service Name: Voice gateway
Service Class ID List:
 "Headset Audio Gateway" (0x1112)
 "Generic Audio" (0x1203)
Protocol Descriptor List:
 "L2CAP" (0x0100)
 "RFCOMM" (0x0003)
   Channel: 3
```

i. 等々。 それぞれのサービスは属性の一覧 (たとえば RFCOMM チャネル) を持っていることに注意してください。サービスによっては、 属性のリストの一部についてメモをとっておく必要があるかもしれません。 Bluetooth 実装のいくつかは、サービスブラウジングに対応しておらず、 空の一覧を返してくるかもしれません。この場合、 特定のサービスを検索をすることは可能です。下記の例は OBEX オブジェクトプッシュ (OPUSH) サービスを検索する方法です。

```
# sdptool search --bdaddr 00:07:e0:00:0b:ca OPUSH
```

FreeBSD 上における Bluetooth クライアントへのサービス提供は sdpd サーバが行います。

```
# sdpd
```

sdptool は、ローカル SDP サーバにサービスを登録するのにも用いられます。 下記の例は PPP (LAN) サービスを備えたネットワークアクセスを登録する方法を示しています。 一部のサービスでは属性 (たとえば RFCOMM チャネル) を要求することに注意してください。

```
# sdptool add --channel=7 LAN
```

ローカル SDP サーバに登録されたサービスの一覧は SDP ブラウザの問い合わせを "特別な" BD_ADDR に送ることで得られます。

```
# sdptool browse ff:ff:ff:00:00:00
```

20.4.8. ダイアルアップネットワーク (DUN) および PPP (LAN) を用いたネットワークアクセスプロファイル

ダイアルアップネットワーク (DUN) プロファイルはほとんどの場合、モデムや携帯電話とともに使用されます。 このプロファイルが対象とする場面は以下のものです。

- コンピュータから携帯電話またはモデムを、
  ダイアルアップインターネットアクセスサーバへの接続、
  または他のダイアルアップサービスを利用するための無線モデムとして使うこと
- データ呼び出しを受けるための、 コンピュータによる携帯電話またはモデムの使用

PPP (LAN) によるネットワークアクセスプロファイルは、 次の状況で利用できます。

- 単一の Bluetooth デバイスへの LAN アクセス
- マルチ Bluetooth デバイスへの LAN アクセス
- (シリアルケーブルエミュレーション上の PPP ネットワーク接続を使用した) PC から PC への接続

FreeBSD ではどちらのプロファイルも ppp(8) と rfcomm_pppd(8) (RFCOMM Bluetooth 接続を PPP が制御可能なように変換するラッパ) で実装されています。 いずれかのプロファイルが使用可能となる前に、 /etc/ppp/ppp.conf 内に新しい PPP ラベルが作成されていなければなりません。 例については、 rfcomm_pppd(8) のマニュアルページを参照してください。

次の例では、DUN RFCOMM チャネル上で BD_ADDR が 00:80:37:29:19:a4 のリモートデバイスへの RFCOMM 接続を開くのに rfcomm_pppd(8) が使われます。実際の RFCOMM チャネル番号は SDP を介してリモートデバイスから得ます。 手動で RFCOMM チャネルを指定することもでき、その場合 rfcomm_pppd(8) は SDP 問い合わせを実行しません。 リモートデバイス上の RFCOMM チャネルを見つけるには、 sdptool を使ってください。

```
# rfcomm_pppd -a 00:80:37:29:19:a4 -c -C dun -l rfcomm-dialup
```

PPP (LAN) サービスでネットワークアクセスを提供するためには、 sdpd サーバが動いていなければなりません。 これはローカル SDP サーバに LAN サービスを登録するのにも必要です。 LAN サービスは RFCOMM チャネル属性を必要とすることに注意してください。 /etc/ppp/ppp.conf ファイル内に LAN クライアントの新しいエントリを作成しなければなりません。 例については rfcomm_pppd(8) のマニュアルページを参照してください。 最後に、RFCOMM PPP サーバが実行され、 ローカル SDP サーバに登録されているのと同じ RFCOMM チャネルで待ち受けていなければなりません。 次の例は RFCOMM PPP サーバを起動する方法を示しています。

```
# rfcomm_pppd -s -C 7 -l rfcomm-server
```

20.4.9. OBEX プッシュ (OPUSH) プロファイル

OBEX はモバイルデバイス間で広く使われている単純なファイル転送プロトコルです。 これは主に赤外線通信で利用されており、ノートパソコンや PDA 間の汎用的なファイル転送、および PIM

アプリケーションを搭載した携帯電話その他のデバイス間で名刺やカレンダーエントリを転送するのに用いられます。

OBEX サーバおよびクライアントは、ここ からダウンロードできる obexapp-1.0 という第三者のパッケージとして実装されています。 このパッケージは openobex ライブラリ (上記の obexapp に含まれます) および devel/glib12 port を必要とします。 なお、obexapp はルート権限を必要としません。

OBEX クライアントは OBEX サーバとの間でオブジェクトを渡したり (プッシュ) および受け取ったり (プル) するのに使用されます。 オブジェクトは、たとえば名刺や予定などになります。 OBEX クライアントは RFCOMM チャネル番号を SDP によってリモートデバイスから得ることができます。 これは RFCOMM チャネル番号の代わりにサービス名を指定することによって行うことができます。 対応しているサービス名は IrMC, FTRN および OPUSH です。 RFCOMM チャネルを番号で指定することもできます。
下記は、デバイス情報オブジェクトを携帯電話から受け取り、 新しいオブジェクト (名刺) が携帯電話に渡される場合の OBEX セッションの例です。

```
% obexapp -a 00:80:37:29:19:a4 -C IrMC
obex> get
get: remote file> telecom/devinfo.txt
get: local file> devinfo-t39.txt
Success, response: OK, Success (0x20)
obex> put
put: local file> new.vcf
put: remote file> new.vcf
Success, response: OK, Success (0x20)
obex> di
Success, response: OK, Success (0x20)
```

OBEX プッシュサービスを提供するためには、 sdpd サーバが実行されていなければなりません。 また OPUSH サービスをローカル SDP サーバに登録することも必要です。 なお、OPUSH サービスには RFCOMM チャネル属性が必要です。
渡されるオブジェクトをすべて格納するルートフォルダを作成しなければいけません。
ルートフォルダのデフォルトパスは /var/spool/obex です。 最後に OBEX サーバが実行され、 ローカル SDP サーバに登録されているのと同じ RFCOMM チャネルで待ち受けていなければなりません。
下記の例は OBEX サーバの起動方法を示します。

```
# obexapp -s -C 10
```

20.4.10. シリアルポート (SP) プロファイル

シリアルポート (SP) プロファイルは Bluetooth デバイスが RS232 (または同様の) シリアルケーブルエミュレーションを行えるようにします。 このプロファイルが対象とする場面は、レガシーアプリケーションが、仮想シリアルポート抽象を介して Bluetooth をケーブルの代替品として使うところです。

rfcomm_sppd(1) ユーティリティはシリアルポートプロファイルを実装します。 Pseudo tty が仮想シリアルポート抽象概念として用いられます。

下記の例はリモートデバイスのシリアルポートサービスへ接続する方法を示します。 なお、RFCOMM チャネルを指定する必要はありません。- rfcomm_sppd(1) は SDP を介してリモートデバイスからその情報を得ることができます。
これを上書きしたい場合にはコマンドラインで RFCOMM チャネルを指定してください。

```
# rfcomm_sppd -a 00:07:E0:00:0B:CA -t /dev/ttyp6
rfcomm_sppd[94692]: Starting on /dev/ttyp6...
```

接続された pseudo tty はシリアルポートとして利用することができます。

```
# cu -l ttyp6
```

20.4.11. トラブルシューティング

リモートデバイスが接続できません

古い Bluetooth デバイスのなかにはロールスイッチング (role switching) に対応していないものがあります。 デフォルトでは FreeBSD が新しい接続を受け付けるときに、ロールスイッチを実行してマスタになろうとします。
これに対応していないデバイスは接続できないでしょう。
なお、ロールスイッチングは新しい接続が確立されるときに実行されるので、
ロールスイッチングに対応しているかどうかリモートデバイスに問い合わせることはできません。
ローカル側でロールスイッチングを無効にする HCI オプションがあります。

```
# hccontrol -n ubt0hci write_node_role_switch 0
```

何かがうまくいっていないみたいです。 何が実際に起こっているか確認できますか?

できます。 ここ からダウンロードできる第三者パッケージ hcidump-1.5 を使ってください。 hcidump ユーティリティは tcpdump(1) と似ています。 これはターミナル上の Bluetooth パケットの内容の表示および Bluetooth パケットをファイルにダンプするのに使えます。

20.5. ブリッジ

20.5.1. はじめに

IP サブネットを作成して、 それらのセグメントをルータを使って接続することなしに、 (Ethernet セグメントのような)
一つの物理ネットワークを二つのネットワークセグメントに分割することはとても有効な場合があります。 この方法で二つのネットワークを繋ぐデバイスは "ブリッジ" と呼ばれます。
二つのネットワークインタフェースカードを持つ FreeBSD システムは、ブリッジとして動作することができます。

ブリッジは、各ネットワークインタフェイスに繋がるデバイスの MAC 層のアドレス (Ethernet アドレス) を記憶することにより動作します。
ブリッジはトラフィックの送信元と受信先が異なったネットワーク上にある場合にのみトラフィックを転

送します。

多くの点で、ブリッジはポート数の少ない Ethernet スイッチのようなものといえます。

20.5.2. ブリッジがふさわしい状況

今日ブリッジが活躍する場面は大きく分けて二つあります。

トラフィックの激しいセグメント

ひとつは、物理ネットワークセグメントがトラフィック過剰になっているが、なんらかの理由によりネットワークをサブネットに分け、ルータで接続することができない場合です。

編集部門と製作部門がおなじサブネットに同居している新聞社を例に考えてみましょう。編集部門のユーザはファイルサーバとして全員サーバ A を利用し、製作部門のユーザはサーバ B を利用します。すべてのユーザを接続するのには Ethernet が使われており、高負荷となったネットワークは遅くなってしまいます。

もし編集部門のユーザを一つのネットワークセグメントに分離することができ、製作部門のユーザも同様にできるのなら、二つのネットワークセグメントをブリッジで繋ぐことができます。ブリッジの "反対" 側へ向かうネットワークトラフィックだけが転送され、各ネットワークセグメントの混雑は緩和されます。

パケットフィルタ/帯域制御用ファイアウォール

もうひとつはネットワークアドレス変換 (NAT) を使わずにファイアウォール機能を利用したい場合です。

ここでは DSL もしくは ISDN で ISP に接続している小さな会社を例にとってみましょう。この会社は ISP からグローバル IP アドレスを 13 個割り当てられており、ネットワーク上には 10 台の PC が存在します。このような状況では、サブネット化にまつわる問題から、ルータを用いたファイアウォールを利用することは困難です。

ブリッジを用いたファイアウォールなら、IP アドレスの問題を気にすること無く、DSL/ISDN ルータの下流側に置くように設定できます。

20.5.3. ブリッジを設定する

ネットワークインタフェースカードの選択

ブリッジを利用するには少なくとも 2 枚のネットワークカードが必要です。残念なことに FreeBSD 4.0 ではすべてのネットワークインタフェースカードがブリッジ機能に対応しているわけではありません。カードに対応しているかどうかについては bridge(4) を参照してください。

以下に進む前に、二枚のネットワークカードをインストールしてテストしてください。

カーネルコンフィグレーションの変更

カーネルでブリッジ機能を有効にするには

```
options BRIDGE
```

という行をカーネルコンフィグレーションファイルに追加して カーネルを再構築してください。

ファイアウォール対応

ファイアウォールとしてブリッジを利用しようとしている場合には IPFIREWALL オプションも指定する必要があります。
ブリッジをファイアウォールとして設定する際の一般的な情報に関しては、 ファイアウォールの章を参照してください。

IP 以外のパケット (ARP など) がブリッジを通過するようにするためには、ファイアウォール用オプションを設定しなければなりません。 このオプションは IPFIREWALL_DEFAULT_TO_ACCEPT です。この変更により、デフォルトではファイアウォールがすべてのパケットを受け入れるようになることに注意してください。
この設定を行う前に、
この変更が自分のルールセットにどのような影響をおよぼすかを把握しておかなければなりません。

帯域制御機能

ブリッジで帯域制御機能を利用したい場合、 カーネルコンフィグレーションで DUMMYNET オプションを加える必要があります。 詳しい情報に関しては dummynet(4) を参照してください。

20.5.4. ブリッジを有効にする

ブリッジを有効にするには、 /etc/sysctl.conf に以下の行を加えてください。

```
net.link.ether.bridge=1
```

指定したインタフェースでブリッジを可能にするには以下を加えてください。

```
net.link.ether.bridge_cfg=if1,if2
```

(*if1* および *if2* は二つのネットワークインタフェースの名前に置き換えてください)。
ブリッジを経由したパケットを ipfw(8) でフィルタしたい場合には、以下の行も付け加える必要があります

```
net.link.ether.bridge_ipfw=1
```

FreeBSD 5.2-RELEASE 以降では、かわりに以下の行を使用してください。

```
net.link.ether.bridge.enable=1
 net.link.ether.bridge.config=if1,if2
 net.link.ether.bridge.ipfw=1
```

20.5.5. その他の情報

ネットワークからブリッジに telnet(1) したい場合、 ネットワークカードの一つに IP アドレスを割り当てるのが正しいです。 一般的に、両方のカードに IP

アドレスを割り当てるのはよい考えではないとされています。

ネットワーク内に複数のブリッジを設置する場合、
任意のワークステーション間で一つ以上の経路を持つことはできません。 技術的には、
これはスパニングツリーのリンク制御はサポートされていない、 ということを意味します。

ブリッジは、ping(8) にかかる時間を遅らせることがあります。特に、
一方のセグメントからもう一方へのトラフィックでそうなります。

20.6. NFS

FreeBSD がサポートしている多くのファイルシステムの中には、 NFS
とも呼ばれているネットワークファイルシステムがあります。 NFS はあるマシンから他のマシンへと、
ネットワークを通じてディレクトリとファイルを共有することを可能にします。 NFS を使うことで、
ユーザやプログラムはリモートシステムのファイルを、
それがローカルファイルであるかのようにアクセスすることができます。

NFS が提供可能な最も特筆すべき利点いくつかは以下のものです。

- 一般的に使われるデータを単一のマシンに納めることができ、
  ユーザはネットワークを通じてデータにアクセスできるため、
  ローカルワークステーションが使用するディスク容量が減ります。
- ネットワーク上のすべてのマシンに、 ユーザが別々にホームディレクトリを持つ必要がありません。
  NFS サーバ上にホームディレクトリが設定されれば、
  ネットワークのどこからでもアクセス可能です。
- フロッピーディスクや CDROM ドライブ、 ZIP ドライブなどのストレージデバイスを、
  ネットワーク上の他のマシンで利用することができます。
  ネットワーク全体のリムーバブルドライブの数を減らせるかもしれません。

20.6.1. NFS はどのように動作するのか

NFS は最低二つの主要な部分、 サーバと一つ以上のクライアントからなります。
クライアントはサーバマシン上に格納されたデータにリモートからアクセスします。
これが適切に機能するには、 いくつかのプロセスが設定されて実行されていなければなりません。

FreeBSD 5.X では portmap ユーティリティは rpcbind
ユーティリティに置き換わりました。 したがって FreeBSD 5.X
では、ユーザは下記の例で、 portmap の例のすべてを rpcbind
に置き換える必要があります。

サーバは以下のデーモンを動作させなければなりません。

デーモン	説明
nfsd	NFS クライアントからのリクエストを処理する NFS デーモン
mountd	nfsd(8) から渡されたリクエストを実際に実行する NFS マウントデーモン

デーモン	説明
portmap	NFS サーバの利用しているポートを NFS クライアントから取得できるようにするためのポートマッパデーモン

クライアント側では nfsiod というデーモンも実行できます。 nfsiod デーモンは NFS サーバからのリクエストを処理します。 これは任意であり、性能を改善しますが、通常の正しい動作には必要としません。詳細については nfsiod(8) マニュアルページを参照してください。

20.6.2. NFS の設定

NFS の設定は比較的素直な工程です。 動かさなければならないプロセスは /etc/rc.conf ファイルを少し変更すれば起動時に実行させられます。

NFS サーバでは /etc/rc.conf ファイルの中で、以下のオプションが設定されていることを確かめてください。

```
portmap_enable="YES"
nfs_server_enable="YES"
mountd_flags="-r"
```

`mountd` は NFS サーバが有効になっていれば、 自動的に実行されます。

クライアント側では /etc/rc.conf 内に以下の設定があることを確認してください。

```
nfs_client_enable="YES"
```

/etc/exports ファイルは NFS サーバがどのファイルシステムをエクスポート (ときどき "共有" と呼ばれます) するのかを指定します。 /etc/exports ファイル中の各行は、エクスポートするファイルシステム、
およびそのファイルシステムにアクセスできるマシンを指定します。
ファイルシステムにアクセスできるマシンとともに、 アクセスオプションも指定できます。
このファイルで指定できるオプションはたくさんありますが、
ここではほんの少しだけ言及します。exports(5) マニュアルページを読めば、他のオプションは簡単にみつけられるでしょう。

いくつか /etc/exports の設定例を示します。

以下の例はファイルシステムのエクスポートの考え方を示しますが、
あなたの環境とネットワーク設定に応じて設定は少し変わるでしょう。 たとえば次の行は /cdrom ディレクトリを、サーバと同じドメイン名か (そのため、いずれもドメイン名がありません)、 /etc/hosts に記述されている三つの例となるマシンに対してエクスポートします。 `-ro` フラグは共有されるファイルシステムを読み込み専用にします。 このフラグにより、リモートシステムは共有されたファイルシステムに対して何の変更も行えなくなります。

```
/cdrom -ro host1 host2 host3
```

以下の設定は IP アドレスで指定した 3 つのホストに対して /home をエクスポートします。
この設定はプライベートネットワークで DNS が設定されていない場合に便利でしょう。
内部のホスト名に対して /etc/hosts を設定するという手段もあります。 詳細については hosts(5)
を参照してください。 -alldirs フラグはサブディレクトリがマウントポイントとなることを認めます。
言い替えると、これはサブディレクトリをマウントしませんが、 クライアントが要求するか、
または必要とするディレクトリだけをマウントできるようにします。

```
/home  -alldirs  10.0.0.2 10.0.0.3 10.0.0.4
```

以下の設定は、サーバとは異なるドメイン名の 2 台のクライアントがアクセスできるように /a
をエクスポートします。 -maproot=root フラグは、リモートシステムの root ユーザが、
エクスポートされたファイルシステムに root として書き込むことを許可します。 -maproot=root
フラグが無ければ、 リモートマシンの root 権限を持っていても、
共有されたファイルシステム上のファイルを変更することはできないでしょう。

```
/a  -maproot=root  host.example.com box.example.org
```

クライアントがエクスポートされたファイルシステムにアクセスするためには、
そうする権限が与えられていなければなりません。 /etc/exports ファイルに
クライアントが含まれているかどうか確認してください。

/etc/exports ファイルでは、
それぞれの行が一つのファイルシステムを一つのホストにエクスポートすることを表します。
リモートホストはファイルシステム毎に一度だけ指定することができ、
それに加えて一つのデフォルトエントリを置けます。たとえば /usr
が単一のファイルシステムであると仮定します。 次の /etc/exports は無効です。

```
/usr/src   client
/usr/ports client
```

単一のファイルシステムである /usr は、2 行に渡って、同じホスト client
へエクスポートされています。 この場合、正しい書式は次のとおりです。

```
/usr/src /usr/ports  client
```

あるホストにエクスポートされるある 1 つのファイルシステムのプロパティは、 1
行ですべて指定しなければなりません。
クライアントの指定のない行は、単一のホストとして扱われます。
これはファイルシステムをエクスポートできる方法を制限しますが、
多くの場合これは問題になりません。

下記は、 /usr および /exports がローカルファイルシステムである場合の、
有効なエクスポートリストの例です。

```
# 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
```

変更が有効となるように、 /etc/exports が変更されたら mountd を再起動しなければなりません。 これは mountd プロセスに HUP シグナルを送ることで実行できます。

```
# kill -HUP `cat /var/run/mountd.pid`
```

他には、再起動すれば、FreeBSD はすべてを適切に設定します。
しかしながら、再起動は必須ではありません。 root
権限で以下のコマンドを実行すれば、すべてが起動するでしょう。

NFS サーバでは

```
# portmap
# nfsd -u -t -n 4
# mountd -r
```

NFS クライアントでは

```
# nfsiod -n 4
```

これでリモートのファイルシステムを実際にマウントする準備がすべてできました。
この例では、サーバの名前は server で、 クライアントの名前は client とします。
リモートファイルシステムを一時的にマウントするだけ、
もしくは設定をテストするだけなら、クライアント上で root
権限で以下のコマンドを実行するだけです。

```
# mount server:/home /mnt
```

これで、サーバの /home ディレクトリが、クライアントの /mnt
にマウントされます。もしすべてが正しく設定されていれば、 クライアントの /mnt に入り、
サーバにあるファイルすべてを見れるはずです。

リモートファイルシステムを起動のたびに自動的にマウントしたいなら、 ファイルシステムを /etc/fstab
ファイルに追加してください。 例としてはこのようになります。

```
server:/home    /mnt    nfs rw  0   0
```

fstab(5) マニュアルページに利用可能なオプションがすべて掲載されています。

20.6.3. 実用的な使い方

NFS には実用的な使用法がいくつもあります。 ここで典型的な使用法をいくつか紹介しましょう。

- 何台ものマシンで CDROM などのメディアを共有するように設定します。
  これは安上がりで、たいていは、
  複数のマシンにソフトウェアをインストールするのにより便利な方法です。
- 大規模なネットワークでは、 すべてのユーザのホームディレクトリを格納するメイン NFS サーバを構築すると、ずっと便利でしょう。 どのワークステーションにログインしても、
  ユーザがいつでも同じホームディレクトリを利用できるように、
  これらのホームディレクトリはネットワークに向けてエクスポートされます。
- 何台ものマシンで /usr/ports/distfiles ディレクトリを共有できます。こうすると、
  何台ものマシン上に port をインストールする必要がある時に、
  それぞれのマシンでソースコードをダウンロードすることなく、 直ちにソースにアクセスできます。

20.6.4. amd による自動マウント

amd(8) (自動マウントデーモン) は、
ファイルシステム内のファイルまたはディレクトリがアクセスされると、
自動的にリモートファイルシステムをマウントします。
また、一定の間アクセスされないファイルシステムは amd によって自動的にアンマウントされます。
amd を使用することは、通常 /etc/fstab 内に記述する恒久的なマウントに対する、
単純な代替案となります。

amd はそれ自身を NFS サーバとして /host および /net ディレクトリに結びつけることによって動作します。
このディレクトリ内のどこかでファイルがアクセスされると、 amd は対応するリモートマウントを調べて、 自動的にそれをマウントします。 /net が、エクスポートされたファイルシステムを IP アドレスで指定してマウントするのに利用される一方で、 /host
は、エクスポートされたファイルシステムをリモートホスト名で指定してマウントするのに利用されます。

/host/foobar/usr 内のファイルにアクセスすると、 amd はホスト `foobar` からエクスポートされた /usr をマウントします。

例 *44. amd* によるエクスポートされたファイルシステムのマウント

`showmount` コマンドを用いて、 リモートホストのマウントで利用できるものが見られます。
たとえば、`foobar` と名付けられたホストのマウントを見るために次のように利用できます。

```
% showmount -e foobar
Exports list on foobar:
/usr                               10.10.10.0
/a                                 10.10.10.0
% cd /host/foobar/usr
```

例のように `showmount` はエクスポートとして /usr を表示します。 /host/foobar/usr

にディレクトリを変更すると、 amd はホスト名 foobar を解決し、お望みのエクスポートをマウントしようと試みます。

amd は /etc/rc.conf 内に次の行を記述すれば、 起動スクリプトによって起動されます。

```
amd_enable="YES"
```

さらに amd_flags オプションによって amd にフラグをカスタマイズして渡せます。デフォルトでは amd_flags は次のように設定されています。

```
amd_flags="-a /.amd_mnt -l syslog /host /etc/amd.map /net /etc/amd.map"
```

/etc/amd.map ファイルは、 エクスポートがマウントされるデフォルトオプションを決定します。 /etc/amd.conf ファイルは、 amd のより高度な機能の一部を設定します。

詳細については amd(8) および amd.conf(8) マニュアルページを参照してください。

20.6.5. 他のシステムとの統合についての問題

ISA バス用のイーサネットアダプタの中には性能が悪いため、 ネットワーク、特に NFS で深刻な問題がおきるものがあります。 これは FreeBSD に限ったことではありませんが FreeBSD でも起こり得ます。

この問題は (FreeBSD を使用した) PC がシリコングラフィックス社やサン・マイクロシステムズ社などの高性能なワークステーションにネットワーク接続されている場合に頻繁に起こります。 NFS マウントはうまく動作するでしょう。 また、いくつかの操作もうまく動作するかもしれませんが、 他のシステムに対する要求や応答は続いていても、 突然サーバがクライアントの要求に対して応答しなくなります。これは、 クライアントが FreeBSD か上記のワークステーションであるときにクライアント側に起きる現象です。 多くのシステムでは、いったんこの問題が現われると、 行儀良くクライアントを終了する手段はありません。 NFS がこの状態に陥ってしまうと正常に戻すことはできないため、 多くの場合クライアントをリセットすることが唯一の解決法となります。

"正しい" 解決法は、より高性能のイーサネットアダプタを FreeBSD システムにインストールすることですが、 満足に動作させる簡単な方法があります。 FreeBSD システムが サーバ になるのなら、 クライアントからのマウント時に -w=1024 オプションをつけて下さい。FreeBSD システムが クライアント になるのなら、 NFS ファイルシステムを -r=1024 オプションつきでマウントして下さい。 これらのオプションは自動的にマウントをおこなう場合には クライアントの fstab エントリの 4 番目のフィールドに指定してもよいですし、 手動マウントの場合は mount コマンドの -o パラメータで指定してもよいでしょう。

NFS サーバとクライアントが別々のネットワーク上にあるような場合、 これと間違えやすい他の問題が起きることに注意して下さい。 そのような場合は、ルータが必要な UDP 情報をきちんとルーティングしているかを確かめて下さい。 していなければ、たとえあなたが何をしようと解決できないでしょう。

次の例では fastws は高性能ワークステーションのホスト (インタフェース) 名で、freebox は低性能のイーサネットアダプタを備えた FreeBSD システムのホスト (インタフェース) 名です。 また /sharedfs はエクスポートされる NFS ファイルシステムであり (exports(5) を参照) 、 /project はエクスポートされたファイルシステムの、 クライアント上のマウントポイントとなります。
すべての場合において、アプリケーションによっては hard や soft, bg といった追加オプションがふさわしいかもしれないことに注意して下さい。

クライアント側 FreeBSD システム (freebox) の /etc/fstab の例は以下のとおりです。

```
fastws:/sharedfs /project nfs rw,-r=1024 0 0
```

freebox 上で手動で mount コマンドを実行する場合は次のようにして下さい。

```
# mount -t nfs -o -r=1024 fastws:/sharedfs /project
```

サーバ側 FreeBSD システム (fastws) の /etc/fstab の例は以下のとおりです。

```
freebox:/sharedfs /project nfs rw,-w=1024 0 0
```

fastws 上で手動で mount コマンドで実行する場合は次のようにして下さい。

```
# mount -t nfs -o -w=1024 freebox:/sharedfs /project
```

近いうちにどのような 16 ビットのイーサネットアダプタでも、上記の読み出し、書き込みサイズの制限なしで操作できるようになるでしょう。

失敗が発生したとき何が起きているか関心のある人に、 なぜ回復不可能なのかも含めて説明します。NFS は通常 (より小さいサイズへ分割されるかもしれませんが) 8 K の "ブロック" サイズで動作します。イーサネットのパケットサイズは最大 1500 バイト程度なので、 上位階層のコードにとっては 1 つのユニットであって、 NFS "ブロック" は複数のイーサネットパケットに分割されるものの、上位階層のコードにとっては 1 つのユニットであって、 ユニットとして受信され、組み立て直され、肯定応答 (ACK) されなければなりません。 高性能のワークステーションは次々に NFS ユニットを構成するパケットを、 標準の許す限り間隔を詰めて次々に送り出すことができます。小さく、容量の低いカードでは、 同じユニットの前のパケットがホストに転送される前に、後のパケットがそれを踏みつぶしてしまいます。
このため全体としてのユニットは、再構成も肯定応答もできません。 その結果、ワークステーションはタイムアウトして再送を試みますが、 8 K のユニット全体を再送しようとするので、 このプロセスは際限無く繰り返されてしまいます。

ユニットサイズをイーサネットのパケットサイズの 制限以下に抑えることにより、受信した完全なイーサネットパケットについて個々に肯定応答を返せることが保証されるので、デッドロック状態を避けられるようになります。

それでも、高性能なワークステーションが力任せに次々と PC システムにデータを送ったときには踏みつぶしが起きるかもしれません。
しかし、高性能のカードを使っていれば、NFS "ユニット"

で必ずそのような踏みつぶしが起きるとは限りません。
踏みつぶしが起きたら、影響を受けたユニットは再送されて、
受信され、組み立てられ、肯定応答される十分な見込みがあります。

20.7. ディスクレス稼働

FreeBSD マシンはネットワークを通じて起動でき、 そして NFS サーバからマウントしたファイルシステムを使用して、
ローカルディスクなしで動作することができます。
標準の設定ファイルを変更する以上の、システムの修正は必要ありません。
必要な要素のすべてが用意されているので、 このようなシステムを設定するのは簡単です。

- ネットワークを通じてカーネルを読み込む方法は、 少なくとも二つあります。
  - PXE: Intel® の Preboot Execution Environment システムは、一部のネットワークカードまたはマザーボードに組み込まれた、 スマートなブート ROM の一形態です。 詳細については pxeboot(8) を参照してください。
  - port の etherboot (net/etherboot) は、 ネットワークを通じてカーネルを起動する ROM 化可能なコードを提供します。 コードはネットワークカード上のブート PROM に焼き付けるか、あるいはローカルフロッピー (ハード) ディスクドライブ、 または動作している MS-DOS® システムから読み込むことができます。 多くのネットワークカードに対応しています。
- サンプルスクリプト (/usr/shared/examples/diskless/clone_root) はサーバ上で、ワークステーションのルートファイルシステムの作成と維持をやり易くします。
  このスクリプトは少し書き換えないといけないでしょうが、 早く取り掛かれるようにします。
- ディスクレスシステム起動を検知しサポートする標準のシステム起動ファイルが /etc 内にあります。
- 必要なら、NFS ファイルまたはローカルディスクのどちらかにスワップできます。

ディスクレスワークステーションを設定する方法はいろいろあります。 多くの要素が関わっており、その多くはローカルの状況に合わせてカスタマイズできます。下記は、 単純さと標準の FreeBSD 起動スクリプトとの互換性を強調した完全なシステムの設定を説明します。
記述されているシステムの特徴は次のとおりです。

- ディスクレスワークステーションは、 共有された読み取り専用の ルートファイルシステムと、共有された読み取り専用の /usr を使用します。

  ルート ファイルシステムは、 標準的な FreeBSD (典型的にはサーバの) のルートのコピーで、一部の設定ファイルが、ディスクレス稼働、
  また場合によってはそのワークステーションに特有のもので上書きされています。

  書き込み可能でなければならない ルート の部分は mfs(8) ファイルシステムで覆われます。
  システムが再起動するときにはすべての変更が失われるでしょう。

- カーネルは DHCP (または BOOTP) および TFTP を用いて etherboot によって読み込まれます。

記述されているとおり、 このシステムは安全ではありません。
ネットワークの保護された範囲で使用されるべきであり、
他のホストから信頼されてはいけません。

20.7.1. セットアップの手順

DHCP/BOOTP の設定

ネットワークを通じて設定を取得し、
ワークステーションを起動するために一般的に使用されるプロトコルには、 BOOTP と DHCP の 2 つがあります。 それらはワークステーションのブートストラップ時に何ヵ所かで使用されます。

- etherboot はカーネルを見つけるために DHCP (デフォルト) または BOOTP (設定オプションが必要) を使用します (PXE は DHCP を使用します) 。
- NFS ルートの場所を定めるためにカーネルは BOOTP を使用します。

BOOTP だけを使用するようにシステムを設定することもできます。 bootpd(8) サーバプログラムは FreeBSD のベースシステムに含まれています。

しかしながら、DHCP には BOOTP に勝る点が多々あります。 (よりよい設定ファイル、PXE が使えること、 そしてディスクレス稼働には直接関係しない多くの長所) ここでは BOOTP だけ利用する場合と、 BOOTP と DHCP を組み合わせた設定を扱います。特に ISC DHCP ソフトウェアパッケージを利用する後者の方法に重点をおきます。

ISC DHCP を使用する設定

isc-dhcp サーバは、 BOOTP および DHCP リクエストの両方に答えることができます。

4.4-RELEASE の時点で isc-dhcp 3.0 はベースシステムの一部では無くなりました。 まずはじめに net/isc-dhcp3-server port または対応する package をインストールする必要があるでしょう。 ports および package に関する一般的な情報については アプリケーションのインストール - packages と ports を参照してください。

isc-dhcp がインストールされると、 動作するために設定ファイルを必要とします (通常 /usr/local/etc/dhcpd.conf が指定されます) 。 下記にコメントを含めた例を示します。

```
default-lease-time 600;
max-lease-time 7200;
authoritative;

option domain-name "example.com";
option domain-name-servers 192.168.4.1;
option routers 192.168.4.1;

subnet 192.168.4.0 netmask 255.255.255.0 {
  use-host-decl-names on; ①
  option subnet-mask 255.255.255.0;
  option broadcast-address 192.168.4.255;

  host margaux {
    hardware ethernet 01:23:45:67:89:ab;
    fixed-address margaux.example.com;
    next-server 192.168.4.4;②
    filename "/tftpboot/kernel.diskless";③
    option root-path "192.168.4.4:/data/misc/diskless";④
```

```
        }
      }
```

① このオプションは host 宣言の値を、 ディスクレスホストへのホスト名として送るように dhcpd に指示します。 別の方法として、ホスト宣言内に option host-name margaux を加えるものがあります。

② TFTP サーバを next-server ディレクティブに指定します (デフォルトは DHCP サーバと同じホストを使います)。

③ カーネルとして etherboot が読み込むファイルを filename ディレクティブに指定します。

④ ルートファイルシステムへのパスを、 通常の NFS 書式で root-path オプションに指定します。

BOOTP を使用する設定

続けて、bootpd で同等のことをする設定です。 これは /etc/bootptab におきます。

BOOTP を使用するために、デフォルトではない NO_DHCP_SUPPORT オプション付きで etherboot をコンパイルしなければならないことと、PXE は DHCP を 必要 とすることに注意してください。 bootpd の唯一明白な利点は、 これがベースシステムに存在するということです。

```
        .def100:\
          :hn:ht=1:sa=192.168.4.4:vm=rfc1048:\
          :sm=255.255.255.0:\
          :ds=192.168.4.1:\
          :gw=192.168.4.1:\
          :hd="/tftpboot":\
          :bf="/kernel.diskless":\
          :rp="192.168.4.4:/data/misc/diskless":

        margaux:ha=0123456789ab:tc=.def100
```

Etherboot を用いるブートプログラムの準備

Etherboot のウェブサイト には主に Linux システムについて述べた 広範囲の文書 が含まれています。 しかし、それにもかかわらず有用な情報を含んでいます。 下記は FreeBSD システム上での etherboot の使用法についての概観を示します。

まずはじめに net/etherboot の package または port をインストールしなければなりません。 etherboot port は通常 /usr/ports/net/etherboot にあります。 ports ツリーがシステムにインストールされている場合、 このディレクトリ内で make を実行すれば、よきに計らってくれます。 ports および packages に関する情報は アプリケーションのインストール - packages と ports を参照してください。

ここで説明している方法では、ブートフロッピーを使用します。 他の方法 (PROM または DOS プログラム) については etherboot の文書を参照してください。

ブートフロッピーを作成するためには、 etherboot をインストールしたマシンのドライブにフロッピーディスクを挿入します。
それからカレントディレクトリを etherboot ツリー内の src ディレクトリにして次のように入力します。

```
# gmake bin32/devicetype.fd0
```

devicetype は ディスクレスワークステーションのイーサネットカードタイプに依存します。 正しい *devicetype* を決定するために、 同じディレクトリ内の NIC ファイルを参照してください。

TFTP および **NFS** サーバの設定

TFTP サーバ上で `tftpd` を有効にする必要があります。

1. `tftpd` が提供するファイルを置くディレクトリ (たとえば /tftpboot) を作成してください。
2. /etc/inetd.conf ファイルに以下の行を追加してください。

```
tftp    dgram   udp     wait    root  /usr/libexec/tftpd    tftpd -s /tftpboot
```

少なくとも PXE のいくつかのバージョンが TCP 版の TFTP を要求するようです。その場合 `dgram udp` を `stream tcp` に置き換えた 2 番目の行を追加してください。

3. `inetd` に設定ファイルを再読み込みさせてください。

```
# kill -HUP `cat /var/run/inetd.pid`
```

tftpboot ディレクトリはサーバ上のどこにでも置けます。 その場所が inetd.conf および dhcpd.conf の両方に設定されていることを確かめてください。

さらに NFS を有効にして NFS サーバの適切なファイルシステムをエクスポートする必要があります。

1. この行を /etc/rc.conf に追加してください。

```
nfs_server_enable="YES"
```

2. 下記を /etc/exports に加えることで、ディスクレスマシンのルートディレクトリが位置するファイルシステムをエクスポートしてください (ボリュームのマウントポイントを適当に調節し、 *margaux* をディスクレスワークステーションの名前に置き換えてください)。

```
/data/misc -alldirs -ro margaux
```

3. `mountd` に設定ファイルを再読み込みさせてください。 /etc/rc.conf 内で NFS をはじめて有効にする必要があったのなら、 代わりに再起動した方がよいかもしれません。

```
# kill -HUP `cat /var/run/mountd.pid`
```

ディスクレス用のカーネル構築

次のオプションを (通常のものに) 追加した、
ディスクレスクライアント用のカーネルコンフィグレーションファイルを作成してください。

```
options     BOOTP          # Use BOOTP to obtain IP address/hostname
options     BOOTP_NFSROOT  # NFS mount root filesystem using BOOTP info
options     BOOTP_COMPAT   # Workaround for broken bootp daemons.
```

BOOTP_NFSV3 および BOOTP_WIRED_TO を利用してもよいかもしれません (LINT を参照してください)。

カーネルを構築して (FreeBSD カーネルのコンフィグレーション を参照)、 dhcpd.conf に記述した名称で tftp ディレクトリにコピーしてください。

ルートファイルシステムの準備

dhcpd.conf に root-path として記載された
ディスクレスワークステーションのためのルートファイルシステムを作成する必要があります。

これを行う最も簡単な方法は /usr/shared/examples/diskless/clone_root
シェルスクリプトを使用することです。
このスクリプトは、少なくともファイルシステムが作成される場所 (DEST 変数)
を調節するために変更する必要があります。

説明についてはスクリプトの一番上にあるコメントを参照してください。
ベースシステムをどのように構築するか、 またファイルがどのようにディスクレス稼働、サブネット、
または個々のワークステーションに固有のバージョンによって、
選択的にオーバライドできるかを説明します。 また、ディスクレスな場合の /etc/fstab ファイルおよび
/etc/rc.conf ファイルの例を示します。

/usr/shared/examples/diskless 内の README
ファイルには、多くの興味深い背景情報が書かれています。 しかし diskless
ディレクトリ内の他の例と同じく、 clone_root と /etc/rc.diskless[12]
で実際に使われているものとは異なる設定方法が説明されています。 ここに書かれている方法は rc
スクリプトの変更が必要になりますが、 こちらの方が気に入ったというのでなければ、
参照にとどめてください。

スワップの設定

必要なら、サーバに置かれたスワップファイルに NFS 経由でアクセスできます。 bootptab または
dhcpd.conf の正確なオプションは、 現時点では明確には文書化されていません。 下記の設定例は isc-
dhcp 3.0rc11 を使用して動作したと報告されているものです。

1. dhcpd.conf に下記の行を追加してください。

```
            # Global section
```

```
option swap-path code 128 = string;
option swap-size code 129 = integer 32;

host margaux {
  ... # Standard lines, see above
  option swap-path "192.168.4.4:/netswapvolume/netswap";
  option swap-size 64000;
}
```

これは、少なくとも FreeBSD クライアントにおいては、 DHCP/BOOTP オプションコードの 128 は NFS スワップファイルへのパスで、オプションコード 129 は KB 単位のスワップサイズだということです。 もっと古いバージョンの `dhcpd` では `option option-128 "…` という書式が受け付けられましたが、 もはや対応していません。

代わりに、/etc/bootptab では次の書式を使います。

```
T128="192.168.4.4:/netswapvolume/netswap":T129=0000fa00
```

/etc/bootptab では、スワップの大きさは 16 進数で表さなければなりません。

2. NFS スワップファイルサーバ側でスワップファイルを作成します。

```
# mkdir /netswapvolume/netswap
# cd /netswapvolume/netswap
# dd if=/dev/zero bs=1024 count=64000 of=swap.192.168.4.6
# chmod 0600 swap.192.168.4.6
```

192.168.4.6 はディスクレスクライアントの IP アドレスです。

3. NFS スワップファイルサーバ上で /etc/exports に下記の行を追加してください。

```
        /netswapvolume  -maproot=0:10 -alldirs margaux
```

それから、上述したように mountd にエクスポートファイルを再読み込みさせてください。

雑多な問題

読み取り専用の /usr で動作させる

ディスクレスワークステーションが X を起動するように設定されている場合、 xdm 設定ファイルを調整しなければならないでしょう。 これはデフォルトでエラーファイルを /usr に置きます。

FreeBSD ではないサーバを使用する

ルートファイルシステムを提供するサーバが FreeBSD で動作していない場合、 FreeBSD マシン上でルートファイルシステムを作成し、 `tar` または `cpio` を利用して置きたい場所にコピーしなければならないでしょう。

この状況では、major/minor 整数サイズが異なっていることにより /dev 内のスペシャルファイルに関する問題が時々おこります。 この問題を解決するには、非 FreeBSD サーバからディレクトリをエクスポートして、 そのディレクトリを FreeBSD マシンでマウントし、FreeBSD マシン上で MAKEDEV を実行して正しいデバイスエントリを作成します (FreeBSD 5.0 およびそれ以降では、devfs(5) を使用してユーザに意識させずにデバイスノードを割り当てるので、これらのバージョンでは MAKEDEV は必要ありません)。

20.8. ISDN

ISDN 技術とハードウェアに関しては、 Dan Kegel's ISDN Page がよい参考になるでしょう。

手軽な ISDN の導入手順は以下のようになります。

- ヨーロッパ在住の方は ISDN カードの節に進んでください。
- ダイヤルアップ専用でない回線上で、
  インターネットプロバイダをつかってインターネットに接続するために ISDN を使用することを第一に考えている場合は、 ターミナルアダプタの使用を考えてみてください。
  この方法はもっとも柔軟性があり、 プロバイダを変更した場合の問題も少ないでしょう。
- 2 つの LAN を接続する場合や、 ISDN 専用線を使用する場合には、
  スタンドアロンなルータまたはブリッジの使用を勧めます。

費用はどの解決法を選ぶかを決める重要な要因です。
以下に、最も安価な方法から、高価な方法まで順に説明していきます。

20.8.1. ISDN カード

FreeBSD の ISDN 実装は、パッシブカードを使用した DSS1/Q.931 (または Euro-ISDN) 標準だけに対応しています。FreeBSD 4.4 からは、ファームウェアが他の信号プロトコルにも対応している一部のアクティブカードにも対応しました。 その中には、はじめて対応された一次群速度インタフェース (PRI) ISDN カードもあります。

isdn4bsd は IP over raw HDLC または同期 PPP を利用して他の ISDN ルータに接続できるようにします。 PPP では、カーネル PPP を sppp(4) ドライバを修正した isppp ドライバとともに利用するか、または ユーザプロセス ppp(8) を利用するかのどちらかになります。ユーザ ppp(8) を利用すると、二つ以上の ISDN B チャネルを併せて利用できます。 ソフトウェア 300 ボーモデムのような多くのユーティリティとともに、留守番電話アプリケーションも利用可能です。

FreeBSD が対応している PC ISDN カードの数は増加しており、ヨーロッパ全域や世界のその他多くの地域でうまく使えることが報告されています。

対応しているパッシブ ISDN カードのほとんどは Infineon (前身は Siemens) の ISAC/HSCX/IPAC ISDN チップセットを備えたカードですが、 Cologne Chip から供給されたチップを備えた ISDN カード (ISA バスのみ)、Winbond W6692 チップを備えた PCI カード、 Tiger300/320/ISAC チップセットを組み合わたカードの一部、 および AVM Fritz!Card PCI V.1.0 や AVM Fritz!Card PnP のようなベンダ独自のチップセットに基づいたカードもあります。

現在のところ、対応しているアクティブカードは AVM B1 (ISA および PCI) BRI カードと AVM T1 PCI PRI

カードです。

isdn4bsd についての文書は FreeBSD システム内の /usr/shared/examples/isdn/ ディレクトリまたは isdn4bsd のウェブサイトを参照してください。 そこにはヒントや正誤表や isdn4bsd ハンドブックのような、 さらに多くの文書に対するポインタがあります。

異なる ISDN プロトコルや、現在対応されていない ISDN PC カードに対応することや、その他 isdn4bsd を拡張することに興味があるなら、Hellmuth Michaelis <hm@FreeBSD.org> に連絡してください。

isdn4bsd のインストール、設定、 そしてトラブルシューティングに関して質問があれば freebsd-isdn メーリングリストが利用可能です。

20.8.2. ISDN ターミナルアダプタ

ターミナルアダプタ (TA) は ISDN で、 通常の電話線におけるモデムに相当するものです。

ほとんどの TA は、標準のヘイズ AT コマンドセットを使用しているので、単にモデムと置き換えて使うことができます。

TA は、基本的にはモデムと同じように動作しますが、接続方法は異なり、通信速度も古いモデムよりはるかに速くなります。 PPP の設定を、モデムの場合と同じように行ってください。
特にシリアル速度を使用できる最高速度に設定するのを忘れないでください。

プロバイダへの接続に TA を使用する最大のメリットは、動的 PPP を行えることです。 最近 IP アドレス空間がますます不足してきているため、 ほとんどのプロバイダは、 固定 IP アドレスを割り当てないようになっています。 ほとんどのスタンドアローンルータは、動的 IP アドレス割り当てに対応していません。

最近の ISDN ルータでは IP アドレスの動的割り当てに対応しているものも多いようです。
ただし制限がある場合もありますので、 詳しくはメーカに問い合わせてください。

TA を使用した場合の機能や接続の安定性は、使用している PPP デーモンに完全に依存します。そのため、FreeBSD で PPP の設定が完了していれば、使用している既存のモデムを ISDN の TA に簡単にアップグレードすることができます。ただし、それまでの PPP のプログラムに問題があった場合、その問題は TA に置き換えてもそのまま残ります。

最高の安定性を求めるのであれば、 ユーザランド PPP ではなく、カーネル PPPを使用してください。

以下の TA は、FreeBSD で動作確認ずみです。

- Motorola BitSurfer および Bitsurfer Pro
- Adtran

他の TA もほとんどの場合うまく動作するでしょう。TA のメーカーでは、TA がほとんどの標準モデム AT コマンドセットを受け付けるようにするよう努力しているようです。

外部 TA を使う際の最大の問題点は、モデムの場合と同じく良いシリアルカードが必要であるということです。

シリアルデバイスの詳細と、 非同期シリアルポートと同期シリアルポートの差を理解するには、FreeBSD シリアルハードウェアチュートリアルを参照してください。

標準の PC シリアルポート (非同期) に接続された TA は 128 Kbs の接続を行っていても、最大通信速度が 115.2 Kbs に制限されてしまいます。128 Kbs の ISDN の性能を最大限に生かすためには TA を同期シリアルカードに接続しなければなりません。

内蔵 TA を購入すれば、 同期/非同期問題を回避できるとは思わないでください。内蔵 TA には、 単に標準 PC シリアルポートのチップが内蔵されているだけです。 内蔵 TA の利点といえば、 シリアルケーブルを買わなくていいということと、
電源コンセントが一つ少なくて済むということくらいでしょう。

同期カードと TA の組合せでも、スタンドアロンのルータと同程度の速度は確保できます。 さらに、386 の FreeBSD マシンと組合せると、 より柔軟な設定が可能です。

同期カード/TA を選ぶか、スタンドアロンルータを選ぶかは、 多分に宗教的な問題です。 メーリングリストでもいくつか議論がありました。議論の全容については、 アーカイブを検索してください。

20.8.3. スタンドアロン ISDN ブリッジ/ルータ

ISDN ブリッジあるいはルータは、 FreeBSD あるいは他の OS に特有のものでは皆目ありません。 ルーティングやブリッジング技術に関する詳細は、 ネットワークの参考書をご覧ください。

この節では、 ルータとブリッジのどちらでもあてはまるように記述します。

ローエンド ISDN ルータ/ブリッジ製品は、 価格が下がってきていることもあり、 より広く選択されるようになるでしょう。ISDN ルータは、 ローカルイーサネットネットワークに直接接続し、 自身で他のブリッジ /ルータとの接続を制御する小さな箱です。PPP
や他の広く使用されているプロトコルをつかって通信するためのソフトウェアが組み込まれています。

ルータは、完全な同期 ISDN 接続を使用するため、通常の TA と比較してスループットが大幅に向上します。

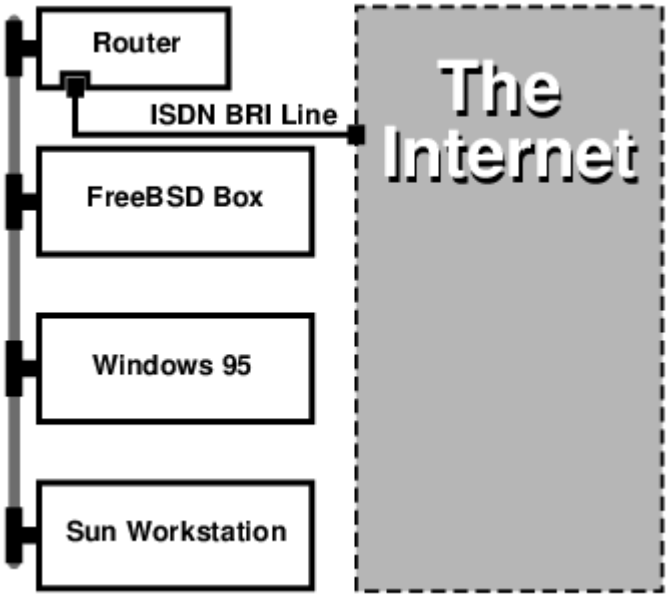
ISDN ルータ/ブリッジを使用する場合の最大の問題点は、 各メーカーの製品間に相性の問題がまだ存在することです。
インターネットプロバイダとの接続を考えている場合には、 プロバイダと相談することをお勧めします。

事務所の LAN と家庭の LAN の間など、二つの LAN セグメントの間を接続しようとしている場合は、 これはもっともメンテナンスが簡単で、安くあがる解決方法です。 接続の両側の機材を購入するので、 リンクがうまくいくであろうことを保証できます。

たとえば、 家庭のコンピュータや支店のネットワークを本社のネットワークに接続するためには、 以下のような設定が使用できます。

例 *45.* 支店または家庭のネットワーク

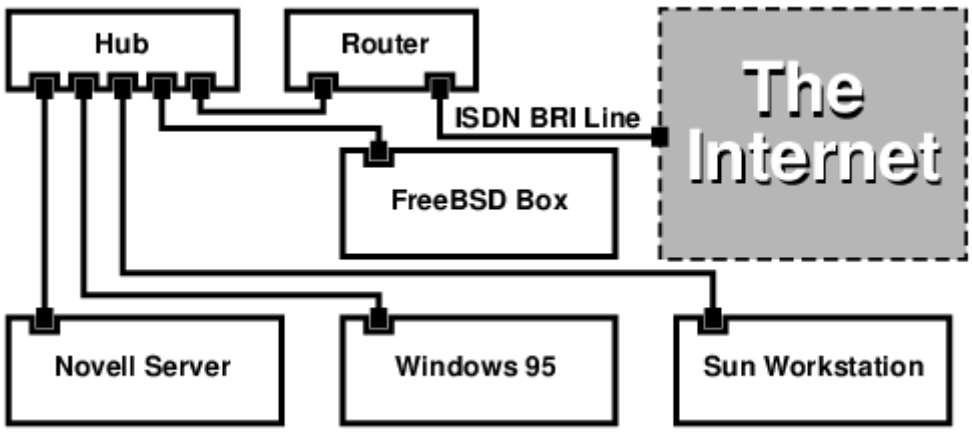
```
ネットワークは 10 Base 2 イーサネット ("thinnet") のバス型トポロジを用いています。ルータとネットワークの間は、 必要に応じて AUI/10BT トランシーバを使って接続してください。
```



家庭/支店で一台しかコンピュータを使用しないのであれば、
クロスのツイストペアケーブルを使用して、
直接スタンドアロンルータに接続することも可能です。

例 *46.* 本社 *LAN* や他の *LAN*

ネットワークは 10 base T イーサネット ("Twisted Pair") のスター型トポロジを用いています。



ほとんどのルータ/ブリッジの大きな利点は、 別々の二つのサイトに対して、同時 にそれぞれ独立した二つの PPP 接続が可能であることです。 これは、シリアルポートを 2 つもった特定の (通常は高価な) モデルを除いて、通常の TA では対応していません。 チャネルボンディングや MPP などと混同しないでください。

たとえば、事務所で専用線 ISDN 接続を使用していて、 別の ISDN 回線を購入したくないときには大変便利な機能です。この場合、
事務所のルータは、インターネットに接続するための一つの専用線 B チャネル接続 (64 Kbs) を管理し、別の B チャネルを他のデータ接続に使用できます。 2 つ目の B チャネルは他の場所とのダイアルイン、ダイアルアウトに使用したり、バンド幅を増やすために、 1 つ目の B チャネルと動的に結合すること (MPPなど) ができます。

またイーサネットブリッジは、IP パケット以外も中継できます。 IPX/SPX など、使用するすべてのプロトコルを送ることが可能です。

20.9. NIS/YP

20.9.1. NIS/YP とは?

NIS とは Network Information Services の略で Sun Microsystems によって UNIX® の (もともとは SunOS™ の) 集中管理のために開発されました。現在では事実上の業界標準になっており、 主要な UNIX® ライクシステム (Solaris™, HP-UX, AIX®, Linux, NetBSD, OpenBSD, FreeBSD、等々) はすべてこれをサポートしています。

NIS は元々、イエローページといっていましたが、 商標問題から Sun はその名前を変えました。古い用語 (および yp) はまだよく見られ、使用されています。

NIS は RPC を使ったクライアント/サーバシステムです。 これを使うと NIS ドメイン内のマシン間で、共通の設定ファイルを共有することができます。 また NIS を使うことでシステム管理者は最小限の設定データで NIS クライアントを立ち上げることができ、 1 ヶ所から設定データの追加、削除、変更が可能です。

NIS は Windows NT® のドメインシステムに似ています。 内部の実装は似ても似つかないものですが、基本的な機能を対比することはできます。

20.9.2. 知っておくべき用語 / プロセス

NIS サーバの立ち上げや NIS クライアントの設定など、 NIS を FreeBSD に導入するにあたって、目にするであろう用語や重要なユーザプロセスがいくつかあります。

用語	説明
NIS ドメイン名	NIS マスタサーバとそのクライアントすべて (スレーブサーバを含む) には NIS ドメイン名がついています。 Windows NT® ドメイン名と同様に、NIS ドメイン名は DNS とは何の関係もありません。
portmap	RPC (Remote Procedure Call, NIS で使用されるネットワークプロトコル) を利用するために実行しておかなければなりません。 `portmap` が動作していなければ、 NIS サーバを起動することも、 NIS クライアントとして動作させることもできません。
ypbind	NIS クライアントを NIS サーバに "結びつけ" ます。 これは NIS ドメイン名をシステムから取得し RPC を用いてサーバに接続します。`ypbind` は NIS 環境におけるクライアントとサーバ間の通信の中枢です。 クライアントマシンの `ypbind` が停止した場合は、NIS サーバへアクセスすることができなくなります。

用語	説明
ypserv	は NIS サーバでのみ実行されるべきもので、 NIS サーバプロセスそのものです。ypserv(8) が停止した場合、サーバはもはや NIS リクエストに応答することができなくなるでしょう (できれば、後を引き継ぐスレーブサーバがあるとよいでしょう)。 今まで使っていたサーバが機能を停止したとき、別のサーバに再接続しに行かない NIS の実装もいくつかあります (FreeBSD のものは違います)。 そのような場合に復帰するための唯一の方法は、サーバプロセス (あるいはサーバ全体)、もしくはクライアントの `ypbind` プロセスを再スタートすることです。
rpc.yppasswdd	NIS マスターサーバで動かすべき、もう一つのプロセスです。これは NIS クライアントが NIS パスワードを変更することを可能にするデーモンです。 このデーモンが動作していないときは、ユーザは NIS マスタサーバにログインし、そこでパスワードを変更しなければなりません。

20.9.3. 動作のしくみ

NIS 環境にあるホストは、 マスターサーバ、スレーブサーバ、クライアントの 3 種類に分類されます。
サーバは、ホストの設定情報の中心的な情報格納庫の役割をします。
マスターサーバは元となる信頼できる情報を保持し、
スレーブサーバは冗長性を確保するためこの情報をミラーします。
そしてクライアントは、サーバから情報の提供を受けて動作します。

この方法を用いることで、数多くのファイルにある情報が共有できます。 よく NIS で共有されるのは、master.passwd や group, hosts といったファイルです。 クライアント上のプロセスが、通常ならローカルのファイルにある情報を必要とするときは、 クライアントは代わりに接続している NIS サーバに問い合わせを行います。

マシンの分類

- *NIS* マスターサーバ。 このサーバは Windows NT® で言うところのプライマリドメインコントローラにあたります。 すべての NIS クライアントで利用されるファイルを保守します。 passwd や group、 その他 NIS クライアントが参照するファイルは、 マスターサーバにあります。

一つのマシンが一つ以上の NIS ドメインのマスターサーバになることは可能です。しかし、ここでは比較的小規模の NIS 環境を対象としているため、そのような場合については扱いません。

- *NIS* スレーブサーバ。 Windows NT® のバックアップドメインコントローラに似たもので、 NIS スレーブサーバは NIS マスターサーバのデータファイルのコピーを保持します。 NIS

スレーブサーバは重要な環境で必要とされる冗長性を提供し、
マスターサーバの負荷のバランスをとります。 NIS
クライアントは常に最初にレスポンスを返したサーバを NIS サーバとして接続しますが、
これにはスレーブサーバも含まれます。

- *NIS* クライアント。 NIS クライアントは大部分の Windows NT® ワークステーションのように、ログオンに際して NIS サーバ (Windows NT® ワークステーションの場合は Windows NT® ドメインコントローラ) に接続して認証します。

20.9.4. NIS/YP を使う

この節では NIS 環境の立ち上げ例を取り上げます。

この節ではあなたが FreeBSD 3.3 以降を使っているものとします。
ここで与えられる指示は おそらく FreeBSD の 3.0
以降のどのバージョンでも機能するでしょうが、 それを保証するものではありません。

計画を立てる

あなたが大学の小さな研究室の管理人であるとしましょう。 この研究室は 15 台の FreeBSD マシンからなっていて、 現在はまだ集中管理されていません。 すなわち、各マシンは /etc/passwd と /etc/master.passwd を各々が持っています。 これらのファイルは手動でお互いに同期させています。 つまり現時点では、新しいユーザをあなたが追加するとき、 `adduser` を 15 ヶ所すべてで実行しなければなりません。 これは明らかに変える必要があるため、 あなたはこのうち 2 台をサーバにして NIS を導入することを決めました。

その結果、研究室の設定はこのようなものになります。

マシンの名前	**IP** アドレス	役割
`ellington`	`10.0.0.2`	NIS マスタ
`coltrane`	`10.0.0.3`	NIS スレーブ
`basie`	`10.0.0.4`	教員用のワークステーション
`bird`	`10.0.0.5`	クライアントマシン
`cli[1-11]`	`10.0.0.[6-17]`	その他のクライアントマシン

もし NIS によるシステム管理の設定を行なうのが初めてなら、 どのようにしたいのか、
ひととおり最後まで考えてみることをお勧めします。 ネットワークの規模によらず、
いくつか決めるべきことがあるからです。

NIS ドメイン名を決める

ここでいうドメイン名は、今まであなたが使っていた、 いわゆる "ドメイン名" と呼んでいたものとは違います。 正確には "NIS ドメイン名" と呼ばれます。 クライアントがサーバに情報を要求するとき、 その要求には自分が属する NIS ドメインの名前が含まれています。 これは 1 つのネットワークに複数のサーバがある場合に、 どのサーバが要求を処理すれば良いかを決めるために使われます。 NIS ドメイン名とは、 関連のあるホストをグループ化するための名前である、 と考えると良いでしょう。

組織によってはインターネットのドメイン名を NIS ドメイン名に使っているところがあります。

これはネットワークのトラブルをデバッグするときに混乱の原因となるため、 お勧めできません。 NIS ドメイン名はネットワーク内で一意なければいけません。そして、
ドメイン名がドメインに含まれるマシンを表すようなものであれば分かり易いです。 たとえば Acme 社のアート (Art) 部門であれば NIS ドメイン名を "acme-art" とすれば良いでしょう。この例では NIS ドメイン名として *test-domain* を使用します。

しかしながらオペレーティングシステムによっては (特に SunOS™)、 NIS ドメイン名をネットワークドメイン名として使うものもあります。
あなたのネットワークにそのような制限のあるマシンが 1 台でもあるときは、NIS のドメイン名としてインターネットのネットワークドメイン名を使わなければ いけません。

サーバマシンの物理的必要条件

NIS サーバとして使うマシンを選ぶ際には、 いくつか注意すべき点があります。 NIS に関する困ったことの一つに、 クライアントのサーバへの依存度があります。 クライアントが自分の NIS ドメインのサーバに接続できないと、 マシンが使用不能になることがあまりに多いのです。
もし、ユーザやグループに関する情報が得られなければ、
ほとんどのシステムは一時的に停止してしまいます。
こういったことを念頭に置いて、頻繁にリブートされるマシンや、
開発に使われそうなマシンを選ばないようにしなければなりません。 理想的には NIS サーバはスタンドアロンで NIS サーバ専用のマシンにするべきです。
ネットワークの負荷が重くなければ、 他のサービスを走らせているマシンを NIS サーバにしてもかまいません。 ただし NIS サーバが使えなくなると、 すべてのクライアントに影響をおよぼす、 という点には注意しなければなりません。

NIS サーバ

元となるすべての NIS 情報は、 NIS マスターサーバと呼ばれる 1 台のマシンに格納されます。この情報が格納されるデータベースを NIS マップと呼びます。 FreeBSD では、このマップは /var/yp/[domainname] に置かれます。 [domainname] は、 サーバがサービスする NIS ドメインです。
1 台の NIS サーバが複数のドメインをサポートすることも可能です。つまり、このディレクトリを各々のドメインごとに作ることができます。 それぞれのドメインは、独立したマップの集合を持つことになります。

NIS のマスターサーバとスレーブサーバ上では、 `ypserv` デーモンがすべての NIS 要求を処理します。 `ypserv` は NIS クライアントからの要求を受け付け、
ドメイン名とマップ名を対応するデータベースファイルへのパスに変換し、
データをクライアントに返送します。

NIS マスターサーバの設定

やりたいことにもよりますが NIS マスターサーバの設定は比較的単純です。 FreeBSD は初期状態で NIS に対応しています。 必要なのは以下の行を /etc/rc.conf に追加することだけで、 あとは FreeBSD がやってくれます。

```
nisdomainname="test-domain"
```

1. この行はネットワークの設定後に (たとえば再起動後に) NIS のドメイン名を *test-domain* に設定します。

```
nis_server_enable="YES"
```

2. これは FreeBSD に次にネットワークが立ち上がったとき NIS のサーバプロセスを起動させます。

```
nis_yppasswdd_enable="YES"
```

3. これは `rpc.yppasswdd` デーモンを有効にします。上述したようにこれはユーザが NIS のパスワードをクライアントのマシンから変更することを可能にします。

NIS の設定によっては、 さらに他のエントリを付け加える必要があるかもしれません。詳細については、下記の NIS クライアントとしても動作している NIS サーバ 節を参照してください。

さて、あとはスーパユーザ権限で `/etc/netstart` コマンドを実行するだけです。 これにより /etc/rc.conf で定義された値を使ってすべての設定が行なわれます。

NIS マップの初期化

NIS マップ とは /var/yp ディレクトリにあるデータベースファイルです。 これらは NIS マスタの /etc ディレクトリの設定ファイルから作られます。 唯一の例外は /etc/master.passwd ファイルです。これは `root` や他の管理用アカウントのパスワードまでその NIS ドメインのすべてのサーバに伝えたくないという、 もっともな理由によるものです。このため NIS マップの初期化の前に以下を行う必要があります。

```
# cp /etc/master.passwd /var/yp/master.passwd
# cd /var/yp
# vi master.passwd
```

システムに関するアカウント (`bin`, `tty`, `kmem`, `games` など) や、NIS クライアントに伝えたくないアカウント (たとえば `root` や他の UID が 0 (スーパユーザ) のアカウント) をすべて NIS マップから取り除かなければなりません。

/var/yp/master.passwd がグループまたは誰もが読めるようになっていないようにしてください (モード 600)! 必要なら `chmod` コマンドを使ってください。

すべてが終わったら NIS マップを初期化します! FreeBSD には、これを行うために `ypinit` という名のスクリプトが含まれています (詳細はそのマニュアルページをご覧ください)。このスクリプトはほとんどの UNIX® OS に存在しますが、すべてとは限らないことを覚えておいてください。 Digital Unix/Compaq Tru64 UNIX では `ypsetup` と呼ばれています。NIS マスタのためのマップを作るためには `-m` オプションを `ypinit` に与えます。上述のステップを完了しているなら、以下を実行して NIS マップを生成します。

```
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 you don't, 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.
```

`ypinit` は /var/yp/Makefile を /var/yp/Makefile.dist から作成します。作成された時点では、そのファイルはあなたが FreeBSD マシンだけからなるサーバが 1 台だけの NIS 環境を扱っていると仮定しています。 *test-domain* はスレーブサーバを一つ持っていますので /var/yp/Makefile を編集しなければなりません。

```
ellington# vi /var/yp/Makefile
```

以下の行を (もし既にコメントアウトされていないならば) コメントアウトしなければなりません。

```
NOPUSH = "True"
```

NIS スレーブサーバの設定

NIS スレーブサーバの設定はマスターサーバの設定以上に簡単です。 スレーブサーバにログオンし /etc/rc.conf ファイルを前回と同様に編集します。唯一の違うところは `ypinit` の実行に `-s` オプションを使わなければいけないことです。 `-s` オプションは NIS マスターサーバの名前を要求し、コマンドラインは以下のようになります。

```
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 you don't, 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.
Don't forget to update map ypservers on ellington.
```

この例の場合 /var/yp/test-domain というディレクトリが必要になります。 NIS マスターサーバのマップファイルのコピーは、 このディレクトリに置いてください。 これらを確実に最新のものに維持する必要があります。 次のエントリをスレーブサーバの /etc/crontab に追加することで、最新のものに保つことができます。

```
20      *       *       *       *       root    /usr/libexec/ypxfr passwd.byname
21      *       *       *       *       root    /usr/libexec/ypxfr passwd.byuid
```

この二行はスレーブサーバにあるマップファイルを、
マスターサーバのマップファイルと同期させるものです。
このエントリは必須というわけではありませんが、マスターサーバは NIS マップに対する変更をスレーブサーバに伝えようとしますし、
サーバが管理するシステムにとってパスワード情報はとても重要なので、
強制的に更新してしまうことはよい考えです。特に、
マップファイルの更新がきちんと行なわれるかどうかわからないくらい混雑するネットワークでは、
重要になります。

スレーブサーバ上でも `/etc/netstart` コマンドを実行して、NIS サーバを再起動してください。

NIS クライアント

NIS クライアントは `ypbind` デーモンを使って、特定の NIS サーバとの間に結合 (binding) と呼ばれる関係を成立させます。 `ypbind` はシステムのデフォルトのドメイン (`domainname` コマンドで設定されます) を確認し、RPC 要求をローカルネットワークにブロードキャストします。 この RPC 要求により `ypbind` が結合を成立させようとしているドメイン名が指定されます。 要求されているドメイン名に対してサービスするよう設定されたサーバが
ブロードキャストを受信すると、 サーバは `ypbind` に応答し`ypbind` は応答のあったサーバのアドレスを記録します。複数のサーバ
(たとえば一つのマスターサーバと、複数のスレーブサーバ) が利用可能な場合、`ypbind` は、最初に応答したサーバのアドレスを使用します。 これ以降、クライアントのシステムは、 すべての NIS の要求をそのサーバに向けて送信します。 `ypbind` は、
サーバが順調に動作していることを確認するため、 時々 "ping" をサーバに送ります。 反応が戻ってくるべき時間内に ping に対する応答が来なければ、 `ypbind` は、そのドメインを結合不能 (unbound) として記録し、別のサーバを見つけるべく、
再びブロードキャストパケットの送信を行います。

NIS クライアントの設定

FreeBSD マシンを NIS クライアントにする設定は非常に単純です。

1. ネットワークの起動時に NIS ドメイン名を設定して `ypbind` を起動させるために /etc/rc.conf ファイルを編集して以下の行を追加します。

   ```
   nisdomainname="test-domain"
   nis_client_enable="YES"
   ```

2. NIS サーバから、 利用可能なパスワードエントリをすべて取り込むため、 /etc/master.passwd

からすべてのユーザアカウントを取り除いて、 vipw コマンドで以下の行を /etc/master.passwd の最後に追加します。

```
+::::::::::
```

この行によって NIS サーバのパスワードマップにアカウントがある人全員にアカウントが与えられます。 この行を変更すると、 さまざまな NIS クライアントの設定を行なうことが可能です。 詳細は ネットグループ を、さらに詳しい情報については、O'Reilly の Managing NFS and NIS を参照してください。

/etc/master.passwd 内に少なくとも一つのローカルアカウント (つまり NIS 経由でインポートされていないアカウント) を置くべきです。 また、このアカウントは wheel グループのメンバーであるべきです。 NIS がどこか調子悪いときには、 リモートからこのアカウントでログインし、 root になって修復するのに利用できます。

3. NIS サーバにあるすべてのグループエントリを取り込むため、 以下の行を /etc/group に追加します。

```
+:*::
```

上記の手順がすべて完了すれば、 ypcat passwd によって NIS サーバの passwd マップが参照できるようになっているはずです。

20.9.5. NIS セキュリティ

一般にドメイン名さえ知っていれば、 どこにいるリモートユーザでも ypserv(8) に RPC を発行して NIS マップの内容を引き出すことができます。 こういった不正なやりとりを防ぐため、 ypserv(8) には securenets と呼ばれる機能があります。これは、アクセスを決められたホストだけに制限するのに使える機能です。 ypserv(8) は起動時に /var/yp/securenets ファイルから securenets に関する情報を読み込みます。

上記のパス名は -p オプションで指定されたパス名によって変わります。このファイルは、空白で区切られたネットワーク指定とネットマスクのエントリからなっていて、 "#" で始まる行はコメントとみなされます。 簡単な securenets ファイルの例を以下に示します。

```
# 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
10.0.0.0      255.255.240.0
```

ypserv(8) が上記のルールの一つと合致するアドレスからの要求を受け取った場合、処理は正常に行なわれます。もしアドレスがルールに合致しなければ、その要求は無視されて警告メッセージがログに記録されます。また /var/yp/securenets が存在しない場合、 `ypserv` はすべてのホストからの接続を受け入れます。

`ypserv` は Wietse Venema 氏による tcpwrapper パッケージもサポートしています。そのため /var/yp/securenets の代わりに tcpwrapper の設定ファイルを使ってアクセス制御を行なうことも可能です。

これらのアクセス制御機能は一定のセキュリティを提供しますが、
どちらも特権ポートのテストのような "IP spoofing" 攻撃に対して脆弱です。すべての NIS 関連のトラフィックはファイアウォールでブロックされるべきです。

/var/yp/securenets を使っているサーバは、古い TCP/IP 実装を持つ正当なクライアントへのサービスに失敗することがあります。
これらの実装の中にはブロードキャストのホストビットをすべて 0 でセットしてしまったり、
ブロードキャストアドレスの計算でサブネットマスクを見落としてしまったりするものがあります。
これらの問題にはクライアントの設定を正しく行なえば解決できるものもありますが、
問題となっているクライアントシステムを引退させるか、/var/yp/securenets を使わないようにしなければならないものもあります。

このような古風な TCP/IP の実装を持つサーバで /var/yp/securenets を使うことは実に悪い考えであり、あなたのネットワークの大部分において NIS の機能喪失を招きます。

tcpwrapper パッケージを使うとあなたの NIS サーバのレイテンシ (遅延) が増加します。特に混雑したネットワークや遅い NIS サーバでは、遅延の増加によって、
クライアントプログラムのタイムアウトが起こるかもしれません。
一つ以上のクライアントシステムがこれらの兆候を示したなら、
あなたは問題となっているクライアントシステムを NIS スレーブサーバにして自分自身に結び付くように強制すべきです。

20.9.6. 何人かのユーザのログオンを遮断する

わたしたちの研究室には `basie` という、教員専用のマシンがあります。わたしたちはこのマシンを NIS ドメインの外に出したくないのですが、マスタ NIS サーバの passwd ファイルには教員と学生の両方が載っています。どうしたらいいでしょう?

当該人物が NIS のデータベースに載っていても、そのユーザがマシンにログオンできないようにする方法があります。そうするには *-username* をクライアントマシンの /etc/master.passwd ファイルの末尾に付け足します。*username* はあなたがログインさせたくないと思っているユーザのユーザ名です。これは `vipw` で行うべきです。`vipw` は /etc/master.passwd への変更をチェックし、編集終了後パスワードデータベースを再構築します。たとえば、ユーザ *bill* が

basie にログオンするのを防ぎたいなら、以下のようにします。

```
basie# vipw
[add -bill to the end, exit]
vipw: rebuilding the database...
vipw: done

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:/sbin/nologin
operator:*:2:5::0:0:System &:/:/sbin/nologin
bin:*:3:7::0:0:Binaries Commands and Source,,,:/:/sbin/nologin
tty:*:4:65533::0:0:Tty Sandbox:/:/sbin/nologin
kmem:*:5:65533::0:0:KMem Sandbox:/:/sbin/nologin
games:*:7:13::0:0:Games pseudo-user:/usr/games:/sbin/nologin
news:*:8:8::0:0:News Subsystem:/:/sbin/nologin
man:*:9:9::0:0:Mister Man Pages:/usr/shared/man:/sbin/nologin
bind:*:53:53::0:0:Bind Sandbox:/:/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:/sbin/nologin
pop:*:68:6::0:0:Post Office Owner:/nonexistent:/sbin/nologin
nobody:*:65534:65534::0:0:Unprivileged user:/nonexistent:/sbin/nologin
+:::::::::
-bill

basie#
```

20.9.7. ネットグループの利用

前節までに見てきた手法は、 極めて少ないユーザ/マシン向けに個別のルールを必要としている場合にはうまく機能します。
しかし大きなネットワークでは、 ユーザに触られたくないマシンへログオンを防ぐのを 忘れるでしょうし、 そうでなくとも各マシンを個別に設定して回らなければならず、 集中管理という NIS の恩恵を失ってしまいます。

NIS の開発者はこの問題を ネットグループ と呼ばれる方法で解決しました。 その目的と意味合いは UNIX® のファイルシステムで使われている一般的なグループと比較できます。 主たる相違は数値 ID が存在しないことと、
ユーザアカウントと別のネットグループを含めたネットグループを定義できることです。

ネットグループは百人/台以上のユーザとマシンを含む、
大きく複雑なネットワークを扱うために開発されました。
あなたがこのような状況を扱わなければならないなら便利なものなのですが、
一方で、この複雑さは単純な例でネットグループの説明をすることをほとんど不可能にしています。
この節の残りで使われている例は、この問題を実演しています。

あなたの行なった、 研究室への NIS の導入の成功が上司の目に止ったとしましょう。
あなたの次の仕事は、あなたの NIS

ドメインをキャンパスの他のいくつものマシンを覆うものへ拡張することです。
二つの表は新しいユーザと新しいマシンの名前とその説明を含んでいます。

ユーザの名前	説明
alpha, beta	IT 学科の通常の職員
charlie, delta	IT 学科の新しい見習い
echo, foxtrott, golf, ...	一般の職員
able, baker, ...	まだインターン

マシンの名前	説明
war, death, famine, pollution	最も重要なサーバ。IT 職員だけがログオンを許されます。
pride, greed, envy, wrath, lust, sloth	あまり重要でないサーバ。 IT 学科の全員がログオンを許されます。
one, two, three, four, ...	通常のワークステーション。 本当の 職員だけがログオンを許されます。
trashcan	重要なデータの入っていないひどく古いマシン。 インターンでもこのマシンの使用を許されます。

もしあなたがこの手の制限を各ユーザを個別にブロックする形で実装するなら、
あなたはそのシステムにログオンすることが許されていない各ユーザについて *-user* という 1 行を、各システムの passwd に追加しなければならなくなるでしょう。 もしあなたが 1 エントリでも忘れればトラブルに巻き込まれてしまいます。
最初のセットアップの時にこれを正しく行えるのはありえることかも知れませんが、
遂には連日の業務の間に例の行を追加し忘れてしまうでしょう。
結局マーフィーは楽観主義者だったのです。

この状況をネットグループで扱うといくつかの有利な点があります。 各ユーザを別個に扱う必要はなく、
ユーザを一つ以上のネットグループに割り当て、
ネットグループの全メンバのログインを許可したり禁止したりすることができます。
新しいマシンを追加するときはネットグループへログインの制限を定義するだけ、
新しいユーザを追加するときはそのユーザを一つ以上のネットグループへ追加するだけで、
それぞれ行なうことができます。 これらの変更は互いに独立なので、
"ユーザとマシンの組合わせをどうするか" は存在しなくなります。 あなたの NIS のセットアップが注意深く計画されていれば、
マシンへのアクセスを認めるにも拒否するにも中心の設定をたった一カ所変更するだけです。

最初のステップは NIS マップネットグループの初期化です。 FreeBSD の ypinit(8) はこのマップをデフォルトで作りませんが、 その NIS の実装はそれが作られさえすればそれをサポートするものです。 空のマップを作るには、単に

```
ellington# vi /var/yp/netgroup
```

とタイプして内容を追加していきます。 わたしたちの例では、すくなくとも IT 職員、IT 見習い、一般職員、 インターンの 4 つのネットグループが必要です。

```
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)
```

IT_EMP, IT_APP 等はネットグループの名前です。
それぞれの括弧で囲まれたグループが一人以上のユーザアカウントをそれに登録しています。 グループの 3 つのフィールドは

1. その記述が有効なホスト (群) の名称。
   ホスト名を特記しなければそのエントリはすべてのホストで有効です。
   もしあなたがホスト名を特記するなら、
   あなたは闇と恐怖と全き混乱の領域に入り込んでしまうでしょう。
2. このネットグループに所属するアカウントの名称。
3. そのアカウントの NIS ドメイン。 もしあなたが一つ以上の NIS ドメインの不幸な仲間なら、
   あなたは他の NIS ドメインからあなたのネットグループにアカウントを導入できます。

各フィールドには、ワイルドカードが使えます。 詳細は netgroup(5) をご覧ください。

8 文字以上のネットグループ名は、特にあなたの NIS ドメインで他のオペレーティングシステムを走らせているときは使うべきではありません。 名前には大文字小文字の区別があります。
そのためネットグループ名に大文字を使う事は、
ユーザやマシン名とネットグループ名を区別する簡単な方法です。

(FreeBSD 以外の) NIS クライアントの中には多数のエントリを扱えないものもあります。 たとえば SunOS™ の古い版では 15 以上のエントリを含むネットグループはトラブルを起こします。 この制限は 15 ユーザ以下のサブネットグループをいくつも作り、
本当のネットグループはこのサブネットグループからなるようにすることで回避できます。

```
BIGGRP1  (,joe1,domain)  (,joe2,domain)  (,joe3,domain) [...]
BIGGRP2  (,joe16,domain)  (,joe17,domain) [...]
BIGGRP3  (,joe31,domain)  (,joe32,domain)
BIGGROUP  BIGGRP1 BIGGRP2 BIGGRP3
```

単一のネットグループに 225 人以上のユーザをいれたいときは、このやり方を繰り返すことができます。

新しい NIS マップの有効化と配布は簡単です。

```
ellington# cd /var/yp
ellington# make
```

これで新しい 3 つの NIS マップ netgroup, netgroup.byhost, netgroup.byuser ができるはずです。新しい NIS マップが利用できるか確かめるには ypcat(1) を使います。

```
ellington% ypcat -k netgroup
ellington% ypcat -k netgroup.byhost
ellington% ypcat -k netgroup.byuser
```

最初のコマンドの出力は /var/yp/netgroup の内容に似ているはずです。2 番目のコマンドはホスト別のネットグループを作っていなければ出力されません。3 番目のコマンドはユーザに対するネットグループのリストを得るのに使えます。

クライアント側の設定は非常に簡単です。サーバ *war* を設定するには、vipw(8) を実行して以下の行

```
+:::::::::
```

を

```
+@IT_EMP:::::::::
```

に入れ替えるだけです。

今、ネットグループ *IT_EMP* で定義されたユーザのデータだけが *war* のパスワードデータベースに読み込まれ、そのユーザだけがログインを許されています。

残念ながらこの制限はシェルの ~ の機能や、ユーザ名や数値の ユーザ ID の変換ルーチンにも影響します。つまり、`cd ~user` はうまく動かず、`ls -l` はユーザ名のかわりに数値の ID を表示し `find . -user joe -print` は "No such user" で失敗します。これを避けるためには、すべてのユーザのエントリを サーバにログインすることを許さずに読み込まなければなりません。

これはもう一行を /etc/master.passwd に追加することで実現できます。その行は以下の

`+:::::::::/sbin/nologin` を含んでおり、これは "すべてのエントリを読み込むが、読み込まれたエントリのシェルは /sbin/nologin で置き換えられる" ということを意味します。passwd エントリの他のフィールドを /etc/master.passwd の既定値から置き換えることも可能です。

`:::::::::/sbin/nologin` の行が `@IT_EMP:::::::::` の行より後ろに位置することに注意してください。さもないと NIS から読み込まれた全ユーザが /sbin/nologin をログインシェルとして持つことになります。

この変更の後では、新しい職員が IT 学科に参加しても NIS マップを一つ書き換えるだけで済みます。同様にして、あまり重要でないサーバのローカルの /etc/master.passwd のかつての `+:::::::::` 行を以下のように置き換えます。

```
+@IT_EMP:::::::::
+@IT_APP:::::::::
```

```
+:::::::::/sbin/nologin
```

この行は、一般のワークステーションでは以下のようになります。

```
+@IT_EMP:::::::::
+@USERS:::::::::
+:::::::::/sbin/nologin
```

これでしばらく順調に運用していましたが、 数週間後、ポリシに変更がありました。 IT 学科はインターンを雇い始め、IT
インターンは一般のワークステーションと余り重要ではないサーバを使うことが許され、 IT 見習いはメインサーバへのログインが許されました。 あなたは新たなネットグループ IT_INTERN を追加して新しい IT インターンたちをそのグループに登録し、
すべてのマシンの設定を変えて回ることにしました。 古い諺にこうあります。
"集中管理における過ちは、大規模な混乱を導く"。

いくつかのネットグループから新たなネットグループを作るという NIS の機能は、このような状況に対処するために利用できます。
その方法の一つは、役割別のネットグループを作ることです。
たとえば、重要なサーバへのログイン制限を定義するために *BIGSRV* というネットグループを作り
あまり重要ではないサーバへは *SMALLSRV* というネットグループを、そして一般のワークステーション用に *USERBOX* という第 3 のネットグループを 作ることができます。これらのネットグループの各々は、
各マシンにログインすることを許されたネットグループを含みます。 あなたの NIS マップネットグループの新しいエントリは、 以下のようになるはずです。

```
BIGSRV    IT_EMP  IT_APP
SMALLSRV  IT_EMP  IT_APP  ITINTERN
USERBOX   IT_EMP  ITINTERN USERS
```

このログイン制限の定義法は、
同一の制限を持つマシンのグループを定義できるときには便利なものです。
残念ながらこのようなケースは例外的なものです。 ほとんどの場合、
各マシンに基づくログイン制限の定義機能が必要となるでしょう。

マシンごとのネットグループの定義は、
上述したようなポリシの変更を扱うことができるもうひとつの方法です。 このシナリオでは、各マシンの /etc/master.passwd は "+" で始まる 2 つの行からなります。
最初のものはそのマシンへのログインを許されたアカウントを追加するもので、 2 番目はその他のアカウントを /sbin/nologin をシェルとして追加するものです。
マシン名をすべて大文字で記述したものをネットグループの名前として使うのは良い考えです。
言い換えれば、件の行は次のようになるはずです。

```
+@BOXNAME:::::::::
+:::::::::/sbin/nologin
```

一度、各マシンに対してこの作業を済ませてしまえば、　　二度とローカルの　　/etc/master.passwd を編集する必要がなくなります。　以降のすべての変更は　NIS　マップの編集で扱うことができます。
以下はこのシナリオに対応するネットグループマップに、いくつかの便利な定義を追加した例です。

```
# 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]
```

もしユーザアカウントを管理するのにデータベースの類を使っているなら、
データベースのレポートツールからマップの最初の部分を作れるようにするべきです。
そうすれば、新しいユーザは自動的にマシンにアクセスできるでしょう。

最後に使用上の注意を:　　マシン別のネットグループを使うことが常に賢明というわけではありません。
あなたが数ダースから数百の同一の環境のマシンを学生の研究室に配置しているのならば、　　NIS マップのサイズを手頃な範囲に押さえるために、
マシン別のネットグループのかわりに役割別のネットグループを使うべきです。

20.9.8. 忘れてはいけないこと

NIS 環境にある今、 今までとは違ったやり方が必要なことがいくつかあります。

- 研究室にユーザを追加するときは、それをマスター NIS サーバに だけ 追加しなければならず、さらに *NIS* マップを再構築することを忘れてはいけません。 これを忘れると新しいユーザは NIS マスタ以外のどこにもログインできなくなります。 たとえば、新しくユーザ "jsmith" をラボに登録したいときは以下のようにします。

  ```
  # pw useradd jsmith
  # cd /var/yp
  # make test-domain
  ```

  `pw useradd jsmith` のかわりに `adduser jsmith` を使うこともできます。

- 管理用アカウントを *NIS* マップから削除してください。 管理用アカウントやパスワードを、それらのアカウントへアクセスさせてはいけないユーザが居るかも知れないマシンにまで伝えて回りたいとは思わないでしょう。

- *NIS* のマスタとスレーブをセキュアに、 そして機能停止時間を最短に保ってください。 もし誰かがこれらのマシンをクラックしたり、 あるいは単に電源を落としたりすると、彼らは実質的に多くの人を研究室へログインできなくしてしまえます。

  これはどの集中管理システムにとってももっとも大きな弱点でしょう。 あなたの NIS サーバを守らなければ怒れるユーザと対面することになるでしょう!

20.9.9. NIS v1 との互換性

FreeBSD の ypserv は、 NIS v1 クライアントを部分的にサポートしています。 FreeBSD の NIS 実装は NIS v2 プロトコルのみを使用していますが、
ほかの実装では、古いシステムとの下位互換性を持たせるため v1 プロトコルをサポートしているものもあります。 そのようなシステムに付いている ypbind デーモンは、必要がないにもかかわらず NIS v1 のサーバとの結合を成立させようとします (しかも v2 サーバからの応答を受信した後でも、 ブロードキャストをし続けるかも知れません)。 FreeBSD の ypserv は、 クライアントからの通常のリクエストはサポートしていますが、 v1 のマップ転送リクエストはサポートしていないことに注意してください。 つまり FreeBSD の ypserv を、 v1 だけをサポートするような古い NIS サーバと組み合わせてマスターやスレーブサーバとして使うことはできません。
幸いなことに、現在、そのようなサーバが使われていることは ほとんどないでしょう。

20.9.10. NIS クライアントとしても動作している NIS サーバ

複数のサーバが存在し、サーバ自身が NIS クライアントでもあるようなドメインで ypserv が実行される場合には注意が必要です。 一般的に良いとされているのは、
他のサーバと結合をつくるようにブロードキャストさせるのではなく、
サーバをそれ自身に結合させることです。
もし、サーバ同士が依存関係を持っていて、一つのサーバが停止すると、
奇妙なサービス不能状態に陥ることがあります。
その結果、すべてのクライアントはタイムアウトを起こして 他のサーバに結合しようと試みますが、

これにかかる時間はかなり大きく、 サーバ同士がまた互いに結合してしまったりすると、
サービス不能状態はさらに継続することになります。

`ypbind` に `-S` オプションフラグを指定して実行することで、
ホストを特定のサーバに結合することが可能です。 NIS
サーバを再起動するたびに、これを手動で行いたくないなら、 次の行を /etc/rc.conf
に追加すればよいでしょう。

```
nis_client_enable="YES" # run client stuff as well
nis_client_flags="-S NIS domain,server"
```

詳細については ypbind(8) を参照してください。

20.9.11. パスワード形式

NIS を実装しようする人の誰もがぶつかる問題の一つに、 パスワード形式の互換性があります。 NIS
サーバが DES 暗号化パスワード使っている場合には、 同様に DES
を使用しているクライアントしか対応できません。 たとえば Solaris™; の NIS
クライアントがネットワーク内にある場合、 ほぼ確実に DES
暗号化パスワードを使用しなければならないでしょう。

サーバとクライアントがどのライブラリを使用しているかは、 /etc/login.conf を確認してください。
ホストが DES 暗号パスワードを使用するように設定されている場合、 `default`
クラスには以下のようなエントリが含まれます。

```
default:\
    :passwd_format=des:\
    [Further entries elided]
```

`passwd_format` 特性について他に利用可能な値は `blf` および `md5` (それぞれ Blowfish および MD5
暗号化パスワード) です。

/etc/login.conf を変更したときは、 ログイン特性データベースも再構築しなければなりません。 これは
`root` 権限で下記のようにコマンドを実行すればできます。

```
# cap_mkdb /etc/login.conf
```

すでに /etc/master.passwd 内に記録されているパスワード形式は、
ログイン特性データベースが再構築された後、
ユーザが彼らのパスワードをはじめて変更するまで変更されないでしょう。

次に、 パスワードが選択した形式で暗号化されることを確実にするために、 さらに /etc/auth.conf 内の
`crypt_default` において、
選択したパスワード形式に高い優先順位がついていることも確認してください。
そうするためには、選択した形式をリストの先頭に置いてください。 たとえば DES
暗号化されたパスワードを使用するときは、 エントリは次のようになります。

```
crypt_default   =   des blf md5
```

FreeBSD 上の各 NIS サーバおよびクライアントにおいて上記の手順に従えば、ネットワーク内でどのパスワード形式が使用されるかが
それらのマシン間で整合されているということを確信できます。 NIS クライアント上で問題があれば、ここから問題となりそうな部分を探すと良いでしょう。 覚えておいてください: 異種混在ネットワークに NIS サーバを配置したいときには、 DES が最大公約数的な標準となるでしょうから、すべてのシステムで DES を使用しなければならないかもしれません。

20.10. DHCP

20.10.1. DHCP とは何でしょうか?

DHCP (Dynamic Host Configuration Protocol) は、 システムをネットワークに接続するだけで、ネットワークでの通信に必要な情報を入手することができる仕組みです。 FreeBSD では ISC (Internet Software Consortium) による DHCP の実装を使用しています。したがって、ここでの説明のうち実装によって異なる部分は ISC のもの用になっています。

20.10.2. この節で説明していること

この節は ISC DHCP システムのクライアント側およびサーバ側の構成要素の両方について説明します。クライアント側のプログラムである dhclient は FreeBSD のベースシステム内に含まれています。そして、サーバ側の要素は net/isc-dhcp3-server port から利用可能です。下記の説明の他に、 dhclient(8), dhcp-options(5) および dhclient.conf(5) マニュアルページが役にたつ情報源です。

20.10.3. DHCP の動作

クライアントとなるマシン上で、 DHCP のクライアントである dhclient を実行すると、まず設定情報の要求をブロードキャストします。デフォルトでは、 このリクエストには UDP のポート 68 を使用します。 サーバは UDP のポート 67 で応答し、クライアントの IP アドレスと、ネットマスクやルータ、DNS サーバなどの関連する情報を提供します。 これらの情報のすべては DHCP の "リース" の形で送られ、DHCP サーバ管理者によって決められたある一定の時間内でのみ有効になります。
これによって、ネットワークに存在しなくなったホストの IP アドレスは自動的に回収されることになります。

DHCP クライアントはサーバから非常に多くの情報を取得することができます。 dhcp-options(5) に非常に大きなリストが載っています。

20.10.4. FreeBSD への組み込み

FreeBSD は ISC の DHCP クライアントである dhclient を完全に組み込んでいます。 DHCP クライアントはインストーラと基本システムの両方で提供されています。 ですから DHCP サーバを走らせているネットワーク上ではネットワーク関係の設定についての詳細な知識は必要になりません。 dhclient は、3.2 以降のすべての FreeBSD の配布物に含まれています。

DHCP は sysinstall で対応されており、sysinstall でのネットワークインタフェイス設定の際は、

"このインタフェイスの設定として DHCP を試してみますか? (Do you want to try DHCP configuration of this interface?)" という質問が最初になされます。 これに同意することで dhclient が実行され、それが成功すればネットワークの設定情報は自動的に取得されます。

システム起動時に DHCP を使ってネットワーク情報を取得するようにするには、次の二つを行なう必要があります。

- bpf デバイスがカーネルに組み込まれていることを確認します。これを組み込むには、カーネルコンフィグレーションファイルに pseudo-device bpf という行を追加し、カーネルを再構築します。 カーネルの構築に関する詳細は、FreeBSD カーネルのコンフィグレーション を参照してください。

  bpf デバイスは、FreeBSD にはじめから用意されている GENERIC カーネルに組み込まれていますので、
  自分で設定を変えたカスタムカーネルを使っているのでなければ、DHCP を動作させるためにカーネルを再構築する必要はありません。

  セキュリティに関心のある方向けに注意しておきます。bpf デバイスは、パケットスニファ (盗聴プログラム) を動作させることができる (ただし root 権限が必要) デバイスです。 bpf は DHCP を動作させるためにかならず必要ですが、セキュリティが非常に重要な場面では DHCP をいつか使うかもしれないというだけで bpf デバイスをカーネルに追加すべきではないでしょう。

- /etc/rc.conf を編集して、 次の行を追加してください。

```
ifconfig_fxp0="DHCP"
```

  で説明されているように fxp0 の部分を、動的に設定したいインタフェースの名前で置き換えることを忘れないようにしてください。

  もし、使っている dhclient の場所を変更していたり、dhclient にフラグを渡したい場合は、同様に下のように書き加えてください。

```
dhcp_program="/sbin/dhclient"
dhcp_flags=""
```

DHCP サーバ dhcpd は、Ports Collection に net/isc-dhcp3-server の一部として収録されています。 この port には ISC DHCP サーバと文書が含まれています。

20.10.5. 関連ファイル

- /etc/dhclient.conf

  dhclient は設定ファイル /etc/dhclient.conf を必要とします。大抵の場合、このファイルはコメントだけであり、
  デフォルトが通常使いやすい設定になっています。 この設定ファイルは dhclient.conf(5) マニュアルページで説明しています。

- /sbin/dhclient

  `dhclient` は静的にリンクされており、 /sbin に置かれています。dhclient(8) マニュアルページで `dhclient` コマンドについてより詳しく説明しています。

- /sbin/dhclient-script

  `dhclient-script` は FreeBSD 特有の、 DHCP クライアント設定スクリプトです。これについては dhclient-script(8) マニュアルページで説明されていますが、これを編集する必要はほとんど発生しないでしょう。

- /var/db/dhclient.leases

  DHCP クライアントはこのファイルに有効なリースのデータベースをログとして記録します。dhclient.leases(5) にもうすこし詳しい解説があります。

20.10.6. 参考になる文献

DHCP のプロトコルは RFC 2131 に完全に記述されています。また dhcp.org にも有用な情報源が用意されています。

20.10.7. DHCP サーバのインストールと設定

この節で説明していること

この節は DHCP の ISC (Internet Software Consortium) 実装を用いて FreeBSD システムを DHCP サーバとして動作させる方法の情報を提供します。

DHCP のサーバ部分は FreeBSD の一部として提供されません。したがって、このサービスを提供するために net/isc-dhcp3-server port をインストールする必要があるでしょう。 Ports Collection を使用する情報についての詳細は アプリケーションのインストール - packages と ports を参照してください。

DHCP サーバのインストール

FreeBSD システムを DHCP サーバとして設定するために、bpf(4) デバイスがカーネルに組み込まれていることを保証する必要があります。
そうするためには、カーネルコンフィギュレーションファイルに `pseudo-device bpf` を追加して、カーネルを再構築してください。 カーネルの構築に関する詳細は FreeBSD カーネルのコンフィグレーション を参照してください。

bpf デバイスは、 FreeBSD にはじめから用意されている GENERIC カーネルの一部なので、DHCP を動作させるためにカスタムカーネルを作成する必要はありません。

セキュリティを特に意識する人は、bpf bpf はパケットスニファ (盗聴プログラム) が正常に (このようなプログラムはさらに特権アクセスを必要としますが) 動作することを可能にするデバイスでもあることに注意してください。 bpf は DHCP を使用するために必要 です。 しかし、セキュリティをとても気にしているなら、DHCP をいつか使うかもしれないというだけで bpf デバイスをカーネルに含めるべきではないでしょう。

次に行わねばならないのは、 net/isc-dhcp3-server port によってインストールされた dhcpd.conf

のサンプルを編集することです。 デフォルトでは、これは /usr/local/etc/dhcpd.conf.sample で、編集する前にこれを /usr/local/etc/dhcpd.conf にコピーするべきでしょう。

DHCP サーバの設定

dhcpd.conf はサブネットおよびホストに関する宣言で構成されます。
例を使って説明するのが最も簡単でしょう。

```
option domain-name "example.com";①
option domain-name-servers 192.168.4.100;②
option subnet-mask 255.255.255.0;③

default-lease-time 3600;④
max-lease-time 86400;⑤
ddns-update-style none;⑥

subnet 192.168.4.0 netmask 255.255.255.0 {
  range 192.168.4.129 192.168.4.254;⑦
  option routers 192.168.4.1;⑧
}

host mailhost {
  hardware ethernet 02:03:04:05:06:07;⑨
  fixed-address mailhost.example.com;⑩
}
```

① このオプションは、 デフォルト探索ドメインとしてクライアントに渡されるドメインを指定します。
これが意味するところの詳細については resolv.conf(5) を参照してください。

② このオプションはクライアントが使用する、 コンマで区切られた DNS サーバのリストを指定します。

③ クライアントに渡されるネットマスクです。

④ クライアントは特定のリース期限を要求することもできます。
それ以外の場合は、サーバはこのリース期限値 (秒) でリースを割り当てるでしょう。

⑤ これはサーバがリースする時間の最大値です。 クライアントがこれより長いリースを要求しても、
max-lease-time 秒だけしか有効にならないでしょう。

⑥ このオプションは、リースが受理、またはリリースされたときに DHCP サーバが DNS を更新しようとするかどうかを指定します。 ISC 実装では、このオプションは 必須 です。

⑦ これはどの範囲の IP アドレスが、
クライアントに割り当てるために予約されたプールに使用されるかを示します。
この範囲に含まれている IP アドレスはクライアントに渡されます。

⑧ クライアントに供給されるデフォルトゲートウェイを宣言します。

⑨ (リクエストが生じた時に DHCP サーバがホストを認識できるように) ホストのハードウェア MAC アドレスを指定します。

⑩ ホストに常に同じ IP アドレスを付与することを指定します。 DHCP
サーバはリース情報を返す前にホスト名の名前解決をするので、
ここにホスト名を書いても構いません。

dhcpd.conf を書き終えたら以下のコマンドでサーバを起動できます。

```
# /usr/local/etc/rc.d/isc-dhcpd.sh start
```

今後サーバの設定に変更を加える必要が生じた時には、 SIGHUP シグナルを dhcpd に送っても、多くのデーモンがそうであるようには、 設定ファイルが再読み込み されない ことに注意してください。SIGTERM シグナルを送ってプロセスを停止し、それから上記のコマンドを用いて再起動させる必要があります。

ファイル

- /usr/local/sbin/dhcpd

  dhcpd は静的にリンクされ /usr/local/sbin に置かれます。 dhcpd に関するそれ以上の情報は port とともにインストールされる dhcpd(8) マニュアルページにあります。

- /usr/local/etc/dhcpd.conf

  dhcpd はクライアントへのサービス提供をはじめる前に設定ファイル /usr/local/etc/dhcpd.conf を必要とします。このファイルは、サーバの稼働に関する情報に加えて、サービスされているクライアントに提供される情報のすべてを含む必要があります。この設定ファイルについての詳細は、 port によってインストールされる dhcpd.conf(5) マニュアルページを参照してください。

- /var/db/dhcpd.leases

  DHCP サーバは発行したリースのデータベースをこのファイルにログとして保持します。 port によってインストールされる dhcpd.leases(5) にはもう少し詳しい説明があります。

- /usr/local/sbin/dhcrelay

  dhcrelay は、DHCP サーバがクライアントからのリクエストを、 別のネットワーク上にある DHCP サーバに転送する高度な環境下で使用されます。 この機能が必要なら、net/isc-dhcp3-server port をインストールしてください。 port とともに提供される dhcrelay(8) マニュアルページにはより詳細な情報が含まれます。

20.11. DNS

20.11.1. 概観

FreeBSD はデフォルトでは DNS プロトコルの最も一般的な実装である BIND (Berkeley Internet Name Domain) を使用します。DNS はホスト名を IP アドレスに、そして IP アドレスをホスト名に関連づけるプロトコルです。 たとえば www.FreeBSD.org に対する問い合わせは The FreeBSD Project の ウェブサーバの IP アドレスを受け取るでしょう。 その一方で ftp.FreeBSD.org に対する問い合わせは、 対応する FTP マシンの IP アドレスを返すでしょう。同様に、その逆のことも可能です。 IP アドレスに対する問い合わせを行うことで、そのホスト名を解決することができます。 DNS 検索を実行するために、システム上でネームサーバを動作させる必要はありません。

DNS は、 個々のドメイン情報を格納およびキャッシュした、

権威のあるルートサーバおよび他の小規模なネームサーバによる多少複雑なシステムによって、
インターネット全体にわたって協調して動作します。

この文書は FreeBSD で安定版として利用されている BIND 8.x について説明します。 FreeBSD では BIND 9.x を net/bind9 port からインストールできます。

RFC1034 および RFC1035 は DNS プロトコルを定義しています。

現在のところ BIND は Internet Software Consortium (www.isc.org) によって保守されています。

20.11.2. 用語

この文書を理解するには DNS 関連の用語をいくつか理解しなければいけません。

用語	定義
正引き DNS	ホスト名から IP アドレスへの対応です。
オリジン (origine)	特定のゾーンファイルによってカバーされるドメインへの参照です。
named, BIND, ネームサーバ	FreeBSD 内の BIND ネームサーバパッケージの一般名称です。
リゾルバ (resolver)	マシンがゾーン情報についてネームサーバに問い合わせるシステムプロセスです。
逆引き DNS	正引き DNS の逆です。つまり IP アドレスからホスト名への対応です。
ルートゾーン	インターネットゾーン階層の起点です。 すべてのゾーンはルートゾーンの下に属します。 これはファイルシステムのすべてのファイルがルートディレクトリの下に属することと似ています。
ゾーン	同じ権威によって管理される個々の DNS ドメイン、 DNS サブドメイン、あるいは DNS の一部分です。

ゾーンの例:

- `.` はルートゾーンです。
- `org.` はルートゾーンの下のゾーンです。
- `example.org` は `org.` ゾーンの下のゾーンです。
- `foo.example.org.` はサブドメインで、 `example.org.` の下のゾーンです。
- `1.2.3.in-addr.arpa` は 3.2.1.* の IP 空間に含まれるすべての IP アドレスを参照するゾーンです。

見て分かるように、ホスト名のより詳細な部分はその左側に現れます。 たとえば `example.org.` は `org.` より限定的です。同様に `org.` はルートゾーンより限定的です。
ホスト名の各部分のレイアウトはファイルシステムに非常に似ています。 たとえば /dev はルートの下であることなどです。

20.11.3. ネームサーバを実行する理由

ネームサーバは通常二つの形式があります: 権威のあるネームサーバとキャッシュネームサーバです。

権威のあるネームサーバは以下の場合に必要です。

- 問い合わせに対して信頼できる返答をすることで、 ある人が DNS 情報を世界に向けて発信したいとき。
- example.org といったドメインが登録されており、 その下にあるホスト名に IP アドレスを割り当てる必要があるとき。
- IP アドレスブロックが (IP からホスト名への) 逆引き DNS エントリを必要とするとき。
- プライマリサーバがダウンしているかまたはアクセスできない場合に、 代わりに問い合わせに対してスレーブと呼ばれるバックアップネームサーバが返答しなければならないとき。

キャッシュネームサーバは以下の場合に必要です。

- ローカルのネームサーバが、 外部のネームサーバに問い合わせするよりも、 キャッシュしてより速く返答できるとき。
- ネットワークトラフィックの総量を減らしたいとき (DNS のトラフィックはインターネットトラフィック全体の 5% 以上を占めることが測定されています)

www.FreeBSD.org に対する問い合わせを発したとき、 リゾルバは大体の場合上流の ISP のネームサーバに問い合わせをして返答を得ます。 ローカルのキャッシュ DNS サーバがあれば、 問い合わせはキャッシュ DNS サーバによって外部に対して一度だけ発せられます。 情報がローカルに蓄えられるので、
追加の問い合わせはいずれもローカルネットワークの外側にまで確認しなくてもよくなります。

20.11.4. 動作のしくみ

FreeBSD では BIND デーモンは自明な理由から named と呼ばれます。

ファイル	説明
named	BIND デーモン
ndc	ネームデーモンコントロールプログラム
/etc/namedb	BIND のゾーン情報が置かれるディレクトリ
/etc/namedb/named.conf	デーモンの設定ファイル

ゾーンファイルは通常 /etc/namedb ディレクトリ内に含まれており、ネームサーバによって処理される DNS ゾーン情報を含んでいます。

20.11.5. BIND の起動

BIND はデフォルトでインストールされているので、 すべてを設定することは比較的単純です。

named デーモンが起動時に開始されることを保証するには、 /etc/rc.conf に以下の変更をいれてください。

```
named_enable="YES"
```

デーモンを手動で起動するためには (設定をした後で)

```
# ndc start
```

20.11.6. 設定ファイル

`make-localhost` の利用

次のコマンドが

```
# cd /etc/namedb
# sh make-localhost
```

ローカル逆引き DNS ゾーンファイルを /etc/namedb/localhost.rev に適切に作成することを確認してください。

/etc/namedb/named.conf

```
// $FreeBSD$
//
// 詳細については named(8) マニュアルページを参照してください。プライマリサーバ
// を設定するつもりなら、DNS がどのように動作するかの詳細を確実に理解してくださ
// い。単純な間違いであっても、影響をうける相手に対する接続を壊したり、無駄な
// インターネットトラフィックを大量に引き起こし得ます。

options {
        directory "/etc/namedb";

// "forwarders" 節に加えて次の行を有効にすることで、ネームサーバに決して自発的
// に問い合わせを発せず、常にそのフォワーダにたいして尋ねるように強制すること
// ができます:
//
//      forward only;

// あなたが上流のプロバイダ周辺の DNS サーバを利用できる場合、その IP アドレス
// をここに入力し、下記の行を有効にしてください。こうすれば、そのキャッシュの
// 恩恵にあやかることができ、インターネット全体の DNS トラフィックが減るでしょう。
/*
        forwarders {
                127.0.0.1;
        };
*/
```

コメントが言っている通り、上流のキャッシュの恩恵を受けるために `forwarders`

をここで有効にすることができます。
通常の状況では、ネームサーバはインターネットの特定のネームサーバを調べて、
探している返答を見つけるまで再帰的に問い合わせを行います。
これが有効になっていれば、まず上流のネームサーバ (または 与えられたネームサーバ) に問い合わせて、
そのキャッシュを利用するでしょう。　　問い合わせをする上流のネームサーバが極度に通信量が多く、
高速であった場合、これを有効にする価値があるかもしれません。

ここに　127.0.0.1　を指定しても動作　しません。　上流のネームサーバの　IP アドレスに変更してください。

```
        /*
         * あなたと利用したいネームサーバとの間にファイアウォールがある場合、
         * 下記の quiery-source 指令を有効にする必要があるでしょう。
         * 過去の BIND のバージョンは常に 53 番ポートに問い合わせをしますが、
         * BIND 8.1 はデフォルトで非特権ポートを使用します。
         */
        // query-source address * port 53;

        /*
         * 砂場内で動作させている場合、ダンプファイルのために異なる場所を指定
         * しなければならないかもしれません。
         */
        // dump-file "s/named_dump.db";
};

// 注意: 下記は将来のリリースで対応されるでしょう。
/*
host { any; } {
        topology {
                127.0.0.0/8;
        };
};
*/

// セカンダリを設定することはより簡単な方法で、そのおおまかな姿が下記で説明さ
// れています。
//
// ローカルネームサーバを有効にする場合、このサーバが最初に尋ねられるように
// /etc/resolv.conf に 127.0.0.1 を入力することを忘れないでください。さらに、
// /etc/rc.conf 内で有効にすることも確認してください。

zone "." {
        type hint;
        file "named.root";
};

zone "0.0.127.IN-ADDR.ARPA" {
        type master;
        file "localhost.rev";
};
```

```
zone
"0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.IP6.INT" {
        type master;
        file "localhost.rev";
};

// 注意: 下記の IP アドレスを使用しないでください。これはダミーでありデモや文書
// だけを目的としたものです。
//
// セカンダリ設定の例です。少なくともあなたのドメインが属するゾーンに対するセカ
// ンダリになることは便利かもしれません。プライマリの責を負っている IP アドレス
// をネットワーク管理者に尋ねてください。
//
// 逆引き参照ゾーン（IN-ADDR.ARPA）を含めることを決して忘れないでください!
// (これは ".IN-ADDR.ARPA" を付け加えられたそれぞれの IP アドレスの最初のバイト
// の逆順です。)
//
// プライマリゾーンの設定をはじめる前に DNS および BIND がどのように動作するか
// 完全に理解してください。時々自明でない落し穴があります。それに比べるとセカン
// ダリを設定するのは単純です。
//
// 注意: 下記の例を鵜呑みにして有効にしないでください。:-) 実際の名前とアドレス
// を代わりに使用してください。
//
// 注意!!! FreeBSD は bind を砂場のなかで動かします (rc.conf 内の named_flags
// を参照してください)。セカンダリゾーンを含んだディレクトリは、bind によって
// 書き込み可能でなければなりません。次の手順が推奨されます:
//
//      mkdir /etc/namedb/s
//      chown bind:bind /etc/namedb/s
//      chmod 750 /etc/namedb/s
```

BIND を砂場 (sandbox) で (訳注: chroot をもちいて) 動作させるための詳細は 砂場で named を実行する を参照してください。

```
/*
zone "example.com" {
        type slave;
        file "s/example.com.bak";
        masters {
                192.168.1.1;
        };
};

zone "0.168.192.in-addr.arpa" {
        type slave;
        file "s/0.168.192.in-addr.arpa.bak";
        masters {
                192.168.1.1;
```

```
        };
};
*/
```

named.conf の中で、 上記は転送と逆引きゾーンのためのスレーブエントリの例です。

新しくサービスするそれぞれのゾーンについて、新規のエントリを named.conf に加えなければいけません。

たとえば example.org に対する最もシンプルなゾーンエントリは以下のようになります。

```
zone "example.org" {
    type master;
    file "example.org";
};
```

このゾーンは type 命令で示されているようにマスタで、ゾーン情報を file 命令で指示された /etc/namedb/example.org ファイルに保持しています。

```
zone "example.org" {
    type slave;
    file "example.org";
};
```

スレーブの場合、 ゾーン情報は特定のゾーンのマスタネームサーバから転送され、指定されたファイルに保存されます。 マスタサーバが停止するか到達できない場合には、スレーブサーバが転送されたゾーン情報を保持していて、 サービスできるでしょう。

ゾーンファイル

example.org に対するマスタゾーンファイル (/etc/namedb/example.org に保持されます) の例は以下のようになります。

```
$TTL 3600

example.org. IN SOA ns1.example.org. admin.example.org. (
                        5               ; Serial
                        10800           ; Refresh
                        3600            ; Retry
                        604800          ; Expire
                        86400 )         ; Minimum TTL

; DNS Servers
@       IN NS           ns1.example.org.
@       IN NS           ns2.example.org.

; Machine Names
localhost       IN A    127.0.0.1
```

```
ns1             IN A    3.2.1.2
ns2             IN A    3.2.1.3
mail            IN A    3.2.1.10
@               IN A    3.2.1.30

; Aliases
www             IN CNAME        @

; MX Record
@               IN MX   10      mail.example.org.
```

"." が最後についているすべてのホスト名は正確なホスト名であり、一方で "." で終了しないすべての行はオリジンが参照されることに注意してください。 たとえば `www` は `www + オリジン` に展開されます。この架空のゾーンファイルでは、 オリジンは `example.org.` なので `www` は `www.example.org.` に展開されます。

ゾーンファイルの書式は次のとおりです。

```
recordname      IN recordtype   value
```

DNS レコードに使われる最も一般的なものは以下のとおりです。

SOA
: ゾーン権威の起点

NS
: 権威のあるネームサーバ

A
: ホストのアドレス

CNAME
: 別名としての正規の名称

MX
: メールエクスチェンジャ

PTR
: ドメインネームポインタ (逆引き DNS で使用されます)

```
example.org. IN SOA ns1.example.org. admin.example.org. (
                        5               ; Serial
                        10800           ; Refresh after 3 hours
                        3600            ; Retry after 1 hour
                        604800          ; Expire after 1 week
                        86400 )         ; Minimum TTL of 1 day
```

`example.org.`
: このゾーンのオリジンでもあるドメイン名

`ns1.example.org.`
: このゾーンに対して権威のあるプライマリネームサーバ

`admin.example.org.`
: このゾーンの責任者。@ を置き換えた電子メールアドレスを指定します。 (admin@example.org は admin.example.org になります)

`5`
: ファイルのシリアル番号です。 これはファイルが変更されるたびに増加させる必要があります。現在では多くの管理者は yyyymmddrr という形式をシリアル番号として使用することを好みます。2001041002 は最後に修正されたのが 2001/04/10 で、後ろの 02 はその日で二回目に修正されたものであるということを意味するでしょう。 シリアル番号は、それが更新されたときにスレーブネームサーバに対してゾーンを通知するので重要です。

```
@       IN NS           ns1.example.org.
```

これは NS エントリです。 このゾーンに対して権威のある返答を返すネームサーバはすべて、このエントリを一つ有していなければなりません。 ここにある @ は example.org. を意味します。 @ はオリジンに展開されます。

```
localhost       IN A    127.0.0.1
ns1             IN A    3.2.1.2
ns2             IN A    3.2.1.3
mail            IN A    3.2.1.10
@               IN A    3.2.1.30
```

A レコードはマシン名を示します。 上記のように ns1.example.org は 3.2.1.2 に結びつけられるでしょう。 ふたたびオリジンを示す @ がここに使用されていますが、これは example.org が 3.2.1.30 に結び付けられることを意味しています。

```
www             IN CNAME        @
```

CNAME レコードは通常マシンに別名を与えるときに使用されます。 例では www はオリジン、すなわち example.org (3.2.1.30) のアドレスをふられたマシンへの別名を与えます。 CNAME はホスト名の別名、または複数のマシン間で一つのホスト名をラウンドロビン (訳注: 問い合わせがあるたびに別の IP アドレスを返すことで、 一台にアクセスが集中することを防ぐ手法) するときに用いられます。

```
@               IN MX   10      mail.example.org.
```

MX レコードは、ゾーンに対してどのメールサーバがやってきたメールを扱うことに責任を持っているかを示します。mail.example.org はメールサーバのホスト名で、10 はメールサーバの優先度を示します。

優先度が 3,2 または 1 などのメールサーバをいくつも置くことができます。 example.org へ送ろうとしているメールサーバははじめに一番優先度の高いメールサーバに接続しようとします。
そして接続できない場合、二番目に優先度の高いサーバに接続しようとし、
以下、メールが適切に配送されるまで同様に繰り返します。

in-addr.arpa ゾーンファイル (逆引き DNS) に対しても A または CNAME の代わりに PTR エントリが用いられることを除けば、 同じ書式が使われます。

```
$TTL 3600

1.2.3.in-addr.arpa. IN SOA ns1.example.org. admin.example.org. (
                        5               ; Serial
                        10800           ; Refresh
                        3600            ; Retry
                        604800          ; Expire
                        3600 )          ; Minimum

@       IN NS   ns1.example.org.
@       IN NS   ns2.example.org.

2       IN PTR  ns1.example.org.
3       IN PTR  ns2.example.org.
10      IN PTR  mail.example.org.
30      IN PTR  example.org.
```

このファイルは上記の架空のドメインの IP アドレスからホスト名への対応を与えます。

20.11.7. キャッシュネームサーバ

キャッシュネームサーバはどのゾーンに対しても権威をもたないネームサーバです。
キャッシュネームサーバは単に自分で問い合わせをし、
後で使えるように問い合わせの結果を覚えておきます。 これを設定するには、ゾーンを何も含まずに、通常通りネームサーバを設定してください。

20.11.8. 砂場で named を実行する

セキュリティを強めるために named(8) を非特権ユーザで実行し、 砂場のディレクトリ内に chroot(8) して実行したいと思うかもしれません。 こうすると named デーモンは砂場の外にはまったく手を出すことができません。 named が乗っ取られたとしても、これによって起こりうる損害が小さくなるでしょう。 FreeBSD にはデフォルトで、そのための bind というユーザとグループがあります。

多くの人々は named を chroot するように設定する代わりに、 jail(8) 環境内で named を実行することを奨めるでしょう。 この節ではそれは扱いません。

named は砂場の外 (共有ライブラリ、ログソケットなど) にアクセスできないので、 named を正しく動作させるためにいくつもの段階を経る必要があります。
下記のチェックリストにおいては、砂場のパスは /etc/namedb で、
このディレクトリの内容には何も手を加えていないと仮定します。 root

権限で次のステップを実行してください。

- named が存在することを期待しているディレクトリをすべて作成します。

```
# cd /etc/namedb
# mkdir -p bin dev etc var/tmp var/run master slave
# chown bind:bind slave var/* ①
```

① これらのディレクトリに対して named が必要なのは書き込み権限だけなので、それだけを与えます。

- 基本ゾーンファイルと設定ファイルの編集と作成を行います。

```
# cp /etc/localtime etc ①
# mv named.conf etc && ln -sf etc/named.conf
# mv named.root master

# sh make-localhost && mv localhost.rev localhost-v6.rev master
# cat > master/named.localhost
$ORIGIN localhost.
$TTL 6h
@   IN  SOA localhost. postmaster.localhost. (
            1   ; serial
            3600    ; refresh
            1800    ; retry
            604800  ; expiration
            3600 )  ; minimum
    IN  NS  localhost.
    IN  A       127.0.0.1
^D
```

① これは named が syslogd(8) に正しい時刻でログを書き込むことを可能にします。

- 4.9-RELEASE より前のバージョンの FreeBSD を使用している場合、 静的リンクされた named-xfer を構築し、砂場にコピーしてください。

```
# cd /usr/src/lib/libisc
# make cleandir && make cleandir && make depend && make all
# cd /usr/src/lib/libbind
# make cleandir && make cleandir && make depend && make all
# cd /usr/src/libexec/named-xfer
# make cleandir && make cleandir && make depend && make NOSHARED=yes all
# cp named-xfer /etc/namedb/bin && chmod 555 /etc/namedb/bin/named-xfer①
```

静的リンクされた named-xfer をインストールしたら、ソースツリーの中にライブラリまたはプログラムの古くなったコピーを残さないように、掃除する必要があります。

```
# cd /usr/src/lib/libisc
# make cleandir
# cd /usr/src/lib/libbind
# make cleandir
# cd /usr/src/libexec/named-xfer
# make cleandir
```

① このステップは時々失敗することが報告されています。
もし失敗した場合、次のコマンドを実行してください。そして /usr/obj ツリーを削除します。これはソースツリーからすべての "がらくた" を一掃します。
もう一度上記の手順を行うと、今度はうまく動作するでしょう。
バージョン 4.9-RELEASE 以降の FreeBSD を使用している場合 /usr/libexec にある `named-xfer` のコピーはデフォルトで静的リンクされています。 砂場にコピーするために単純に cp(1) が使えます。

- named が見ることができ、 書き込むことのできる dev/null を作成します。

```
# cd /etc/namedb/dev && mknod null c 2 2
# chmod 666 null
```

- /etc/namedb/var/run/ndc から /var/run/ndc へのシンボリックリンクを作成します。

```
# ln -sf /etc/namedb/var/run/ndc /var/run/ndc
```

これは単に ndc(8) を実行するたびに `-c` オプションを指定しなくてもよいようにするだけです。 /var/run の中身は起動時に削除されるため、
これが有用だと思うなら、このコマンドをルートの crontab に `@reboot` オプションを指定して追加してください。 詳細については crontab(5) を参照してください。

- named が書き込める追加の log ソケットを作成するように syslogd(8) を設定します。これを行うためには、/etc/rc.conf 内の `syslogd_flags` 変数に `-l /etc/namedb/dev/log` を加えてください。
- 次の行を /etc/rc.conf に加えて named が起動し、 自身を砂場内に `chroot` するように調整します

```
named_enable="YES"
named_flags="-u bind -g bind -t /etc/namedb /etc/named.conf"
```

設定ファイル */etc/named.conf* は 砂場のディレクトリに対して相対的なフルパスで表されることに注意してください。
つまり、上記の行で示されたファイルは実際には /etc/namedb/etc/named.conf です。

次のステップは named がどのゾーンを読み込むか、そしてディスク上のどこにゾーンファイルがあるのかを知るために /etc/namedb/etc/named.conf を編集することです。下記に例をコメントを加えて示します (ここで特にコメントされていない内容については、砂場の中で動作させない DNS サーバの設定と同じです)。

```
options {
        directory "/";①
        named-xfer "/bin/named-xfer";②
        version "";     // Don't reveal BIND version
        query-source address * port 53;
};
// ndc control socket
controls {
        unix "/var/run/ndc" perm 0600 owner 0 group 0;
};
// Zones follow:
zone "localhost" IN {
        type master;
        file "master/named.localhost";③
        allow-transfer { localhost; };
        notify no;
};
zone "0.0.127.in-addr.arpa" IN {
        type master;
        file "master/localhost.rev";
        allow-transfer { localhost; };
        notify no;
};
zone "0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.ip6.int" {
    type master;
    file "master/localhost-v6.rev";
    allow-transfer { localhost; };
    notify no;
};
zone "." IN {
        type hint;
        file "master/named.root";
};
zone "private.example.net" in {
        type master;
        file "master/private.example.net.db";
    allow-transfer { 192.168.10.0/24; };
};
zone "10.168.192.in-addr.arpa" in {
        type slave;
        masters { 192.168.10.2; };
        file "slave/192.168.10.db";④
};
```

① `directory` は / を指定します。 named が必要とするファイルはすべてこのディレクトリにあります。(この指定は "通常の" (訳注: 砂場内で動作させない) ユーザにとっての /etc/namedb と等価です)。

② `named-xfer` バイナリへの (named にとっての) フルパスを指定します。 named はデフォルトで `named-xfer` を /usr/libexec から探すようにコンパイルされているので、これが必要です

③ このゾーンに対するゾーンファイルを named が見つけられるようにファイル名を (上記と同様に `directory` からの相対パスで) 指定します。

④ このゾーンに対するゾーン情報がマスタサーバからが転送されたあとに、named がゾーンファイルのコピーを書き込むファイル名を (上記と同様に `directory` からの相対パスで) 指定します。これが、上記のように設定段階で slave ディレクトリの所有者を `bind` に変更する理由です。

上記のステップを完了したら、サーバを再起動するか syslogd(8) を再起動し、named(8) を起動してください。その際、`syslogd_flags` および `named_flags` に新たに指定したオプションが有効になっていることを確かめてください。これで named を砂場のなかで動作させることができているはずです!

20.11.9. セキュリティ

BIND は DNS の最も一般的な実装ではありますが、常にセキュリティ問題を抱えています。問題になり得る、また悪用可能なセキュリティホールが時々みつかります。

現在のインターネットおよび FreeBSD のセキュリティ問題について常に最新の情報を得るために CERT および freebsd-security-notifications を購読するとよいでしょう。

問題が生じたとしても、最新のソースからビルドした named を用意しておけば、問題にならないかもしれません。

20.11.10. さらなる情報源

BIND/named のマニュアルページ: ndc(8) named(8) named.conf(8)

- ISC Bind 公式ページ
- BIND FAQ
- O'Reilly DNS and BIND 4th Edition
- RFC1034 - Domain Names - Concepts and Facilities (ドメイン名、その概念と基盤)
- RFC1035 - Domain Names - Implementation and Specification (ドメイン名、その実装と仕様)

20.12. NTP

20.12.1. 概説

時間の経過とともに、コンピュータの時計はずれてしまいがちです。
時間が経つと、コンピュータの時計は正確でなくなってゆきます。 NTP (Network Time Protocol) は時計が正確であることを保証する方法の一つです。

インターネットサービスの多くは、コンピュータの時計が正確であることに依存しているか、

あるいは多くを負っています。 たとえば web サーバ は、あるファイルがある時刻以降に修正されていたらそのファイルを送ってほしいという要求を受け取るかもしれません。 cron(8) のようなサービスは所定の時間にコマンドを実行します。 時計が正確でない場合、これらのコマンドは期待したとおりには実行されないかもしれません。

FreeBSD は ntpd(8) NTP サーバを搭載しています。これは、 マシンの時計を合わせるために他の NTP サーバに問い合わせをしたり、 他のマシンに対して時刻を報じるために使用できます。

20.12.2. 適切な NTP サーバの選択

時刻を同期するために利用する NTP サーバを、 一つ以上見つける必要があります。ネットワーク管理者、または ISP はこの目的のために NTP サーバを設定しているかもしれません - 本当にそうなのか確かめるためにドキュメントを確認してください。 あなたの近くの NTP サーバを探せる 公にアクセス可能な NTP サーバのリスト があります。どのサーバを選択するとしても、そのサーバの運営ポリシを理解し、要求されているなら利用許可を求めることを忘れないでください。

使用しているサーバのうちのどれかが到達不能になるか、 その時計の信頼性が低い場合、無関係の NTP サーバをいくつか選択するとよいでしょう。 ntpd(8) は他のサーバから受け取った応答を賢く利用します - 信頼できないサーバより信頼できるサーバを重視します。

20.12.3. マシンの設定

基本設定

マシンが起動するときだけ時計を同期させたい場合は ntpdate(8) が使えます。頻繁に再起動され、たまに同期すれば十分なデスクトップマシンには適切かもしれません。 しかしほとんどのマシンでは ntpd(8) を実行するべきです。

ntpd(8) を動かしているマシンでも、起動時に ntpdate(8) を使用するのはよい考えです。 ntpd(8) プログラムは時計を徐々に変更します。しかし ntpdate(8) は正しい時刻と現在設定されているマシンの時刻がどんなに離れていようとも時計を設定します。

起動時に ntpdate(8) を有効にするためには、 `ntpdate_enable="YES"` を /etc/rc.conf に追加してください。 さらに、同期したいすべてのサーバおよび、ntpdate(8) に渡すあらゆるフラグを `ntpdate_flags` に指定する必要があるでしょう。

一般設定

NTP は ntp.conf(5) に記述された書式の /etc/ntp.conf ファイルによって設定されます。簡単な例を以下に示します。

```
server ntplocal.example.com prefer
server timeserver.example.org
server ntp2a.example.net

driftfile /var/db/ntp.drift
```

`server` オプションは、 使用するサーバを一行に一つずつ指定します。サーバが上記の `ntplocal.example.com` のように `prefer` 引数とともに指定された場合、

このサーバは他のサーバより優先されます。 優先されたサーバからの応答は、
他のサーバの応答と著しく異なる場合は破棄されますが、
そうでなければ他の応答を考慮することなく使用されます。 prefer 引数は、通常、
特別な時間モニタハードウェアを備えているような非常に正確であるとされている NTP
サーバに対して使用されます。

driftfile
オプションはシステム時計の周波数オフセットを格納するために使用するファイルを指定します。
ntpd(8) プログラムは、 時計の自然変動を自動的に補正するためにこれを用います。
これにより、一定時間外部の時刻ソースから切り離されたとしても、
十分正確な時刻を維持することを可能にします。

driftfile オプションは、使用している NTP
サーバから過去に受け取った応答に関する情報を格納するために、
どのファイルが使用されるか指定します。 このファイルは NTP に関する内部情報を含んでいます。
これは他のプロセスによって修正されてはいけません。

サーバへのアクセス制御

デフォルトでは NTP サーバはインターネット上のすべてのホストからアクセスが可能です。
/etc/ntp.conf 内で restrict オプションを指定することによって、
どのマシンがサーバにアクセスできるかを制御できるようにします。

NTP サーバにアクセスするマシンのすべてを拒否したいのなら、 以下の行を /etc/ntp.conf
に追加してください。

```
restrict default ignore
```

あなたのネットワーク内のマシンにだけサーバに接続して時計を同期することを認めたいが、
それらからサーバに対して設定を行うのを許さず、
同期する端末としても利用されないようにしたいのなら、 以下を加えてください。

```
restrict 192.168.1.0 mask 255.255.255.0 notrust nomodify notrap
```

192.168.1.0 をあなたのネットワークの IP アドレスに 255.255.255.0
をあなたのネットワークのネットマスクに置き換えてください。

/etc/ntp.conf には複数の restrict オプションを置けます。 詳細に付いては ntp.conf(5) の Access
Control Support サブセクションを参照してください。

20.12.4. NTP サーバの実行

NTP サーバが起動時に実行されることを保証するために、 xntpd_enable="YES" を /etc/rc.conf
に加えてください。 ntpd(8) にフラグを追加したい場合は /etc/rc.conf 内の xntpd_flags
パラメータを編集してください。

マシンを再起動することなくサーバを実行したいときは、 /etc/rc.conf 内の xntpd_flags
で追加されたパラメータをすべて指定して ntpd を実行してください。以下に例を示します。

```
# ntpd -p /var/run/ntpd.pid
```

FreeBSD 5.X では /etc/rc.conf 内のさまざまなオプションの名前が変わりました。
したがって、上記の xntpd に関するオプションは ntpd に置き換えてください。

20.12.5. 一時的なインターネット接続で ntpd を使用する

ntpd(8) プログラムは正しく機能するために、
インターネットへの常時接続を必要としません。しかしながら、
オンデマンドでダイアルアップされるように設定された一時的な接続の場合、 NTP
トラフィックがダイアルを引き起こしたり、
接続を維持し続けるようなことを避けるようにした方がよいでしょう。 ユーザ PPP
を使用している場合、以下の例のように /etc/ppp/ppp.conf 内で filter ディレクティブが使用できます。

```
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
```

詳細については ppp(8) 内の PACKET FILTERING セクション、および /usr/shared/examples/ppp/
内の例を参照してください。

小さい番号のポートをブロックするインターネットアクセスプロバイダでは、
応答があなたのマシンに到達しないので NTP がきちんと動作しない場合もあります。

20.12.6. さらなる情報源

NTP サーバに関する文書は HTML 形式で /usr/shared/doc/ntp/ にあります。

20.13. ネットワークアドレス変換 (NAT)

20.13.1. 概要

一般に natd(8) として知られている FreeBSD ネットワークアドレス変換デーモンは、 raw IP
パケットを受信して、 ソースアドレスをローカルマシンに変更し、 そのパケットを外向きの IP
パケットの流れに再注入するデーモンです。 natd(8) は、
データが戻ってきたときに、データの本来の場所を判別し、
もともと要求した相手へデータを返すことができるようにソース IP アドレスとポートを変更します。

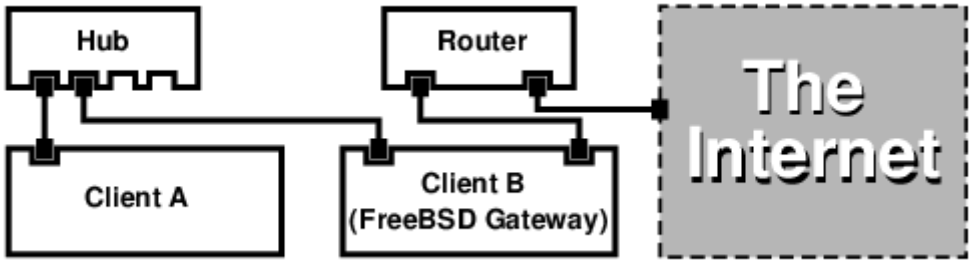
NAT の最も一般的な使用法は、
一般的にはインターネット接続共有として知られているものを実行することです。

20.13.2. 設定

IPv4 の IP 空間が足りなくなりつつあること、および、 ケーブルや DSL のような高速の加入者回線利用者の増加によって、
人々はますますインターネット接続を共有する手段を必要としています。 一つの接続および IP アドレスを通していくつものコンピュータを回線に接続する能力がある natd(8) が合理的な選択になります。

もっともよくあるのは、ユーザが 1 つの IP アドレスでケーブルまたは DSL 回線に接続されたマシンを持っており、 インターネットへのアクセスを LAN 経由でいくつかのコンピュータに提供するのに、
この接続されたコンピュータを使用したいという場合です。

そのためには、インターネットに接続されている FreeBSD マシンはゲートウェイとして動作しなければなりません。 このゲートウェイマシンは 2 つの NIC が必要です (1 つはインターネットルータへ接続するためで、もう 1 つは LAN に接続するためです)。
LAN 上のすべてのマシンはハブまたはスイッチを通して接続されます。



インターネット接続を共有するために、 このような設定がよく使用されています。 LAN 内のマシンの 1 台がインターネットに接続しています。 残りのマシンはその "ゲートウェイ" マシンを通してインターネットにアクセスします。

20.13.3. 設定

次のオプションがカーネルコンフィギュレーションファイルに必要です。

```
options IPFIREWALL
options IPDIVERT
```

さらに、次のオプションを入れてもよいでしょう。

```
options IPFIREWALL_DEFAULT_TO_ACCEPT
options IPFIREWALL_VERBOSE
```

下記の設定を /etc/rc.conf で行わなければなりません。

```
gateway_enable="YES"
firewall_enable="YES"
firewall_type="OPEN"
natd_enable="YES"
```

```
natd_interface="fxp0"
natd_flags=""
```

gateway_enable="YES"	マシンがゲートウェイとして動作するように設定します。 `sysctl net.inet.ip.forwarding=1` コマンドを実行しても同じ効果がえられます。
firewall_enable="YES"	/etc/rc.firewall にあるファイアウォールルールを起動時に有効にします。
firewall_type="OPEN"	これはあらかじめ定義されている、すべてのパケットを通すファイアウォールルールセットを指定します。 他のタイプについては /etc/rc.firewall を参照してください。
natd_interface="fxp0"	パケットを転送するインタフェースを指定します (インターネットに接続されたインタフェース)。
natd_flags=""	起動時に natd(8) に渡される追加の引数

/etc/rc.conf に前述したオプションを定義すると、起動時に `natd -interface fxp0` が実行されます。これは手動でも実行できます。

オプションの定義に natd(8) のコンフィグレーションファイルを使うこともできます。この場合には、/etc/rc.conf に以下の行を追加し、コンフィグレーションファイルを定義してください。

```
natd_flags="-f /etc/natd.conf"
```

/etc/natd.conf ファイルでは、一行ごとにオプションを設定します。たとえば、次節の例では以下のような行を含むファイルを用意してください。

```
redirect_port tcp 192.168.0.2:6667 6667
redirect_port tcp 192.168.0.3:80 80
```

コンフィグレーションファイルに関する、より詳細な情報については、natd(8) マニュアルページの `-f` オプションを調べてください。

LAN にぶら下がっているマシンおよびインタフェースのそれぞれには RFC 1918 で定義されているプライベートネットワーク空間の IP アドレス番号を割り当て、デフォルトゲートウェイアドレスを natd マシンの内側の IP アドレスにすべきです。

たとえば LAN 側のクライアント `A` および `B` は IP アドレス `192.168.0.2` および `192.168.0.3` を割り当てられており、natd マシンの LAN インタフェースは IP アドレス `192.168.0.1` を割り当てられています。クライアント `A` および `B` のデフォルトゲートウェイは natd マシンの `192.168.0.1` に設定されなければなりません。natd マシンの外部、またはインターネットインタフェースは natd(8) の動作に際して特別の修正を必要としません。

20.13.4. ポート転送

natd(8) の短所は、インターネットから LAN 内のクライアントにアクセスできないということです。LAN 内のクライアントは外部に向けて接続を行うことはできますが、入って来るものを受け取ることができません。これは、LAN クライアントのどれかでインターネットサービスを動かそうとした場合に、問題になります。これを何とかする単純な方法は natd マシンから LAN クライアントへ、選択したインターネットポートを転送することです。

たとえばクライアント A で実行されている IRC サーバがあり、クライアント B 上で実行されている web サーバがあるとします。これが正しく動作するには、ポート 6667 (IRC) および 80 (web) への接続を対応するマシンに転送しなければなりません。

-redirect_port に適切なオプションを加えて natd(8) に渡さなければなりません。書式は以下のとおりです。

```
-redirect_port proto targetIP:targetPORT[-targetPORT]
           [aliasIP:]aliasPORT[-aliasPORT]
           [remoteIP[:remotePORT[-remotePORT]]]
```

上記の例では、引数は以下のようにします。

```
-redirect_port tcp 192.168.0.2:6667 6667
-redirect_port tcp 192.168.0.3:80 80
```

これで適切な *tcp* ポートが LAN クライアントマシンに転送されます。

-redirect_port 引数は個々のポートを対応させるポート範囲を示すのに使えます。たとえば *tcp 192.168.0.2:2000-3000 2000-3000* は 2000 番から 3000番ポートに受け取られたすべての接続を、クライアント A 上の 2000 番から 3000 番に転送します。

これらのオプションは natd(8) を直接実行するか、/etc/rc.conf 内の natd_flags="" オプションで設定するか、もしくはコンフィグレーションファイルから渡してください。

設定オプションの詳細については natd(8) をご覧ください。

20.13.5. アドレス転送

複数の IP アドレスが利用可能ですが、それらが 1 台のマシン上になければならないときには、アドレス転送が便利です。これを用いれば natd(8) は LAN クライアントのそれぞれに外部 IP アドレスを割り当てることができます。natd(8) は LAN クライアントから外部へ出て行くパケットを適切な外部の IP アドレスで書き直し、そして特定の IP アドレスに対してやって来るトラフィックのすべてを、指定された LAN クライアントに転送します。これは静的 NAT としても知られています。たとえば 128.1.1.1, 128.1.1.2 および 128.1.1.3 の IP アドレスが、natd ゲートウェイマシンに属しているとします。128.1.1.2 および 128.1.1.3 は LAN クライアントの A および B に転送される一方で、128.1.1.1 は natd ゲートウェイマシンの外部 IP アドレスとして使用することができます。

-redirect_address の書式は以下のとおりです。

```
-redirect_address localIP publicIP
```

localIP	LAN クライアントの内部 IP アドレス
publicIP	LAN クライアントに対応する外部 IP アドレス

上記の例では引数は以下のようになります。

```
-redirect_address 192.168.0.2 128.1.1.2
-redirect_address 192.168.0.3 128.1.1.3
```

`-redirect_port` と同様に、これらの引数は /etc/rc.conf 内の `natd_flags=""` オプションで設定するか、コンフィグレーションファイルから渡すことで指定できます。 アドレス転送では、 特定の IP アドレスで受け取られたデータはすべて転送されるので、 port 転送は必要ありません。

natd マシン上の外部 IP アドレスは、アクティブで外部インタフェースにエイリアスされていなければなりません。 やりかたは rc.conf(5) を参照してください。

20.14. inetd"スーパサーバ"

20.14.1. 概観

inetd(8) は複数のデーモンに対する接続を制御するので、 "インターネットスーパサーバ" と呼ばれます。ネットワークサービスを提供するプログラムは、 一般的にデーモン呼ばれます。inetd は他のデーモンを管理するサーバを努めます。 接続が inetd によって受け付けられると、 inetd は接続がどのデーモンに対するものか判断して、 そのデーモンを起動し、ソケットを渡します。 inetd を 1 つ実行することにより、 それぞれのデーモンをスタンドアロンモードで実行することに比べ、全体としてのシステム負荷を減らします。

基本的に、inetd は他のデーモンを起動するために使用されます。しかし、 chargen, auth および daytime のようなささいなプロトコルは直接扱われます。

この節ではコマンドラインオプションおよび設定ファイル /etc/inetd.conf による inetd の設定の基本を説明します。

20.14.2. 設定

inetd は /etc/rc.conf の仕組によって初期化されます。 デフォルトでは `inetd_enable` オプションは "NO" に設定されています。 しかし多くの場合、sysinstall でセキュリティプロファイルを medium に設定することにより、有効化されます。

```
inetd_enable="YES"
```

または

```
inetd_enable="NO"
```

を /etc/rc.conf に置くことで、起動時に inetd を有効または無効にできます。

さらに inetd_flags オプションによって、いろいろなコマンドラインオプションを inetd に渡すことができます。

20.14.3. コマンドラインオプション

inetd 書式

```
inetd [-d] [-l] [-w] [-W] [-c maximum] [-C rate] [-a address | hostname] [-p filename] [-R rate] [configuration file]
```

-d

デバッグモードにします。

-l

成功した接続のログをとります。

-w

外部サービスに対して TCP Wrapper を有効にします (デフォルト)。

-W

inetd 組み込みの内部サービスに対して TCP Wrapper を有効にします (デフォルト)。

-c maximum

サービス毎に同時に起動可能な最大値のデフォルトを指定します。
デフォルトでは無制限です。サービスごとに指定する max-child パラメータで上書きできます。

-C rate

1 分間にひとつの IP アドレスから起動されるサービスの、最大値のデフォルトを指定します。デフォルトは無制限です。サービスごとに指定する max-connections-per-ip-per-minute パラメータで上書きできます。

-R rate

あるサービスを 1 分間に起動できる最大の数を指定します。デフォルトは 256 です。rate に 0 を指定すると、起動可能な数は無制限になります。

-a

バインドする IP アドレスを一つ指定します。
代わりにホスト名も指定できます。この場合、ホスト名に対応する IPv4 または IPv6 アドレスが使用されます。通常 inetd が jail(8) 内で起動される時点で、ホスト名が指定されます。この場合、ホスト名は jail(8) 環境に対応するものです。

ホスト名指定が使用され、IPv4 および IPv6 両方にバインドしたい場合、/etc/inetd.conf の各サービスに対して、各バインドに対する適切なプロトコルのエントリが必要です。たとえば TCP

ベースのサービスは、 ひとつはプロトコルに "tcp4" を使用し、 もう一つは "tcp6" を使用する、 2つのエントリが必要です。

-p

デフォルトとは異なる PID を保持するファイルを指定します。

/etc/rc.conf 内の inetd_flags オプションを用いて、これらのオプションを inetd に渡すことができます。デフォルトでは inetd_flags は "-wW" に設定されており、 これは inetd の内部および外部サービスに対して TCP wrapper を有効にします。 初心者ユーザはこれらのパラメータを変更する必要は通常ありませんし、 /etc/rc.conf に入力する必要もありません。

外部サービスは、接続を受け取ったときに起動される inetd の外部にあるデーモンで、それに対して、内部サービスは inetd 自身が提供する内部のデーモンです。

20.14.4. inetd.conf

inetd の設定は /etc/inetd.conf ファイルによって制御されます。

/etc/inetd.conf が変更されたときは、 以下のように inetd プロセスに HangUP シグナルを送ることにより、inetd に設定ファイルを再読み込みさせられます。

例 *47. inetd への HangUP シグナル送付*

```
# kill -HUP `cat /var/run/inetd.pid`
```

設定ファイルのそれぞれの行は、 個々のデーモンについての指示になります。 ファイル内のコメントは "#" が先頭につきます。 /etc/inetd.conf の書式は以下のとおりです。

```
service-name
socket-type
protocol
{wait|nowait}[/max-child[/max-connections-per-ip-per-minute]]
user[:group][/login-class]
server-program
server-program-arguments
```

IPv4 を利用する ftpd デーモンのエントリの例です。

```
ftp     stream  tcp     nowait  root    /usr/libexec/ftpd       ftpd -l
```

service-name

これは特定のデーモンのサービス名です。 これは /etc/services 内のサービスリストに対応していなければなりません。 これは inetd がどのポートで受け付けなければならないかを決定します。

新しいサービスが作成された場合、まずはじめに /etc/services 内に記載しなければなりません。

socket-type

`stream`, `dgram`, `raw` または `seqpacket` のどれかを指定します。 `stream` はコネクションに基づいた TCP デーモンに使用しなければならず、 一方で `dgram` は UDP 転送プロトコルを利用したデーモンに対して使用されます。

protocol

次のうちのどれか 1 つを指定します。

プロトコル	説明
tcp, tcp4	TCP IPv4
udp, udp4	UDP IPv4
tcp6	TCP IPv6
udp6	UDP IPv6
tcp46	TCP IPv4 および v6 の両方
udp46	UDP IPv4 および v6 の両方

{wait|nowait}[/max-child[/max-connections-per-ip-per-minute]]

`wait|nowait` は inetd から起動したデーモンが、 自分のソケットを管理できるかどうかを示します。 通常マルチスレッド化されている stream ソケットデーモンは `nowait` を使用するべきである一方、 `dgram` ソケットタイプは wait オプションを使用しなければなりません。 `nowait` は新しいソケット毎に子のデーモンを起動する一方で、 `wait` は通常複数のソケットを 1 つのデーモンに渡します。

inetd が起動できる子のデーモンの最大数は `max-child` オプションで設定できます。 特定のデーモンに対して、起動する数が 10 までという制限が必要な場合、 `nowait` の後に `/10` を置きます。

`max-child` に加えて、他にある 1 つの場所から特定のデーモンへの最大接続数を制限するオプションが利用できます。 `max-connections-per-ip-per-minute` がそれです。ここに 10 を指定すると、特定の IP アドレスからの特定のサービスへの接続を 1 分間につき 10 回に制限します。 これは故意または故意でない資源の浪費および、 マシンへのサービス不能 (DoS) 攻撃を防ぐのに有用です。

`wait` または `nowait` はこの欄に必ず必要です。 `max-child` および `max-connections-per-ip-per-minute` は任意です。

`max-child` または `max-connections-per-ip-per-minute` 制限をかけない stream タイプのマルチスレッドデーモンの設定は `nowait` になります。

作成できる子プロセスの上限が 10 である同じデーモンの設定は `nowait/10` になります。

さらに、 1 分間に IP アドレスあたりの接続制限が 20、 子プロセスの上限が 10 である同じデーモンの設定は `nowait/10/20` になります。

以下のように、これらのオプションはすべて fingerd デーモンのデフォルト設定に使われています。

```
finger stream  tcp     nowait/3/10 nobody /usr/libexec/fingerd fingerd -s
```

user

user はあるデーモンが実行するときのユーザ名を指定します。一般的にデーモンは root ユーザとして実行します。セキュリティを考慮して、いくつかのサーバは daemon ユーザ、または最低の権限が与えられている nobody ユーザとして実行することも多く見られます。

server-program

接続を受け取ったときに実行するデーモンのフルパスです。デーモンが inetd によって内部的に提供されるサービスの場合 internal を使用します。

server-program-arguments

ここには、起動するときにデーモンに渡される、argv[0] から始まる引数を指定して、server-program と協調して動作します。mydaemon -d がコマンドラインの場合、server program arguments の値に mydaemon -d を指定します。また、デーモンが内部サービスの場合、ここに internal を指定します。

20.14.5. セキュリティ

インストールの時に選択したセキュリティプロファイルによっては、多くの inetd のデーモンがデフォルトで有効になっているかもしれません。
あるデーモンが特に必要でない場合には、それを無効にしてください!
問題となっているデーモンが記述されている行の先頭に "#" をおいて inetd にハングアップシグナルを送ってください。fingerd のようないくつかのデーモンは、動かそうとすべきではないかもしれません。なぜなら、
それらは攻撃者に対してあまりにも多くの情報を与えるからです。

セキュリティをあまり考慮せず、接続試行に対してタイムアウトまでの時間が長いか、タイムアウトしないデーモンもあります。
これは、特定のデーモンに攻撃者がゆっくり接続要求を送ることによって、
利用可能なリソースを飽和させることを可能にします。ある種のデーモンに ip-per-minute および max-child 制限を設けることはよい考えかもしれません。

TCP wrapper はデフォルトで有効です。inetd から起動されるさまざまなデーモンに対して TCP 制限を設けることの詳細については hosts_access(5) マニュアルページを参照してください。

20.14.6. その他

daytime, time, echo, discard, chargen および auth はすべて inetd が内部的に提供するサービスです。

auth サービスは identity (ident, identd) ネットワークサービスを提供し、ある程度設定可能です。

詳細については inetd(8) マニュアルを参照してください。

20.15. パラレルライン IP (PLIP)

PLIP はパラレルポート間で TCP/IP 通信を可能にします。
これはネットワークカードの無いマシンやノートパソコンにインストールするときに役に立ちます。

この節では以下について説明します。

- パラレル (ラップリンク または パラレルクロス) ケーブルの作成。
- 2 台のコンピュータの PLIP による接続。

20.15.1. パラレル (クロス) ケーブルの作成

コンピュータ用品店のほとんどでパラレル (クロス) ケーブルを購入することができます。 購入することができないか、単にケーブルがどのような構造であるか知りたい場合は、 次の表に通常のパラレルプリンタケーブルをもとに作成する方法が示されています。

表 *18.* ネットワーク向けのパラレル (クロス) ケーブル結線

A-名称	A-端	B-端	説明	Post/Bit
.... DATA0 -ERROR	.... 2 15	.... 15 2	Data	.... 0/0x01 1/0x08 ....
.... DATA1 +SLCT	.... 3 13	.... 13 3	Data	.... 0/0x02 1/0x10 ....
.... DATA2 +PE	.... 4 12	.... 12 4	Data	.... 0/0x04 1/0x20 ....
.... DATA3 -ACK	.... 5 10	.... 10 5	Strobe	.... 0/0x08 1/0x40 ....
.... DATA4 BUSY	.... 6 11	.... 11 6	Data	.... 0/0x10 1/0x80 ....
GND	18-25	18-25	GND	-

20.15.2. PLIP の設定

はじめに、ラップリンクケーブルを入手しなければなりません。 次に、両方のコンピュータのカーネルが lpt(4) ドライバ対応であることを確認してください。

```
# grep lp /var/run/dmesg.boot
lpt0: <Printer> on ppbus0
lpt0: Interrupt-driven port
```

パラレルポートは割り込み駆動ポートでなければなりません。 FreeBSD 4.X では、 以下のような行がカーネルコンフィギュレーションファイル内になければならないでしょう。

```
device ppc0 at isa? irq 7
```

FreeBSD 5.X では /boot/device.hints ファイルに以下の行がなければならないでしょう。

```
hint.ppc.0.at="isa"
hint.ppc.0.irq="7"
```

それからカーネルコンフィギュレーションファイルに `device plip` という行があるか、または plip.ko カーネルモジュールが読み込まれていることを確認してください。 どちらの場合でも ifconfig(8) コマンドを直接実行したときに、 パラレルネットワークインタフェースが現れるはずです。 FreeBSD 4.X ではこのようになります。

```
# ifconfig lp0
lp0: flags=8810<POINTOPOINT,SIMPLEX,MULTICAST> mtu 1500
```

FreeBSD 5.X ではこのようになります。

```
# ifconfig plip0
plip0: flags=8810<POINTOPOINT,SIMPLEX,MULTICAST> mtu 1500
```

パラレルインタフェースに対して用いられるデバイス名は FreeBSD 4.X (lpX) と FreeBSD 5.X (plipX) 間で異なります。

両方のコンピュータのパラレルインタフェースにラップリンクケーブルを接続します。

両方のネットワークインタフェースパラメータを `root` で設定します。 たとえば、FreeBSD 4.X を動作させている `host1` と FreeBSD 5.X を動作させている `host2` の両ホストを接続したい場合は次のようにします。

```
                host1 <-----> host2
IP Address    10.0.0.1      10.0.0.2
```

次のコマンドで `host1` 上のインタフェースを設定します。

```
# ifconfig lp0 10.0.0.1 10.0.0.2
```

次のコマンドで `host2` 上のインタフェースを設定します。

```
# ifconfig plip0 10.0.0.2 10.0.0.1
```

さて、これで接続が確立したはずです。詳細については lp(4) および lpt(4) マニュアルページをご覧ください。

さらに/etc/hosts に両ホストを加えるとよいでしょう。

```
127.0.0.1               localhost.my.domain localhost
10.0.0.1                host1.my.domain host1
10.0.0.2                host2.my.domain
```

接続がうまくいっているか確かめるために、 両方のホスト上で互いを ping してください。 たとえば host1 で以下を実行します。

```
# ifconfig lp0
lp0: flags=8851<UP,POINTOPOINT,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        inet 10.0.0.1 --> 10.0.0.2 netmask 0xff000000
# netstat -r
Routing tables

Internet:
Destination        Gateway           Flags     Refs     Use      Netif Expire
host2              host1               UH          0       0       lp0
# ping -c 4 host2
PING host2 (10.0.0.2): 56 data bytes
64 bytes from 10.0.0.2: icmp_seq=0 ttl=255 time=2.774 ms
64 bytes from 10.0.0.2: icmp_seq=1 ttl=255 time=2.530 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=255 time=2.556 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=255 time=2.714 ms

--- host2 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 2.530/2.643/2.774/0.103 ms
```

20.16. IPv6

IPv6 (IPng "IP next generation" とも呼ばれます) は、著名な IP プロトコル (IPv4 とも呼ばれます) の新しいバージョンです。 他の最新の *BSD システムと同様に FreeBSD は KAME IPv6 リファレンス実装を含んでいます。したがって、あなたの FreeBSD システムには IPv6を試すために必要なものすべてが備わっています。 この節では IPv6 の設定と実行に関して説明します。

1990 年代のはじめには、人々は IPv4 アドレス空間が急速に縮小していることに気づくようになりました。
インターネットの成長率が増大するにしたがって、 2 つの心配ごとがでてきました。

- アドレスの枯渇。 今日では、プライベートアドレス空間 (10.0.0.0/8, 192.168.0.0/24 など) およびネットワークアドレス変換 (NAT) が使用されているので、それほど心配されていません。
- ルーティングテーブルのエントリが大きくなりすぎていました。 これは今でも心配な事柄です。

IPv6 は以下の、そしてその他多くの問題を扱います。

- 128 bit アドレス空間。言い換えると、理論上 340,282,366,920,938,463,463,374,607,431,768,211,456 個のアドレスが利用可能です。これは地球上の一平方メータあたり、 およそ 6.67 * 10^27 個の IPv6 アドレスがあることを意味します。
- ルータは、 ルーティングテーブル内にネットワーク集約アドレスだけを格納することで、ルーティングテーブルの平均を 8192 項目程度に減らします。

他にも以下のように IPv6 の便利な機能がたくさんあります。

- アドレス自動設定 (RFC2462)
- エニーキャスト (anycast) アドレス ("one-out-of many" 訳注: 複数の異なるノードが応答する 1 つのアドレス。 RFC2526 を参照してください)。
- 強制マルチキャストアドレス
- IPsec (IP セキュリティ)
- シンプルなヘッダ構造
- モバイル IP
- IPv4 から IPv6 への移行手段

詳細については下記を参照してください。

- Sun.com の IPv6 概観
- IPv6.org
- KAME.net
- 6bone.net

20.16.1. IPv6 アドレスの背景

いくつか違うタイプの IPv6 アドレスがあります。 ユニキャスト (Unicast)、エニーキャスト (Anycast) およびマルチキャスト (Multicast) です。

ユニキャストアドレスは周知のアドレスです。 ユニキャストアドレスへ送られたパケットは、まさにそのアドレスに属するインターフェースに到着します。

エニーキャストアドレスはユニキャストアドレスと構文上判別不可能ですが、
インタフェース群に宛てられています。 エニーキャストアドレスに送られたパケットは
(ルータメトリック的に) 最も近いインタフェースに到着します。
エニーキャストアドレスはルータでしか使ってはいけません。

マルチキャストアドレスはインタフェース群を識別します。
マルチキャストアドレスに送られたパケットは、
マルチキャスト群に属するすべてのインタフェースに到着します。

IPv4 のブロードキャストアドレス (通常 `xxx.xxx.xxx.255`) は、IPv6 ではマルチキャストアドレスで表現されます。

表 *19.* 予約された *IPv6* アドレス

IPv6 アドレス	プレフィックス長 (ビット)	説明	備考
::	128 ビット	不特定	IPv4 の 0.0.0.0 参照
::1	128 ビット	ループバックアドレス	IPv4 の 127.0.0.1 参照
::00:xx:xx:xx:xx	96 ビット	IPv4 埋め込みアドレス	下位の 32 ビットは IPv4 アドレスです。 "IPv4 互換 IPv6 アドレス" とも呼ばれます。
::ff:xx:xx:xx:xx	96 ビット	IPv4 射影 IPv6 アドレス	下位の 32 ビットは IPv4 アドレスです。 IPv6 に対応していないホストに対するアドレスです。
fe80:: - feb::	10 ビット	リンクローカル	IPv4 のループバックアドレス参照
fec0:: - fef::	10 ビット	サイトローカル	
ff::	8 ビット	マルチキャスト	
001 (基数 2)	3 ビット	グローバルユニキャスト	すべてのグローバルユニキャストアドレスはこのプールから割り当てられます。 はじめの 3 ビットは "001" です。

20.16.2. IPv6 アドレスを読む

正規の書式では x:x:x:x:x:x:x:x と表されます。それぞれの "x" は 16 ビットの 16 進数です。たとえば FEBC:A574:382B:23C1:AA49:4592:4EFE:9982 となります。

すべてゼロの長い部分文字列がアドレス内によく現れます。 そのため、そのような部分文字列は "::" に短縮することができます。 たとえば、fe80::1 は正規形の fe80:0000:0000:0000:0000:0000:0000:0001 に対応します。

3 番目の形式は、最後の 32 ビットの部分を "." を分割文字として使う、 なじみ深い IPv4 (10 進) 形式で書くことです。 たとえば 2002::10.0.0.1 は (16 進) 正規形の 2002:0000:0000:0000:0000:0000:0a00:0001 に対応し、同時に 2002::a00:1 と書くこととも等価です。

ここまで来れば、下記を理解することができるでしょう。

```
# ifconfig
```

```
rl0: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
        inet 10.0.0.10 netmask 0xffffff00 broadcast 10.0.0.255
        inet6 fe80::200:21ff:fe03:8e1%rl0 prefixlen 64 scopeid 0x1
        ether 00:00:21:03:08:e1
        media: Ethernet autoselect (100baseTX )
```

```
        status: active
```

fe80::200:21ff:fe03:8e1%rl0 は自動的に設定されたリンクローカルアドレスです。
これは自動設定の一環として、 イーサネット MAC アドレスを変換したものを含んでいます。

IPv6 アドレス構造についての詳細は RFC3513 をご覧ください。

20.16.3. 接続

現在、他の IPv6 ホストおよびネットワークに接続するためには 4 つの方法があります。

- 6bone 実験ネットワークに参加する。
- 上流のプロバイダから IPv6 ネットワークの割り当てを受ける。
  手順については、インターネットプロバイダに問い合わせてください。
- IPv6 over IPv4 によるトンネル。
- ダイアルアップ接続の場合 freenet6 port を使用する。

ここでは、現在もっともよく使われている方法と思われる 6bone へ接続する方法を説明します。

はじめに 6bone サイトをみて、 あなたに最も近い 6bone 接続先を見つけてください。
責任者に連絡すると、少しばかり運がよければ、
接続を設定する方法についての指示を受けられるでしょう。 多くのばあい、これには GRE (gif)
トンネルの設定が含まれます。

6bone は 3ffe:: (16 ビット) という IPv6
アドレスを割り振られた実験目的のネットワークでしたが、 2006 年 6
月に運用を停止することになっています。 他の商用や試験的な IPv6
接続サービスを探してください。

ここに gif(4) トンネルを設定する典型的な例を示します。

```
# ifconfig gif0 create
# ifconfig gif0
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
# ifconfig gif0 tunnel MY_IPv4_ADDR  HIS_IPv4_ADDR
# ifconfig gif0 inet6 alias MY_ASSIGNED_IPv6_TUNNEL_ENDPOINT_ADDR
```

大文字になっている単語を、 上流の 6bone ノードから受け取った情報に置き換えてください。

これでトンネルが確立されます。ping6(8) を ff02::1%gif0
に送ることによって、トンネルが動作しているか確かめてください。 ping の応答を 2
つ受け取るはずです。

ff02:1%gif0 というアドレスに興味をそそられている場合のために説明すると、
これはマルチキャストアドレスです。 %gif0 は、ネットワークインタフェース gif0
上のマルチキャストアドレスが使用されるということを示しています。
マルチキャストアドレスに対して ping
を送ったので、トンネルのもう一方の端も応答します。

ここまで来ると 6bone アップリンクに経路設定することは比較的簡単でしょう。

```
# route add -inet6 default -interface gif0
# ping6 -n MY_UPLINK
```

```
# traceroute6 www.jp.FreeBSD.org
(3ffe:505:2008:1:2a0:24ff:fe57:e561) from 3ffe:8060:100::40:2, 30 hops max, 12 byte
packets
     1  atnet-meta6  14.147 ms  15.499 ms  24.319 ms
     2  6bone-gw2-ATNET-NT.ipv6.tilab.com  103.408 ms  95.072 ms *
     3  3ffe:1831:0:ffff::4  138.645 ms  134.437 ms  144.257 ms
     4  3ffe:1810:0:6:290:27ff:fe79:7677  282.975 ms  278.666 ms  292.811 ms
     5  3ffe:1800:0:ff00::4  400.131 ms  396.324 ms  394.769 ms
     6  3ffe:1800:0:3:290:27ff:fe14:cdee  394.712 ms  397.19 ms  394.102 ms
```

この出力はマシンによって異なります。 これで、あなたが www/mozilla のような IPv6 が利用可能なブラウザを持っていれば、 IPv6 サイト www.kame.net にいって踊るカメを見ることができるでしょう。

20.16.4. IPv6 世界の DNS

IPv6 のための新しい DNS レコードが 2 種類あります。

- AAAA レコード
- A6 レコード

AAAA レコードは簡単に使えます。

```
MYHOSTNAME          AAAA    MYIPv6ADDR
```

上記をプライマリゾーン DNS ファイルに加えて、 もらったばかりの IPv6 アドレスにホスト名を割り当ててください。 あなた自身で DNS ゾーンを管理していない場合は、 DNS プロバイダに頼んでください。 bind の最新バージョン (バージョン 8.3 および 9) は AAAA レコードに対応しています。

Part V: 付録

付録 A: FreeBSD の入手方法

A.1. Mirrors

FreeBSD
の公式のミラーサイトは、プロジェクトクラスタの管理者により運用されている数多くのコンピュータから構成されています。 GeoDNS により、ユーザには近くの利用可能なミラーが提供されます。
現在ミラーサイトが置かれている地域は、オーストラリア、ブラジル、ドイツ、日本 (2 つのサイト)、マレーシア、オランダ、南アフリカ、台湾、英国、アメリカ合衆国
(カリフォルニア、ニュージャージーおよびワシントン) です。

公式のミラーサービス:

サービス名	プロトコル	備考
download.FreeBSD.org	https ftp	`ftp.FreeBSD.org` と同じ内容です。`ftp` は古い名前なので、`download.FreeBSD.org` が推奨されます。
git.FreeBSD.org	`https` および `ssh` 経由の git	詳細については、git の利用 の節を参照してください。
pkg.FreeBSD.org	`http` および `https` 経由の pkg(8)	pkg(8) プログラムにより利用される公式の FreeBSD package リポジトリ
vuxml.FreeBSD.org / **www.VuXML.org**	https	FreeBSD プロジェクトの VuXML ウェブページ。`pkg audit` はこのサービスから脆弱性に関する一覧をダウンロードします。

すべての公式のミラーは、IPv4 および IPv6 に対応しています。

FreeBSD のウェブサイト (https://www.FreeBSD.org および https://docs.FreeBSD.org) は、GeoDNS インフラストラクチャでは運用されていません。 この実装は、進行中の課題です。

http://ftp-archive.FreeBSD.org は GeoDNS インフラストラクチャではなく、一つ地域 (US) でのみ運用されています。

プロジェクトでは、新しい地域やスポンサーを募集しています。
クラスター管理チームまで連絡してください。

コミュニティおよび他の会社により管理されているミラーの一覧:

国	ホスト名	プロトコル
オーストラリア ✉	ftp.au.FreeBSD.org	http http_v6 rsync rsync_v6
	ftp3.au.FreeBSD.org	http ftp rsync
オーストリア ✉	ftp.at.FreeBSD.org	http http_v6 ftp ftp_v6 rsync rsync_v6
ブラジル ✉	ftp2.br.FreeBSD.org	http rsync rsync_v6

国	ホスト名	プロトコル
	ftp3.br.FreeBSD.org	http ftp rsync
ブルガリア	ftp.bg.FreeBSD.org	ftp ftp_v6 rsync rsync_v6
チェコ共和国	ftp.cz.FreeBSD.org	http http_v6 rsync rsync_v6
デンマーク	ftp.dk.FreeBSD.org	http http_v6 ftp ftp_v6 rsync rsync_v6
フィンランド	ftp.fi.FreeBSD.org	ftp
フランス	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
ドイツ	ftp.de.FreeBSD.org	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
ギリシャ	ftp.gr.FreeBSD.org	http http_v6 ftp ftp_v6
	ftp2.gr.FreeBSD.org	http http_v6 ftp ftp_v6 rsync
日本	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
韓国	ftp.kr.FreeBSD.org	http https ftp rsync
	ftp2.kr.FreeBSD.org	rsync
ラトビア	ftp.lv.FreeBSD.org	http ftp
オランダ	ftp.nl.FreeBSD.org	http http_v6 ftp ftp_v6 rsync rsync_v6

国	ホスト名	プロトコル
	ftp2.nl.FreeBSD.org	http ftp rsync
ニュージーランド	ftp.nz.FreeBSD.org	http ftp
ノルウェー	ftp.no.FreeBSD.org	ftp ftp_v6 rsync rsync_v6
ポーランド	ftp.pl.FreeBSD.org	http http_v6 ftp rsync rsync_v6
ロシア	ftp.ru.FreeBSD.org	http http_v6 ftp ftp_v6 rsync rsync_v6
	ftp2.ru.FreeBSD.org	https ftp rsync
スロベニア	ftp.si.FreeBSD.org	http http_v6 ftp ftp_v6
南アフリカ	ftp.za.FreeBSD.org	https https_v6 rsync rsync_v6
	ftp2.za.FreeBSD.org	http http_v6 ftp_v6
	ftp4.za.FreeBSD.org	http ftp rsync
スウェーデン	ftp.se.FreeBSD.org	http http_v6 ftp ftp_v6 rsync rsync_v6
台湾	ftp4.tw.FreeBSD.org	https ftp rsync
	ftp5.tw.FreeBSD.org	http ftp
ウクライナ	ftp.ua.FreeBSD.org	http ftp ftp_v6 rsync rsync_v6
英国	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
アメリカ合衆国	ftp11.FreeBSD.org	http http_v6 ftp ftp_v6 rsync rsync_v6
	ftp14.FreeBSD.org	ftp rsync (Former official tier 1)
	ftp5.FreeBSD.org	http http_v6 ftp ftp_v6

コミュニティのミラーによりサポートされているプロトコル一覧は、2022-01-31 に更新されました。この一覧は保証されているわけではありません。

A.2. Git の利用

A.2.1. はじめに

2020 年 12 月、FreeBSD はソースコード、ドキュメントのすべてを管理するメインのバージョン管理システムを git に移行しました。 2021 年 4 月、FreeBSD は Ports Collection のすべてを管理するバージョン管理システムを git に移行しました。

一般的には Git は開発用ツールです。 ユーザは好みに合わせて、FreeBSD ベースシステムのアップデートに freebsd-update (“FreeBSD Update”)、 FreeBSD Ports Collection のアップデートに git (“Ports Collection の利用”)を使用できます。

この章では、FreeBSD への Git のインストール方法および FreeBSD ソースコードリポジトリのローカルコピーの作成方法について説明します。

A.2.2. インストール

Ports Collection または package を使って Git をインストールできます。

```
# pkg install git
```

A.2.3. Git の実行

ソースコードをローカルディレクトリに新しくコピーするには、git clone を使ってください。 このファイルのあるディレクトリのことを ワークツリー と呼びます。

Git は、リポジトリの指定に URL を用います。 リポジトリには base, doc および ports の 3 種類あります。 base は FreeBSD ベースシステムのソースコード、doc はドキュメント、そして ports は FreeBSD Ports Collection のリポジトリです。 これら 3 つのリポジトリはすべて HTTPS および SSH という 2 つの異なるプロトコル経由でアクセスできます。 たとえば、https://git.FreeBSD.org/src.git という URL は、https プロトコルによる src リポジトリの main ブランチを示します。

表 *20. FreeBSD Git* リポジトリの *URL* テーブル

項目	**Git URL**
HTTPS 経由の読み取り専用 src リポジトリ	https://git.FreeBSD.org/src.git
Anonymous ssh による読み取り専用 src リポジトリ	ssh://anongit@git.FreeBSD.org/src.git
HTTPS 経由の読み取り専用 doc リポジトリ	https://git.FreeBSD.org/doc.git
Anonymous ssh による読み取り専用 doc リポジトリ	ssh://anongit@git.FreeBSD.org/doc.git
HTTPS 経由の読み取り専用 ports リポジトリ	https://git.FreeBSD.org/ports.git
Anonymous ssh による読み取り専用 ports リポジトリ	ssh://anongit@git.FreeBSD.org/ports.git

プロジェクトのメンバーが管理する外部のミラーも存在します。 外部ミラー の節を参照してください。

FreeBSD システムのソースコードリポジトリを clone するには、以下のコマンドを実行してください。

```
# git clone -o freebsd https://git.FreeBSD.org/src.git /usr/src
```

ここで -o freebsd オプションは origin を指定します。 FreeBSD のドキュメントの慣例で、origin は freebsd とします。
初めてチェックアウトする際には、リモートリポジトリのすべてのブランチをダウンロードするので時間

がかかります。

ワーキングツリーには最初、CURRENT に対応する main ブランチのソースコードがダウンロードされます。13-STABLE に変更するには以下のように実行してください。

```
# cd /usr/src
# git checkout stable/13
```

ワーキングツリーは、git pull によりアップデートできます。上記の例で作成された /usr/src をアップデートするには、以下のようになります。

```
# cd /usr/src
# git pull --rebase
```

チェックアウトと比較すると、このアップデートでは変更点のあるファイルのみが転送されるので高速です。

A.2.4. ウェブベースのリポジトリブラウザ

FreeBSD プロジェクトは、現在 cgit をウェブベースのリポジトリブラウザ (https://cgit.FreeBSD.org/) として使用しています。

A.2.5. 開発者向けの説明

リポジトリへの書き込みアクセスについてはの詳細は、Committer's Guide をご覧ください。

A.2.6. 外部ミラー

FreeBSD.org は以下のミラーを管理していませんが、プロジェクトのメンバーが現在も維持しています。ユーザおよび開発者は自由にこれらのミラーのリポジトリを pull したりブラウザで見ることができます。doc GitHub リポジトリへの pull request は accept されますが、それ以外について、これらのミラーとのプロジェクトワークフローは議論中です。

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. メーリングリスト

FreeBSD プロジェクトにおける git の一般的な使用方法や質問についてのメインのメーリングリストは freebsd-git です。 コミットメッセージの一覧などの詳細については、メーリングリストの章をご覧ください。

A.2.8. SSH ホスト鍵

- gitrepo.FreeBSD.org ホスト鍵のフィンガープリント:
  - ECDSA 鍵のフィンガープリントは SHA256:seWO5D27ySURcx4bknTNKlC1mgai0whP443PAKEvvZA です。
  - ED25519 鍵のフィンガープリントは SHA256:lNR6i4BEOaaUhmDHBA1WJsO7H3KtvjE2r5q4sOxtIWo です。
  - RSA 鍵のフィンガープリントは SHA256:f453CUEFXEJAXlKeEHV+ajJfeEfx9MdKQUD7lIscnQI です。
- git.FreeBSD.org ホスト鍵のフィンガープリント:
  - ECDSA 鍵のフィンガープリントは SHA256:/UlirUAsGiitupxmtsn7f9b7zCWd0vCs4Yo/tpVWP9w です。
  - ED25519 鍵のフィンガープリントは SHA256:y1ljKrKMD3lDObRUG3xJ9gXwEIuqnh306tSyFd1tuZE です。
  - RSA 鍵のフィンガープリントは SHA256:jBe6FQGoH4HjvrIVM23dcnLZk9kmpdezR/CvQzm7rJM です。

これらは DNS の SSHFP レコードとしても公開されています。

A.3. Subversion の利用

A.3.1. はじめに

2020 年 12 月より、FreeBSD のソースコード、ドキュメントのすべてを管理するメインのバージョン管理システムは git に移行しました。 git リポジトリの stable/11, stable/12 および関連するリリースのブランチは、subversion リポジトリにエクスポートされます。 このエクスポートは、各ブランチの保守終了予定日まで行われる予定です。 2012 年 7 月から 2021 年 3 月までの間 FreeBSD は、FreeBSD Ports Collection のすべてを管理するバージョン管理システムに Subversion を使用していました。 2021 年 4 月より、FreeBSD の Ports Collection のすべてを管理するメインのバージョン管理システムは git に移行しました。

-一般的には Subversion は開発者向けのツールです。 ユーザは好みに応じて、FreeBSD のベースシステムのアップデートに freebsd-update (「FreeBSD Update」)、Ports Collection のアップデートには git (「Ports Collection の利用」) を使用できます。 2021 年 3 月以降、subversion はレガシーブランチ (stable/11 および stable/12) でのみ使用されます。

この節では、FreeBSD システムへの Subversion のインストール方法、および FreeBSD リポジトリをローカルに作成する方法について説明します。 さらに Subversion を利用するための情報についても紹介します。

A.3.2. Svnlite

FreeBSD には、Subversion より軽い svnlite がインストールされています。 Subversion の port または package は、Python もしくは Perl API が必要な時や、最新の Subversion を使用したい時のみ必要となります。

通常の Subversion と、 svnlite との違いは、 使用する時のコマンド名が異なるだけです。

A.3.3. インストール

svnlite を利用できない場合や、 フルバージョンの Subversion を使いたいのであれば、 事前に Subversion をインストールしておく必要があります。

Subversion は Ports Collection からインストールできます。

```
# cd /usr/ports/devel/subversion
# make install clean
```

package を使って Subversion をインストールすることもできます。

```
# pkg install subversion
```

A.3.4. Subversion の実行

ローカルディレクトリにソースコードをダウンロードするには、 svn コマンドを使ってください。 このディレクトリにあるファイルを、 ローカル作業コピー と呼びます。

checkout をはじめて使う前に、 ローカルディレクトリを移動するか削除してください。 svn 以外の方法で用意されたディレクトリでチェックアウトすると、 すでに存在するファイルと、 リポジトリから持ってきたファイルとの間で衝突が起きてしまいます。

Subversion では、リポジトリの指定に *protocol://hostname/path* 形式の URL を用います。 以下に記載されているように、 アクセスする FreeBSD リポジトリは、パス (path) の最初で指定します。 リポジトリは 3 つあります。 base は FreeBSD ベースシステムのソースコード、ports は Ports Collection、 そして doc はドキュメントのリポジトリです。 たとえば、https://svn.FreeBSD.org/base/head/ という URL は、https プロトコルによる src リポジトリのメインブランチを示しています。

以下のように入力して、リポジトリからチェックアウトしてください。

```
# svn checkout https://svn.FreeBSD.org/repository/branch lwcdir
```

ここで、*repository*, *branch* および *root* は以下のとおりです。

- *repository* には、 プロジェクトリポジトリの base, ports または doc のどれかひとつを指定します。
- *branch* は、使うリポジトリによります。 ports および doc では、ほとんどの変更が head

ブランチで行われます。 base リポジトリでは、head ブランチで -CURRENT の最新バージョンを管理しています。 -STABLE ブランチの最新バージョンは、 11.*x* は stable/11, そして 12.*x* は stable/12 で管理しています。

- *lwcdir* は、 指定したブランチの中身が置かれるターゲットのディレクトリです。 通常 ports は /usr/ports、 base は /usr/src、 そして doc では /usr/doc と指定します。

以下の例では、ソースツリーを FreeBSD リポジトリから HTTPS プロトコルを使ってチェックアウトします。 それらは、/usr/src のローカル作業コピーに置かれます。 もし /usr/src がすでに存在していて、それが svn によって生成されたものでなければ、チェックアウトする前に、名前を変更するか削除してください。

```
# svn checkout https://svn.FreeBSD.org/base/head /usr/src
```

初めてチェックアウトする際には、
リモートリポジトリのすべてのブランチをダウンロードする必要があるので、時間がかかります。
我慢してください。

初めてのチェックアウト後は、 以下を実行することでローカル作業コピーをアップデートできます。

```
# svn update lwcdir
```

この例で作成された /usr/src をアップデートするには、 以下のようにしてください。

```
# svn update /usr/src
```

アップデートはチェックアウトにくらべ、 変更点のあるファイルのみが転送されるので高速です。

チェックアウト後、ローカル作業コピーをアップデートするもうひとつの方法は、 /usr/ports, /usr/src または /usr/doc ディレクトリの Makefile で提供されています。 SVN_UPDATE を設定して update ターゲットを使ってください。 たとえば、/usr/src をアップデートするには、以下のようにしてください。

```
# cd /usr/src
# make update SVN_UPDATE=yes
```

A.3.5. Subversion ミラーサイト

FreeBSD Subversion リポジトリは、

```
svn.FreeBSD.org
```

です。これは、公にアクセス可能なミラーネットワークで、 GeoDNS を用いて適切なバックエンドサーバを選択しています。 ブラウザを用いて FreeBSD の Subversion リポジトリを参照するには、https://svnweb.FreeBSD.org/ を利用してください。

HTTPS は推奨されているプロトコルです。 自動的に証明書を検証するために、security/ca_root_nss port をインストールする必要があります。

A.3.6. より詳しい情報

Subversion の利用に関する他の情報は、 Version Control with Subversion や Subversion Documentation といった "Subversion Book" をご覧ください。

A.4. CD および DVD セット

FreeBSD の CD および DVD のセットは以下のオンライン業者から入手できます。

- FreeBSD Mall, Inc.
  1164 Claremont Dr
  Brentwood, CA
  94513
  USA
  Phone: +1 925 240-6652
  Fax: +1 925 674-0821
  Email: info@freebsdmall.com
  WWW: https://www.freebsdmall.com
- Getlinux
  WWW: https://www.getlinux.fr/
- Dr. Hinner EDV
  Schäftlarnstr. 10 // 4. Stock
  D-81371 München
  Germany
  Phone: +49 171 417 544 6
  Email: infow@hinner.de
  WWW: http://www.hinner.de/linux/freebsd.html

付録 B: 参考図書

訳: *Yukihiro Nakai <nakai@FreeBSD.org>, 1996 年 10 月 12 日。*

FreeBSD オペレーティングシステムの個々の部分については
マニュアルページで定義のような説明がなされていますが, どうやってその部分どうしをつなぎあわせて
オペレーティングシステム全体を円滑に動作させるかについては, ほとんど説明されていません。
それを補うためにはのよい本や, UNIX® システム管理についての
利用者向けのマニュアルが欠かせません。

B.1. FreeBSD 専門の書籍

非英語文化圏の書籍:

- FreeBSD 入門與應用 (繁体字中国語)。 Drmaster 発行, 1997. ISBN 9-578-39435-7.
- FreeBSD Unleashed (簡体字中国語訳), China Machine Press 発行. ISBN 7-111-10201-0.
- FreeBSD From Scratch Second Edition (簡体字中国語), China Machine Press 発行. ISBN 7-111-10286-X.
- FreeBSD ハンドブック第 2 版 (簡体字中国語訳), Posts & Telecom Press 発行. ISBN 7-115-10541-3.
- FreeBSD & Windows (簡体字中国語), China Railway Publishing House 発行. ISBN 7-113-03845-X
- FreeBSD Internet Services HOWTO (簡体字中国語), China Railway Publishing House 発行. ISBN 7-113-03423-3
- FreeBSD入門キット AT互換機版 第二版。 宮□忠臣 著。 秀和システム。 ISBN 4-87966-535-5 C3055 2900 円。
- ここまでできる FreeBSD パワーガイド。 霜山 滋, 仲道 嘉夫, 山中 右次 著。 秀和システム。 ISBN 4-87966-637-8 2600円。
- FreeBSD徹底入門。 あさだ たくや / 天川 修平 / 衛藤 敏寿 / 浜田 直樹 / 細川 達己 / 三田 吉郎 著。 翔泳社。 ISBN 4-88135-473-6 3600 円。
- パーソナル UNIX スターターキット FreeBSD。 民田 雅人 / 古場 正行 / 増田 佳泰 / 天池 健 / 宮川 晋 共著。 アスキー。 ISBN 4-7561-1733-3 3000 円。
- FreeBSD ハンドブック (日本語版)。 アスキー。 ISBN 4-7561-1580-2 3800 円。
- FreeBSD mit Methode (ドイツ語版)。 Computer und Literatur Verlag/Vertrieb Hanser 発行。 1998。 ISBN 3-932311-31-0
- FreeBSD de Luxe (ドイツ語), Verlag Modere Industrie 発行, 2003 年。ISBN 3-8266-1343-0.
- FreeBSD インストール & 活用マニュアル, 毎日コミュニケーションズ発行。1998 年. ISBN 4-8399-0112-0.
- Onno W Purbo, Dodi Maryanto, Syahrial Hubbany, Widjil Widodo 著 *FreeBSD を使ったインターネットサーバの構築 (Building Internet Server with FreeBSD)* (インドネシア語), Elex Media Komputindo 発行。
- Absolute BSD: The Ultimate Guide to FreeBSD (繁体字中国語訳) GrandTech Press 発行 (2003 年) 。ISBN 986-7944-92-5.

- The FreeBSD 6.0 Book (繁体字中国語), Drmaster 発行 (2006 年)。ISBN 9-575-27878-X.

英語の書籍:

- Absolute FreeBSD, 2nd Edition: The Complete Guide to FreeBSD, No Starch Press 刊、2007 年。ISBN: 978-1-59327-151-0
- The Complete FreeBSD, O'Reilly、 2003 年。ISBN: 0596005164
- The FreeBSD Corporate Networker's Guide, Addison-Wesley 刊、 2000 年。ISBN: 0201704811
- FreeBSD: An Open-Source Operating System for Your Personal Computer、The Bit Tree Press 刊、2001 年。 ISBN: 0971204500
- Teach Yourself FreeBSD in 24 Hours, Sams 刊、 2002 年。ISBN: 0672324245
- FreeBSD 6 unleashed, Sams 刊、 2006 年。ISBN: 0672328755
- FreeBSD: The Complete Reference, McGrawHill 刊、 2003 年。ISBN: 0072224096

B.2. 利用者向けのガイド

- オハイオ州立大学による UNIX Introductory Course。 オンラインで HTML 版と PostScript 版が利用可能。

  FreeBSD イタリア語ドキュメンテーションプロジェクトの一環 として、このドキュメントのイタリア語訳 が用意されています。

- FreeBSD 友の会 jpman プロジェクト。FreeBSD User's Reference Manual (日本語訳)。毎日コミュニケーションズ, 1998。ISBN4-8399-0088-4 P3800E。
- Edinburgh University による UNIX 環境の初心者向け オンラインガイド。

B.3. 管理者向けのガイド

- FreeBSD 友の会 jpman プロジェクト。FreeBSD System Administrator's Manual (日本語訳)。毎日コミュニケーションズ, 1998. ISBN4-8399-0109-0 P3300E.
- Dreyfus, Emmanuel. Cahiers de l'Admin: BSD 第 2 版。(フランス語、「管理者ノート」)、Eyrolles, 2004. ISBN 2-212-11463-X

B.4. プログラマ向けのガイド

- Computer Systems Research Group, UC Berkeley. *4.4BSD Programmer's Reference Manual*. O'Reilly & Associates, Inc., 1994. ISBN 1-56592-078-3
- Computer Systems Research Group, UC Berkeley. *4.4BSD Programmer's Supplementary Documents*. O'Reilly & Associates, Inc., 1994. ISBN 1-56592-079-1
- Harbison, Samuel P. and Steele, Guy L. Jr. *C: A Reference Manual*. 4th Ed. Prentice Hall, 1995. ISBN 0-13-326224-3 (訳注: 邦訳は以下のものが出版されています。 斎藤 信男監訳。 新・詳説 *C* 言語リファレンス *[H&S*リファレンス*]*。 ソフトバンク, 1994。 ISBN 4-89052-506-8 原本は第 4 版だが, 訳出は第 3 版のみ。)

- Kernighan, Brian and Dennis M. Ritchie. *The C Programming Language*. 2nd Ed. PTR Prentice Hall, 1988. ISBN 0-13-110362-8 (訳注: 邦訳は以下のものが出版されています。 石田 晴久 訳。 プログラミング言語 *C* 第 *2* 版*(*訳書訂正版*)* 共立出版, 1989。 ISBN 4-320-02692-6)
- Lehey, Greg. *Porting UNIX Software*. O'Reilly & Associates, Inc., 1995. ISBN 1-56592-126-7
- Plauger, P. J. *The Standard C Library*. Prentice Hall, 1992. ISBN 0-13-131509-9 (訳注: 邦訳は以下のものが出版されています。 福富 寛 / 門倉 明彦 / 清水 恵介 訳。 標準 *C* ライブラリ *ANSI/ISO/JIS C*規格. トッパン, 1995。 ISBN 4-8101-8541-9)
- Spinellis, Diomidis. Code Reading: The Open Source Perspective. Addison-Wesley, 2003. ISBN 0-201-79940-5
- Spinellis, Diomidis. Code Quality: The Open Source Perspective. Addison-Wesley, 2006. ISBN 0-321-16607-8
- Stevens, W. Richard and Stephen A. Rago. *Advanced Programming in the UNIX Environment*. 2nd Ed. Reading, Mass. : Addison-Wesley, 2005. ISBN 0-201-43307-9 (訳注: 第 1 版の邦訳は以下のものが出版されています。 大木 敦雄 訳。 詳解 *UNIX* プログラミング。トッパン, 1994。 ISBN 4-89052-524-6)
- Stevens, W. Richard. *UNIX Network Programming*. 2nd Ed. PTR Prentice Hall, 1998. ISBN 0-13-949876-1 (訳注: 第 1 版の邦訳は以下のものが出版されています. 篠田 陽一 訳. *UNIX* ネットワークプログラミング。 トッパン, 1992. ISBN 4-8101-8509-5) 第 2 版の邦訳は以下のものが出版されています。 篠田 陽一 訳. *UNIX* ネットワークプログラミング 第 *2* 版 *Vol.1*。 トッパン, 1999。 ISBN 4-8101-8612-1)

B.5. オペレーティングシステム内部

- Andleigh, Prabhat K. *UNIX System Architecture*. Prentice-Hall, Inc., 1990. ISBN 0-13-949843-5
- Jolitz, William. "Porting UNIX to the 386". *Dr. Dobb's Journal*. January 1991-July 1992.
- Leffler, Samuel J., Marshall Kirk McKusick, Michael J Karels and John Quarterman *The Design and Implementation of the 4.3BSD UNIX Operating System*. Reading, Mass. : Addison-Wesley, 1989. ISBN 0-201-06196-1 (訳注: 邦訳は以下のものが出版されています。 中村 明 / 相田 仁 / 計 宇生 / 小池 汎平 訳。 *UNIX 4.3BSD*の設計と実装。丸善, 1991。 ISBN 4-621-03607-6)
- Leffler, Samuel J., Marshall Kirk McKusick, *The Design and Implementation of the 4.3BSD UNIX Operating System: Answer Book*. Reading, Mass. : Addison-Wesley, 1991. ISBN 0-201-54629-9 (訳注: 邦訳は以下のものが出版されています。 相田 仁 / 計 宇生 / 小池 汎平 訳。 *UNIX 4.3BSD*の設計と実装。 アンサーブック, トッパン, 1991。 ISBN 4-8101-8039-5)
- McKusick, Marshall Kirk, Keith Bostic, Michael J Karels, and John Quarterman. *The Design and Implementation of the 4.4BSD Operating System*. Reading, Mass. : Addison-Wesley, 1996. ISBN 0-201-54979-4

  (この本の第二章が FreeBSD ドキュメンテーションプロジェクト の一部として オンラインで公開されています。)
- Marshall Kirk McKusick, George V. Neville-Neil *The Design and Implementation of the FreeBSD Operating System*. Boston, Mass. : Addison-Wesley, 2004. ISBN 0-201-70245-2
- Marshall Kirk McKusick, George V. Neville-Neil, Robert N. M. Watson *The Design and Implementation of the FreeBSD Operating System, 2nd Ed.*. Westford, Mass. : Pearson Education,

Inc., 2014. ISBN 0-321-96897-2

- Stevens, W. Richard. *TCP/IP Illustrated, Volume 1: The Protocols*. Reading, Mass. : Addison-Wesley, 1996. ISBN 0-201-63346-9
- Schimmel, Curt. *Unix Systems for Modern Architectures*. Reading, Mass. : Addison-Wesley, 1994. ISBN 0-201-63338-8
- Stevens, W. Richard. *TCP/IP Illustrated, Volume 3: TCP for Transactions, HTTP, NNTP and the UNIX Domain Protocols*. Reading, Mass. : Addison-Wesley, 1996. ISBN 0-201-63495-3 (訳注: 邦訳は以下のものが出版されています。 中本 幸一 / 石川 裕次 / 田中 伸佳 訳。 詳解 *TCP/IP Vol.3:* トランザクション *TCP, HTTP, NNTP, UNIX* ドメインプロトコル, アジソンウェスレイパブリッシャーズジャパン, 1998。 ISBN 4-8101-8039-5)
- Vahalia, Uresh. *UNIX Internals — The New Frontiers*. Prentice Hall, 1996. ISBN 0-13-101908-2 (訳注: 邦訳は以下のものが出版されています。 徳田 英幸 / 中村 明 / 戸辺 義人 / 津田 悦幸 訳。 最前線*UNIX*のカーネル, ピアソンエデュケーション, 2000。 ISBN 4-89471-189-3)
- Wright, Gary R. and W. Richard Stevens. *TCP/IP Illustrated, Volume 2: The Implementation*. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-63354-X
- Messmer, Hans-Peter. *The Indispensable PC Hardware Book*, 4th Ed. Reading, Mass : Addison-Wesley Pub. Co., 2002. ISBN 0-201-59616-4

B.6. セキュリティの参考資料

- Cheswick, William R. and Steven M. Bellovin. *Firewalls and Internet Security: Repelling the Wily Hacker*. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-63357-4 (訳注: 邦訳は以下のものが出版されています。 川副 博 監訳。ファイアウォール。 ソフトバンク, 1995。 ISBN 4-89052-672-2)
- Garfinkel, Simson. *PGP Pretty Good Privacy* O'Reilly & Associates, Inc., 1995. ISBN 1-56592-098-8

B.7. ハードウェアの参考資料

- Anderson, Don and Tom Shanley. *Pentium Processor System Architecture*. 2nd Ed. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-40992-5
- Ferraro, Richard F. *Programmer's Guide to the EGA, VGA, and Super VGA Cards*. 3rd ed. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-62490-7
- Intel Corporation は、自社の CPU やチップセットに関する文書を自社の 開発者向け Web サイトで公開しています。文書のフォーマットは通常 PDF です。
- Shanley, Tom. *80486 System Architecture*. 3rd Ed. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-40994-1
- Shanley, Tom. *ISA System Architecture*. 3rd Ed. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-40996-8
- Shanley, Tom. *PCI System Architecture*. 4th Ed. Reading, Mass. : Addison-Wesley, 1999. ISBN 0-201-30974-2
- Van Gilluwe, Frank. *The Undocumented PC*, 2nd Ed. Reading, Mass: Addison-Wesley Pub. Co., 1996. ISBN 0-201-47950-8

B.8. UNIX® の歴史

- Lion, John *Lion's Commentary on UNIX, 6th Ed. With Source Code*. ITP Media Group, 1996. ISBN 1573980137
- Raymond, Eric s. *The New Hacker's Dictionary, 3rd edition*. MIT Press, 1996. ISBN 0-262-68092-0 これは ジャーゴンファイル (Jargon File) として知られています。
- Saulus, Peter H. *A quarter century of UNIX*. Addison-Wesley Publishing Company, Inc., 1994. ISBN 0-201-54777-5
- Simon Garfinkel, Daniel Weise, Steven Strassmann. *The UNIX-HATERS Handbook*. IDG Books Worldwide, Inc., 1994. ISBN 1-56884-203-1. 絶版となりましたが、オンライン で入手できます。
- Don Libes, Sandy Ressler *Life with UNIX* - special edition. Prentice-Hall, Inc., 1989. ISBN 0-13-536657-7 (訳注: 邦訳は以下のものが出版されています。 坂本 文 監訳. *Life with UNIX*. アスキー, 1990。 ISBN 4-7561-0783-4 邦訳が Special 版の訳出か否かは不明)
- *BSD 系 OS* の系譜図。 https://svnweb.freebsd.org/base/head/share/misc/bsd-family-tree?view=co か、もしくは、FreeBSD マシンにある /usr/share/misc/bsd-family-tree。
- *Networked Computer Science Technical Reports Library*.
- *Computer Systems Research group (CSRG)* からの古い *BSD* リリース集。http://www.mckusick.com/csrg/: この 4 枚 CD セットには、1BSD から 4.4BSD までと 4.4BSD-Lite2 が含まれます (残念ながら 2.11BSD は含まれていません)。 また 4 枚目の CD には、最終ソースおよび SCCS ファイルが含まれています。
- Kernighan, Brian *Unix: A History and a Memoir*. Kindle Direct Publishing, 2020. ISBN 978-169597855-3

B.9. 定期刊行物、雑誌およびジャーナル

- Admin Magazin (in German), published by Medialinx AG. ISSN: 2190-1066
- BSD Magazine, published by Software Press Sp. z o.o. SK. ISSN: 1898-9144
- BSD Now - Video Podcast, published by Jupiter Broadcasting LLC
- BSD Talk Podcast, by Will Backman
- FreeBSD Journal, published by S&W Publishing, sponsored by The FreeBSD Foundation. ISBN: 978-0-615-88479-0

付録 C: インターネット上のリソース

FreeBSD の進歩は急速であり、印刷したメディアは最新の開発をフォローするのに実用的ではありません。それだけしかない、というわけではありませんが、最新情報を入手する方法としては電子的なリソースがベストです。FreeBSD はボランティアの努力によって、ユーザコミュニティ自体が、一種の "テクニカルサポート部門" としての役割も通常果たしており、電子メール、ウェブフォーラムおよび Usenet のニュースがこれらのコミュニティにたどり着く最も効果的な方法になっています。

以下に、FreeBSD ユーザコミュニティに連絡を取る場合の最も重要な点についての概略を示します。ここに書かれていない他のリソースをご存知であれば、それらをここに含めることができるように、FreeBSD documentation project メーリングリスト にお知らせください。

C.1. ウェブサイト

- The FreeBSD Forums は、FreeBSD の質問および技術的な議論のためのウェブベースのフォーラムです。
- BSDConferences YouTube Channel は、世界中で開催されている BSD カンファレンスでのプレゼンテーションの高品質のビデオです。主要な開発者による FreeBSD の新しい進展についてのプレゼンテーションをぜひともご覧ください。

C.2. メーリングリスト

メーリングリストは、FreeBSD の関係者に対し質問を投稿したり、技術的な議論を行うのに、最も直接的な方法です。さまざまな FreeBSD の関連トピックに対し、幅広いメーリングリストが存在しています。質問を適切なメーリングリストに投稿すれば、早く、的確な反応がいつでも得られることでしょう。

さまざまなメーリングリストの憲章をこのドキュメントの最後に記載します。私たちは、メーリングリストの質、特に技術面に関する質を高く保つために努力しているので、メーリングリストに参加する前にその憲章を読んでください。私たちのメーリングリストの参加者のほとんどは、非常にたくさんの FreeBSD に関連したメッセージを毎日受け取っており、メーリングリストの利用に関する憲章やルールは、メーリングリストの S/N 比を高く保つためのものです。そうしないと、結果的に、メーリングリストがプロジェクトにとって事実上のコミュニケーションの手段になってしまうでしょう。

FreeBSD メーリングリストにメールを送信できるかどうかを確認するには、*FreeBSD test* メーリングリスト にテストメッセージを送信してください。他のメーリングリストには、テストメッセージを送信しないでください。

どのメーリングリストに質問を投稿すべきか迷った場合には、How to get best results from the FreeBSD-questions mailing list をご覧ください。

どこのメーリングリストに投稿する場合でも、メーリングリストを最大限に活用する方法を理解しておいてください。たとえば、Mailing List Frequently Asked Questions (FAQ) 文書を読んで、繰り返し行われる議論を避ける方法を理解してください。

メーリングリストはいずれもアーカイブされており、それらは FreeBSD World Wide Web server で検索することができます。キーワード検索可能なアーカイブの提供は、良くある質問に対する回答を見つけるすぐれた方法ですから、質問を投稿する前に調べてみるべきでしょう。このことは、FreeBSD メーリングリストに送信されたメッセージは、ずっとアーカイブされることを意味しています。プライバシーの保護が問題になるような場合には、使い捨てのメールアドレスを用い、公な情報のみを送ってください。

C.2.1. メーリングリストの概説

一般的なメーリングリスト: 以下のものは誰でも自由に参加できる (そしておすすめの) 一般的なものです。

リスト	目的
freebsd-advocacy	FreeBSD の福音伝道
FreeBSD announcements メーリングリスト	重要なイベントやプロジェクトのマイルストン (モデレータ制)
freebsd-arch	アーキテクチャ、設計に関する議論
freebsd-bugbusters	FreeBSD 障害報告データベースおよび関連するツールの管理に関する議論
freebsd-bugs	バグレポート
freebsd-chat	FreeBSD コミュニティに関連する技術的ではない話題
freebsd-chromium	FreeBSD に固有の Chromium の問題について
FreeBSD-CURRENT メーリングリスト	FreeBSD-CURRENT の使用に関連する議論
freebsd-isp	FreeBSD を用いている インターネットサービスプロバイダの話題
freebsd-jobs	FreeBSD 関連の雇用機会に関する話題
freebsd-quarterly-calls	四半期開発進捗レポートの呼びかけ (モデレータ制)
freebsd-questions	ユーザからの質問と技術サポート
FreeBSD security notifications メーリングリスト	セキュリティに関する通知 (モデレータ制)
FreeBSD-STABLE; メーリングリスト	FreeBSD-STABLE の使用に関連する議論
FreeBSD test メーリングリスト	メッセージの送信試験を行なうために、実際のメーリングリストの代わりに使うアドレス
freebsd-women	FreeBSD の女性支援

技術的なメーリングリスト: 以下のメーリングリストは、技術的な議論のためのものです。それらの利用や内容のためにしっかりとしたガイドラインがあるので、これらのメーリングリストに入ったり、どれか一つにメールを送ったりする前には、それらのメーリングリストの憲章を注意深く読んでください。

リスト	目的
FreeBSD ACPI メーリングリスト	ACPI および電源管理の開発
freebsd-fortran	FreeBSD での Fortran
freebsd-amd64	FreeBSD の AMD64 システムへの移植 (モデレータ制)
freebsd-apache	Apache に関連した ports についての議論
freebsd-arm	FreeBSD の ARM® プロセッサへの移植
freebsd-atm	FreeBSD での ATM ネットワーク使用に関する話題
freebsd-bluetooth	FreeBSD で Bluetooth® 技術の使用
freebsd-cloud	クラウドプラットフォーム (EC2, GCE, Azure など) での FreeBSD
freebsd-cluster	FreeBSD のクラスタ環境での利用
freebsd-database	FreeBSD 上でのデータベースの利用や開発に関する議論
freebsd-desktop	デスクトップでの FreeBSD の利用や改良について
dev-ci	継続的インテグレーションサーバからのビルドおよび試験レポート
dev-reviews	FreeBSD レビューシステムからの通知
freebsd-doc	FreeBSD 関連ドキュメントの作成
freebsd-drivers	FreeBSD のデバイスドライバの書き方について
freebsd-dtrace	FreeBSD における Dtrace の利用と開発
freebsd-eclipse	Eclipse IDE, ツール、 リッチクライアントアプリケーションの FreeBSD ユーザおよび ports
freebsd-elastic	FreeBSD 固有の ElasticSearch に関する議論
freebsd-embedded	組み込みアプリケーションにおける FreeBSD の利用
freebsd-emulation	Linux/MS-DOS®/Windows® のような他のシステムのエミュレーション
freebsd-enlightenment	Enlightenment および Enlightenment アプリケーションの移植
freebsd-erlang	FreeBSD 固有の Erlang に関する議論
freebsd-firewire	FreeBSD FireWire® (iLink, IEEE 1394) に関する技術的な議論
freebsd-fs	ファイルシステム
freebsd-games	FreeBSD でのゲームのサポート
freebsd-gecko	Gecko レンダリングエンジン に関する議論
freebsd-geom	GEOM に関連した議論と実装

リスト	目的
freebsd-git	FreeBSD プロジェクトでの git の使用に関する議論
freebsd-gnome	GNOME および GNOME アプリケーションの移植
freebsd-hackers	一般的な技術の議論
freebsd-haskell	FreeBSD 固有の Haskell に関する議論
freebsd-hardware	FreeBSD の走るハードウェアの一般的な議論
freebsd-i18n	FreeBSD の国際化
freebsd-infiniband	FreeBSD での Infiniband の使用
freebsd-ipfw	IP firewall コードの再設計に関する技術的議論
freebsd-isdn	ISDN 開発者
freebsd-jail	jail(8) に関する議論
freebsd-java	Java™ 開発者や、FreeBSD へ JDK™ を移植する人たち
freebsd-kde	KDE および KDE アプリケーションの移植
freebsd-mips	FreeBSD の MIPS® への移植
freebsd-mono	FreeBSD における Mono および C# アプリケーション
freebsd-new-bus	バスアーキテクチャに関する技術的な議論
freebsd-net	ネットワークおよび TCP/IP ソースコードに関する議論
freebsd-numerics	高品質な libm 機能の実装に関する議論
freebsd-ocaml	FreeBSD 固有の OCaml に関する議論
freebsd-office	FreeBSD でのオフィスアプリケーションについて
freebsd-performance	ハイパフォーマンス / 高負荷での導入のためのパフォーマンスチューニングに関する質問
freebsd-perl	数多く存在する Perl に関連する port の管理について
freebsd-pkg	バイナリ package 管理および package ツールについての議論
freebsd-pf	パケットフィルタファイアウォールシステムに関する議論および質問
freebsd-pkg	バイナリ package 管理および package 関連ツールの議論
freebsd-pkg-fallout	package ビルドに失敗したログ
freebsd-pkgbase	FreeBSD ベースシステムの pkg 化
freebsd-platforms	Intel® 以外のアーキテクチャのプラットフォームへの移植
freebsd-ports	Ports Collection に関する議論

リスト	目的
freebsd-ports-announce	Ports Collection に関する重要なニュースと案内 (モデレータ制)
freebsd-ports-bugs	ports のバグや PR についての議論
freebsd-ppc	FreeBSD の PowerPC® への移植
freebsd-proliant	HP ProLiant サーバプラットフォーム上での FreeBSD に関する技術的な議論
freebsd-python	FreeBSD 固有の Python に関する話題
freebsd-rc	rc.d システムおよび開発に関連した議論
freebsd-realtime	FreeBSD 用のリアルタイム拡張の開発に関する話題
freebsd-riscv	FreeBSD の RISC-V® システムへの移植
freebsd-ruby	FreeBSD 固有の Ruby に関する議論
freebsd-scsi	SCSI サブシステム
FreeBSD security メーリングリスト	FreeBSD に影響するセキュリティに関する話題
freebsd-snapshots	FreeBSD 開発スナップショットのアナウンス
freebsd-sparc64	FreeBSD の SPARC® ベースシステムへの移植
freebsd-standards	C99 および POSIX® 標準への FreeBSD の適合について
freebsd-sysinstall	sysinstall(8) の開発
freebsd-tcltk	FreeBSD 固有の Tcl/Tk に関する議論
freebsd-testing	FreeBSD における試験
freebsd-tex	TeX および関連アプリケーションの FreeBSD への移植
freebsd-threads	FreeBSD のスレッドについて
freebsd-tokenring	FreeBSD でのトークンリングのサポート
freebsd-toolchain	FreeBSD の統合されたツールチェインのメンテナンス
freebsd-translators	FreeBSD 文書およびプログラムの翻訳
freebsd-transport	FreeBSD でのトランスポートレベルネットワークプロトコルに関する議論
freebsd-usb	FreeBSD の USB 対応に関する議論
freebsd-virtualization	FreeBSD によりサポートされているさまざまな仮想化技術についての議論
freebsd-vuxml	VuXML インフラストラクチャに関する議論
freebsd-wireless	802.11 スタック、ツールおよびデバイスドライバの開発に関する議論

リスト	目的
freebsd-x11	FreeBSD での X11 のメンテナンスとサポート
freebsd-xen	FreeBSD の Xen™ への移植 - 実装および利用についての議論
freebsd-xfce	XFCE の FreeBSD への移植や保守について
freebsd-zope	Zope の FreeBSD への移植や保守について

制限されているメーリングリスト: 以下のメーリングリストはより特化された (そしてより厳しい) メンバーのためのものであり、 一般的な興味を惹くようなものではありません。 このようなメーリングリストに参加する前に、 技術的なメーリングリストで自らの存在感をアピールするのは良い考えです。 そうすることにより、 議論の際のエチケットを学ぶことができるでしょう。

メーリングリスト	目的
freebsd-hubs	ミラーサイトを運営している人達 (基盤のサポート)
freebsd-user-groups	ユーザグループの調整
freebsd-wip-status	FreeBSD の進行中のプロジェクトに関する状況

メーリングリストのダイジェスト版:
上述のメーリングリストのすべてでダイジェスト版を利用できます。
メーリングリストに登録すると、アカウントのオプションセクションで、 ダイジェストのオプションを変更できます。

コミットメッセージリスト: 以下のメーリングリストは、 ソースツリーのさまざまな領域に対する変更に対するログメッセージを見ることに興味のある人向けです。

メーリングリスト	ソースの範囲	(ソースの) 範囲の説明
dev-commits-doc-all	/usr/doc	doc リポジトリへ加えられたすべての変更
dev-commits-ports-all	/usr/ports	ports リポジトリへ加えられたすべての変更
dev-commits-ports-main	/usr/ports	ports リポジトリの "main" ブランチに加えられたすべての変更
dev-commits-ports-branches	/usr/ports	ports リポジトリの四半期ごとのブランチに加えられたすべての変更
dev-commits-src-all	/usr/src	src リポジトリへ加えられたすべての変更

メーリングリスト	ソースの範囲	(ソースの) 範囲の説明
dev-commits-src-main	/usr/src	src リポジトリの "main" ブランチ (FreeBSD-CURRENT ブランチ) に加えられたすべての変更
dev-commits-src-branches	/usr/src	src リポジトリのすべての stable ブランチに加えられたすべての変更

SVN メーリングリスト: 以下のメーリングリストは、 ソースツリーのさまざまな領域に対する変更の SVN ログメッセージを見ることに興味のある人向けです。

SVN ログメッセージのみが SVN のメーリングリストに送られます。 SVN から Git への移行後は、以下のメーリングリストには、新しいコミットメッセージは送られませんし、購読もできません。
以下の一覧のリンクは、各メーリングリストのアーカイブへのリンクです。

メーリングリスト	ソースの範囲	(ソースの) 範囲の説明
svn-doc-all	/usr/doc	doc subversion リポジトリへ加えられたすべての変更 (user, projects および translations を除く)
svn-doc-head	/usr/doc	doc subversion リポジトリの "head" ブランチに加えられたすべての変更
svn-doc-projects	/usr/doc/projects	doc subversion リポジトリの projects に加えられたすべての変更
svn-doc-svnadmin	/usr/doc	doc subversion リポジトリの管理用スクリプト、フックおよび他のコンフィグレーションデータに対して加えられたすべての変更
svn-ports-all	/usr/ports	ports subversion リポジトリへ加えられたすべての変更
svn-ports-head	/usr/ports	ports subversion リポジトリの "head" ブランチに加えられたすべての変更
svn-ports-svnadmin	/usr/ports	ports subversion リポジトリの管理用スクリプト、フックおよび他のコンフィグレーションデータに対して加えられたすべての変更

メーリングリスト	ソースの範囲	(ソースの) 範囲の説明
SVN commit messages for the entire src tree (except for "user" and "projects")	/usr/src	src subversion リポジトリへ加えられたすべての変更 (user および projects を除く)
SVN commit messages for the src tree for head/-current	/usr/src	src subversion リポジトリの "head" ブランチ (FreeBSD-CURRENT ブランチ) に加えられたすべての変更
svn-src-projects	/usr/projects	src subversion リポジトリの projects に加えられたすべての変更
svn-src-release	/usr/src	src subversion リポジトリの releases に加えられたすべての変更
svn-src-releng	/usr/src	src subversion リポジトリの releng ブランチ (セキュリティ / リリースエンジニアリングブランチ) に加えられたすべての変更
svn-src-stable	/usr/src	src subversion リポジトリのすべての stable ブランチに加えられたすべての変更
svn-src-stable-6	/usr/src	src subversion リポジトリの stable/6 ブランチに加えられたすべての変更
svn-src-stable-7	/usr/src	src subversion リポジトリの stable/7 ブランチに加えられたすべての変更
svn-src-stable-8	/usr/src	src subversion リポジトリの stable/8 ブランチに加えられたすべての変更
9-stable ソースツリーの SVN コミットメッセージ	/usr/src	src subversion リポジトリの stable/9 ブランチに加えられたすべての変更
svn-src-stable-10	/usr/src	src subversion リポジトリの stable/10 ブランチに加えられたすべての変更

メーリングリスト	ソースの範囲	(ソースの) 範囲の説明
svn-src-stable-11	/usr/src	src subversion リポジトリの stable/11 ブランチに加えられたすべての変更
svn-src-stable-12	/usr/src	src subversion リポジトリの stable/12 ブランチに加えられたすべての変更
svn-src-stable-other	/usr/src	src subversion リポジトリの古い stable ブランチに加えられたすべての変更
svn-src-svnadmin	/usr/src	src subversion リポジトリの管理用スクリプト、フックおよび他のコンフィグレーションデータに対して加えられたすべての変更
svn-src-user	/usr/src	src subversion リポジトリの user に加えられた実験的なすべての変更
svn-src-vendor	/usr/src	src subversion リポジトリの vender に加えられたすべての変更

C.2.2. 参加方法

メーリングリストに参加するには、https://lists.freebsd.org で、希望のメーリングリストをクリックしてください。 表示されるページには、各メーリングリストに登録するために必要な手順が書かれています。

メーリングリストにメールを送るには、 listname@FreeBSD.org にメールを送ってください。すると、メーリングリストに登録されている世界中のメンバに再配布されます。

メーリングリストから登録を解除する場合は、メーリングリストで配信されているメールの最後にある URL をクリックしてください。 または、 listname+unsubscribe@FreeBSD.org にメールを送信することでも登録を解除できます。

技術的なメーリングリストでは、 技術的な議論を保つようにすることが重要です。もし、重要なアナウンスのみを受け取りたいのであれば、 FreeBSD announcements メーリングリストへの参加をお勧めします。 ここには、あまりたくさんのメールは流れません。

C.2.3. メーリングリストの憲章

すべて FreeBSD メーリングリストは誰でもそれらを利用することに固守しなければいけないという一定の簡単なルールがあります。 これらのルールに従わないと、結果として FreeBSD の Postmaster postmaster@FreeBSD.org から 2 回までは警告を受けます。 3 回違反すると、投稿者はすべての

FreeBSD のメーリングリストから削除され、
そのメーリングリストへのさらなる投稿から締め出されるでしょう。
これらのルールや対策が必要なのは残念です。
しかし、今日のインターネットはずいぶんいやらしい環境になっており、 一般の人々は、その (対策の)
メカニズムがいかにもろいかという事すら認識する事が出来ていないと思われます。

道標

- いかなる投稿記事もそのメーリングリストの基本的な憲章を守るべきです。
  そのメーリングリストが技術的な問題に関するものであれば、
  技術的な議論を含む投稿でなければなりません。 現在継続中の不適切な憲章やフレイムは、
  所属しているすべての人に対してメーリングリストの価値を下げてしまうだけですし、
  許される行為ではないでしょう。 とくに話題のない自由形式の議論に対しては FreeBSD chat メーリングリスト が自由に認可されているので、 かわりに使うべきでしょう。
- 一度に 3 つ以上のメーリングリストには決して投稿すべきではありません。 2
  つのメーリングリストには双方に明確な必要性がある場合にのみ投稿すべきです。
  どのリストに対しても、 (メーリングリストの)参加者は (複数のメーリングリストに)
  重複して参加しており、関連する部分が少ない (たとえば、 "-stable" と "-scsi")
  メーリングリストを除いては、 一度に複数のメーリングリストに投稿する理由は全くありません。 `Cc`
  に複数のメーリングリストが含まれたメッセージを受信した場合には、
  そのメールに返事を出す前に、 `Cc` の部分を編集してください。 元記事を書いたのが誰であっても、
  返信する方にもクロスポストの責任があります。
- ユーザであれ開発者であれ、(議論の中で) 個人を攻撃したり冒涜したりすることは許されません。
  個人的なメールを引用したり再投稿したりする許可をもらえなかったり、 もらえそうにない時に、
  それをおこなう ようなネチケット (訳注: ネットワークにおけるエチケット)
  に対するひどい違反は好まれませんが、 してはならないと特別に定められている
  わけではありません。 しかしながら、
  そのような内容がメーリングリストの憲章に沿う場合はほとんどありません。
  このため、メーリングリストの憲章に違反しているということだけで警告 (または禁止)
  に値するものと考えていいでしょう。
- FreeBSD 以外の関連する製品やサービスの広告は、 絶対に禁止し、spam
  による違反者が宣伝していることが明確であったら、 すぐに禁止します。

個々のメーリングリストの憲章:

FreeBSD ACPI メーリングリスト

ACPI および電源管理開発

freebsd-fortran

FreeBSD での Fortran

FreeBSD での Fortran に関連した ports の議論のためのメーリングリストです。 ラップトップから
HPC クラスタまで、 コンパイラ、ライブラリ、 科学および工学のアプリケーションが対象です。

FreeBSD announcements メーリングリスト

重要なイベント/マイルストン

これは、単にたまに発表される重要な FreeBSD

のイベントに関心がある人のためのメーリングリストです。 これは、スナップショットやその他のリリースについてのアナウンスを含みます。 そのアナウンスは新しいFreeBSD の機能のアナウンスを含んでいます。 ボランティア等の呼びかけがあるかもしれません。これは流通量の少ないメーリングリストで、 完全なモデレートメーリングリストです。

freebsd-arch

アーキテクチャと設計の議論

これは、FreeBSD のアーキテクチャに関する議論を行なうためのメーリングリストです。当然、その内容は原則的に技術的なものに限定されます。
このメーリングリストにふさわしい話題は以下のようなものです。

- 複数のカスタマイズされたビルドを同時に行うには、 ビルドシステムをどういじり直せばよいか
- VFS で Heidemann レイヤを動作させるには、 何を修正する必要があるか
- 同一のデバイスドライバを多数のバス、 アーキテクチャに共通で使えるようにするには、デバイスドライバインタフェースをどう改変すれば良いか
- ネットワークドライバの書き方

freebsd-bluetooth

FreeBSD 上での *Bluetooth*®

FreeBSD の Bluetooth® ユーザが集まるフォーラムです。デザイン、実装の詳細、パッチ、障害報告、開発進捗レポート、 機能の要求、Bluetooth®に関連したすべての事柄が対象です。

freebsd-bugbusters

障害報告の取り扱いに関する調整

このメーリングリストは、 バグマイスター、バグバスター、および他の障害報告データベースに純粋に興味を持っているグループの調整や議論についてのフォーラムです。 このメーリングリストは、個別のバグ、パッチ、障害報告について議論を行うためのものではありません。

freebsd-bugs

バグレポート

これは、FreeBSD のバグレポートのためのメーリングリストです。可能である場合はいつでも、バグは send-pr(1) を使うか、 web interfaceを用いて送られる必要があります。

freebsd-chat

FreeBSD のコミュニティに関する技術的ではない話題

このメーリングリストは技術的ではなく、 社会的な情報について、他のメーリングリストでは取り扱わない話題を含みます。 これは、Jordanがシロイタチに似ているかどうか、 大文字で打つかどうか、誰がたくさんコーヒーを飲むか、どこのビールが一番うまいか、 誰が地下室でビールを作っているか、などについての議論を含みます。時々重要なイベント(将来開催されるパーティーや、結婚式、誕生日、 新しい仕事など) のお知らせが、技術的なメーリングリストからでてきます。しかし、 フォローは直接 -

chatメーリングリストにするべきです。

freebsd-chromium

FreeBSD 固有の *Chromium* の問題

FreeBSD における Chromium のサポートについて議論を行うためのメーリングリストです。
Chromium の開発およびインストールに関して議論を行う技術的なメーリングリストです。

freebsd-cloud

さまざまなクラウドプラットフォームでの *FreeBSD* の実行

このメーリングリストでは、FreeBSD を Amazon EC2, Google Compute Engine, Microsoft Azure およびその他のクラウドコンピューティングプラットフォームでの使用について議論を行います。

コアチーム

FreeBSD コアチーム

これは、コアメンバが使う内部メーリングリストです。 FreeBSD に関連する深刻なやっかい事の裁定やハイレベルな綿密な調査を要求するときに、このメーリングリストにメッセージを送る事が出来ます。

FreeBSD-CURRENT メーリングリスト

FreeBSD-CURRENT の使用に関する議論

これは FreeBSD-CURRENT のユーザのためのメーリングリストです。 メーリングリストでの話題は、-CURRENT で登場した新しい機能について、その新機能によってユーザに影響することについての注意、 および -CURRENT のままでいるために必要な手順についての説明を含みます。 "CURRENT" を走らせている人はこのメーリングリストに登録しなくてはなりません。
これは技術的なメーリングリストなので、 完全に技術的な内容を要求します。

freebsd-desktop

デスクトップでの *FreeBSD* の利用や改良について

デスクトップでの FreeBSD について議論を行うためのフォーラムです。
主として、デスクトップの移植者やユーザが FreeBSD のデスクトップサポートに関する問題点や改良について議論する場です。

dev-ci

ビルドおよび試験結果の継続的インテグレーションレポート

ビルドおよび試験結果に関するすべての継続的インテグレーションのレポート

dev-reviews

FreeBSD レビューツールの進行中の作業の通知

パッチを含む FreeBSD レビューツールによる進行中のレビュー作業の自動通知

freebsd-doc

ドキュメンテーションプロジェクト

このメーリングリストは FreeBSD 向けの文書の作成に関連する事柄やプロジェクトについて議論を行なうためのものです。 このメーリングリストに参加しているメンバは、 "FreeBSD ドキュメンテーションプロジェクト" に参加していることになります。 このメーリングリストは公開されているので、参加や投稿は自由に行なうことができます。

freebsd-drivers

FreeBSD のデバイスドライバの書き方について

このメーリングリストは、FreeBSD のデバイスドライバに関連した技術的なフォーラムです。主にデバイスドライバを書く人たちが、FreeBSD カーネルの API を使ったデバイスドライバの書き方について質問を行う場です。

freebsd-dtrace

FreeBSD における *Dtrace* の利用と開発

DTrace は、カーネルおよびユーザ空間のプログラムを実行時に解析するためのフレームワークを提供するもので、FreeBSD に統合されています。 このメーリングリストは、コードの開発者および利用者の議論のアーカイブです。

freebsd-eclipse

Eclipse IDE, ツール、 リッチクライアントアプリケーションの *FreeBSD* ユーザおよび *ports*

このメーリングリストの目的は、FreeBSD プラットフォームでの Eclipse IDE、ツール、リッチクライアントアプリケーションについて、選択、インストール、利用、開発および管理に関係するすべての相互支援を提供すること、そして Eclipse IDE およびプラグインの FreeBSD 環境への移植を手助けすることです。

このメーリングリストのもう一つの目的は、Eclipse コミュニティと FreeBSD コミュニティが相互に利益になるような情報交換の場を提供することです。

このメーリングリストは、主に Eclipse ユーザのニーズに焦点が当てられていますが、Eclipse フレームワークを用いた FreeBSD アプリケーションの開発に関わる方々のフォーラムにもなっています。

freebsd-embedded

組み込みアプリケーションにおける *FreeBSD* の利用

このメーリングリストは、組み込みシステムにおける FreeBSD の利用に関するトピックを議論するためのものです。 これは技術的なメーリングリストなので、完全に技術的な内容を要求します。 このメーリングリストにおいて、組み込みシステムは、デスクトップや通常の一般的なコンピュータ環境ではなく、単一の目的のために使われるコンピュータデバイスを意味します。 これらの例は、携帯電話、ルータやスイッチおよび PBX といったネットワーク機器、PDA, POS システムといったものです。

freebsd-emulation

Linux/MS-DOS®/Windows® 等の他のシステムのエミュレーション

他のオペレーティングシステムに書かれたプログラムを、FreeBSD で走らせることに関連した技術的な議論のためのフォーラムです。

freebsd-enlightenment

Enlightenment

FreeBSD システムでの Enlightenment デスクトップ環境に関連した議論。
技術的なメーリングリストなので、 完全に技術的な内容が要求されます。

freebsd-firewire

FireWire® (iLink, IEEE 1394)

このメーリングリストは、FreeBSD における FireWire® (IEEE 1394, iLink)
サブシステムの設計と実装について議論を行うためのものです。
標準化、バスデバイスとそのプロトコル、 アダプタボード/カード/チップセット、そして、
それらに適切に対応するためのアーキテクチャとコードの実装が特に関連するトピックです。

freebsd-fs

ファイルシステム

FreeBSD のファイルシステムに関する議論。 これは技術的なメーリングリストなので、
完全に技術的な内容を要求します。

freebsd-games

FreeBSD のゲーム

FreeBSD にゲームを持ち込むことに関連した議論を行うメーリングリストです。 活発にゲームを
FreeBSD に移植する作業を行っている方や、 問題を提起したり、
その他の解決方法を議論したりする方のためのものです。
技術的な議論に興味のある方の参加も歓迎されます。

freebsd-gecko

Gecko レンダリングエンジン

FreeBSD を使った Gecko アプリケーションについてのフォーラムです。

このメーリングリストでは、Gecko Ports アプリケーション、インストール、開発および FreeBSD
でのサポートといった話題を中心に議論が行われます。

freebsd-geom

GEOM

GEOM および関連した実装に関する議論。 これは技術的なメーリングリストなので、
完全に技術的な内容が要求されます。

freebsd-git

FreeBSD での *git* の使用

FreeBSD のプロジェクトコラボレーションおける github ミラーおよびその他の git の使用など、git
インフラストラクチャをどのように使うかといった議論が行われます。 FreeBSD github ミラーから
git を使用する方々が議論に参加しています。 ミラーの使用を考えている方や、git の一般的な
FreeBSD での使用を考えている方はここで質問できます。

freebsd-gnome

GNOME

FreeBSD の GNOME Desktop Environment に関する議論。
これは技術的なメーリングリストなので、 完全に技術的な内容を要求します。

freebsd-infiniband

FreeBSD での Infiniband の使用

FreeBSD における Infiniband, OFED および OpenSM に関する技術的なメーリングリストです。

freebsd-ipfw

IP Firewall

これは FreeBSD の IP firewall コードの再設計に関する技術的な議論のためのフォーラムです。
これは技術的なメーリングリストなので、 完全に技術的な内容を要求します。

freebsd-isdn

ISDN コミュニケーション

このメーリングリストは、 FreeBSD に対する ISDN
サポートの開発の議論をおこなう人のためのものです.

freebsd-java

Java™ の開発

このメーリングリストは、FreeBSD 向けの重要な Java™ アプリケーションの開発や、JDK™
の移植やメンテナンスの議論をする人のためのものです.

freebsd-jobs

求人情報および就職希望情報

FreeBSD に関連した就職情報、および FreeBSD
に関連した職業を探している方が履歴書を投稿するためのフォーラムです。
このメーリングリストは一般的な就職情報のためのものでは ありません。
一般的な就職情報については、 既に別な場所に適切なフォーラムがあるので、
そちらに投稿してください。

他の FreeBSD.org メーリングリスト同様に、 このメーリングリストは全世界に配信されます。
地域に関する情報や、
在宅勤務なのか移転のための支援を受けられるかどうかを明確にしてください。

メールでは、オープンフォーマットのみを使う必要があります。 -
プレインテキストが好ましいのですが、 多くの読者は、 Portable Document Format (PDF), HTML
および他にもいくつかのフォーマットを使用できるでしょう。 Microsoft® Word (.doc)
のようなクローズドフォーマットは、 メーリングリストのサーバにより拒否されてしまいます。

freebsd-kde

KDE

FreeBSD システムにおける KDE に関する議論。 これは技術的なメーリングリストなので、

完全に技術的な内容を要求します。

freebsd-hackers

技術的な議論

これは FreeBSD に関する技術的な議論のためのフォーラムです。
これは最もテクニカルなメーリングリストです。 このメーリングリストは、FreeBSD
上でアクティブに活動をしている人のためのもので、
問題を持ち出したり、代わりの解決法を議論します。
技術的な議論をフォローするのに興味がある人も歓迎します。
これは技術的なメーリングリストなので、 完全に技術的な内容を要求します。

freebsd-hardware

FreeBSD のハードウェアの一般的な議論

FreeBSD が走っているハードウェアのタイプや、
何を買ったり避けたりするかに関する様々な問題や、 提案に関する議論。

freebsd-hubs

ミラーサイト

FreeBSD ミラーサイトを運用している人達向けの、
アナウンスと議論を行なうメーリングリストです。

freebsd-isp

インターネットサービスプロバイダのについての話題

このメーリングリストは、 FreeBSD を用いたインターネット サービスプロバイダ (ISP)
に関する話題の議論のためのものです。 これは技術的なメーリングリストなので、
完全に技術的な内容を要求します。

freebsd-mono

FreeBSD における *Mono* および *C#* アプリケーション

FreeBSD 上での Mono 開発フレームワークに関連した議論を行うためのメーリングリストです。
これは、技術的なメーリングリストです。 Mono または C# アプリケーションの FreeBSD
への移植を活発に行っている方が、
問題を提起したり、他の解決方法について議論を行うためのものです。
技術的な議論に興味を持っている方の参加も歓迎されます。

freebsd-ocaml

FreeBSD 固有の *OCaml* に関する議論

このメーリングリストは、FreeBSD における OCaml に関する議論の場です。
このメーリングリストは技術的なメーリングリストです。 OCaml
port、サードパーティ製ライブラリ、
およびフレームワークに関して作業を行っている個人のためのものです。
技術的な議論に興味を持っている個人の参加が歓迎されます。

freebsd-office

FreeBSD でのオフィスアプリケーション

FreeBSD におけるオフィスアプリケーションのインストール、開発およびサポートについての議論の場です。

freebsd-ops-announce

プロジェクトのインフラストラクチャに関するアナウンス

FreeBSD.org プロジェクトのインフラストラクチャの変更や関連した問題について興味を持っている向けのメーリングリストです。

このモデレートメーリングリストは、アナウンスに制限されています (返答や要求、議論、意見を述べる場ではありません)。

freebsd-performance

FreeBSD のチューニングや速度向上に関する議論

このメーリングリストは、ハッカー、管理者および関連グループが、FreeBSD のパフォーマンスに関するトピックを議論する場です。
このメーリングリストで議論されるべきトピックは、高負荷における FreeBSD の導入において経験するパフォーマンスの問題や FreeBSD の限界に挑むような話題を含みます。
FreeBSD のパフォーマンスを改善したいと考えている方は、このメーリングリストに登録してください。このメーリングリストは、FreeBSD の高速化、堅牢さ、拡張性に興味をもっている、経験のある FreeBSD ユーザ、ハッカー、管理者向けの高度な技術的メーリングです。
ドキュメントに目を通さずに質問して答えを求めるような Q and A タイプのメーリングリストではなく、
未解決で答えのないパフォーマンスに関連したトピックへの貢献や問い合わせの場です。

freebsd-pkg

バイナリ *package* 管理および *package* ツールについての議論

ソフトウェアのインストールにバイナリ package を用いる FreeBSD システム管理のすべての側面に関する議論。 バイナリ package のツールキットとフォーマット、それらの FreeBSD における開発とサポート、 package リポジトリ管理そしてサードパーティ製 package を含みます。

package の作成に失敗する ports に関する議論は ports の問題として考えるべきであり、このメーリングリストで議論することは適切ではありません。

freebsd-pf

パケットフィルタファイアウォールシステムに関する議論および質問

FreeBSD のパケットフィルタ (pf) ファイアウォールシステムに関連した議論。技術的な議論およびユーザによる質問の両方が歓迎されます。 このメーリングリストは、ALTQ QoS フレームワークについて議論する場でもあります。

freebsd-pkg-fallout

package ビルドに失敗したログ

package ビルドクラスタにおいて package ビルドに失敗したすべてのログ

freebsd-pkgbase

FreeBSD ベースシステムの *pkg* 化

FreeBSD ベースシステムの pkg 化に関する実装および課題についての議論。

freebsd-platforms

Intel® 以外のプラットフォームへの移植

クロスプラットフォームの FreeBSD の問題。Intel® 以外のプラットフォームへの FreeBSD の移植についての一般的な議論や提案。これは技術的なメーリングリストなので、完全に技術的な内容を要求します。

freebsd-ports

"ports" の議論

FreeBSD "Ports Collection" (/usr/ports) に関連する話題や、Ports Collection の基盤および ports の一般的な構成の整備活動に関する議論。これは技術的なメーリングリストなので、厳密に技術的な内容のみが扱われます。

freebsd-ports-announce

FreeBSD "Ports Collection" に関する重要なニュースと案内

"Ports Collection" (/usr/ports) の開発者、ports 作成者およびユーザへの重要なニュース。アーキテクチャ/インフラストラクチャの変更、新しい機能、重要なアップグレードの案内、そしてリリースエンジニアリング情報が扱われます。このメーリングリストの流量は少なく、アナウンスを目的としたものです。

freebsd-ports-bugs

"ports" のバグに関する議論

"Ports Collection" (/usr/ports) の障害報告や新たな ports や変更についての提案についての議論。これは技術的なメーリングリストなので、厳密に技術的な内容のみが扱われます。

freebsd-proliant

HP ProLiant サーバプラットフォーム上での *FreeBSD* に関する技術的な議論

このメーリングリストは、HP ProLiant サーバ上での FreeBSD の利用に関した技術的な議論に用いられます。ProLiant に特有のドライバ、管理ソフトウェア、設定ツール、および BIOS アップデートなどが含まれます。hpasmd, hpasmcli および hpacucli モジュールについて議論する主要な場です。

freebsd-python

FreeBSD における *Python*

FreeBSD における Python サポートの改良に関連した議論を行うためのメーリングリストです。これは技術的なメーリングリストです。Python の移植に関する作業を行っている方や、サードパーティ製モジュールおよび Zope を FreeBSD に移植している方を対象としたメーリングリストです。
技術的な議論に興味を持っている方の参加も歓迎されます。

freebsd-questions

ユーザからの質問

FreeBSD に関する質問のためのメーリングリストです。 技術的なメーリングリストに対しては、極めて技術的な質問でなければ、 "どのようにして" という質問を送るべきではありません。

freebsd-ruby

FreeBSD 固有の *Ruby* に関する議論

FreeBSD での Ruby サポートに関連した議論を行うためのメーリングリストです。これは技術的なメーリングリストです。Ruby ports, サードパーティライブラリおよびフレームワークについて作業を行っている人達を対象としています。

技術的な議論に興味を持つ方の参加も歓迎されます。

freebsd-scsi

SCSI サブシステム

これは FreeBSD のための SCSI サブシステムについて作業している人向けです。これは技術的なメーリングリストなので、 完全に技術的な内容を要求します。

FreeBSD security メーリングリスト

セキュリティの関連の話題

FreeBSD コンピュータのセキュリティの話題 (DES, Kerberos, よく知られているセキュリティホールや、 それらのふさぎ方など) これは技術的なメーリングリストなので、 完全に技術的な議論を要求します。 これは、Q and A のメーリングリストではありません。 FAQ に対する Q and A (質問と答えの両方) の貢献は、歓迎されます。

FreeBSD security notifications メーリングリスト

セキュリティ関連の通知

FreeBSD のセキュリティ問題や、 修正に関する通知を行ないます。このメーリングリストは議論を行なうためのメーリングリストではありません。 議論は FreeBSD-security で行ないます。

freebsd-snapshots

FreeBSD 開発スナップショットのアナウンス

このメーリングリストは、head/ および stable/ ブランチからの新しい FreeBSD 開発スナップショットのアナウンスを行います。

FreeBSD-STABLE; メーリングリスト

FreeBSD-STABLE の使用に関する議論

これは FreeBSD-STABLE のユーザ用のメーリングリストです。 "STABLE" は、 RELEASE 後もバグフィックスおよび新しい機能の追加など、 開発が続いているブランチです。バイナリ互換性のため、ABI は安定するように維持されます。 メーリングリストでの話題は、 -STABLE で登場した新しい機能について、その新機能によってユーザに影響することについての注意、 および -STABLE

のままでいるために必要な手順についての説明を含みます。 "STABLE"
を走らせている人はこのメーリングリストに登録すべきです。
これは技術的なメーリングリストなので、 完全に技術的な内容を要求します。

freebsd-standards

C99 POSIX への適合

C99 および POSIX 標準への FreeBSD の適合に関連した技術的な議論を行うためのフォーラムです。 ::
FreeBSD による教育

FreeBSD による教育について議論を行うための技術的ではないメーリングリストです。

freebsd-testing

FreeBSD における試験

ATF/Kyua、ビルド試験のインフラストラクチャ、 他のオペレーティングシステム（NetBSD, ...）から FreeBSD への移植に関する試験など、FreeBSD
の試験に関して議論を行う技術的なメーリングリストです。

freebsd-tex

TeX および関連アプリケーションの *FreeBSD* への移植

TeX および関連アプリケーションの FreeBSD
への移植について議論する技術的なメーリングリストです。 TeX の FreeBSD
への移植作業を活発に行っている個人が、
問題を提起したり、他の解決策について議論するためのものです。
技術的な議論に興味を持っている個人の参加も歓迎されます。

freebsd-toolchain

FreeBSD の統合されたツールチェインのメンテナンス

FreeBSD のツールチェインのメンテナンスに関連した議論を行うためのメーリングリストです。
Clang および GCC の状況についての議論の他に、アセンブラ、
リンカおよびデバッガ等のソフトウェアの議論も行われます。

freebsd-transport

FreeBSD でのトランスポートレベルネットワークプロトコルに関する議論

The transport mailing list exists for the discussion of issues and designs around the transport level protocols in the FreeBSD network stack, including TCP, SCTP and UDP. TCP, SCTP および UDP などの FreeBSD
ネットワークスタックのトランスポートレベルプロトコルに関する問題や設計についての議論を行うためのメーリングリストです。
ドライバ特有の話題であったりネットワークプロトコルなどの他のネットワークに関するトピックは、
で議論してください。

freebsd-translators

FreeBSD 文書およびプログラムの翻訳

FreeBSD 文書を英語から他の言語へと翻訳を行っている方々が、
翻訳方法やツールについて議論を行うメーリングリストです。 新しいメンバーは、自己紹介と、

興味のある翻訳言語をお知らせください。

freebsd-usb

FreeBSD の USB 対応に関する議論

これは、FreeBSD の USB 対応に関連した議論を行うメーリングリストです。

freebsd-user-groups

ユーザグループの調整のメーリングリスト

これは、ローカルなユーザグループがお互いに、または、
コアチームが指定した個人と問題を議論する、
それぞれのローカルエリアのユーザグループからの調整人向けのメーリングリストです。
このメーリングリストはユーザグループ間のミーティングの概要やプロジェクトの調整に制限されるべきです。

freebsd-xfce

XFCE

これは、XFCE 環境を FreeBSD へ移植することを議論する、技術的なメーリングリストです。 活発に XFCE を FreeBSD に移植する作業を行なっている人に向けたもので、
問題を提起したり、新しい解決法を議論することを目的としています。
技術的な議論に興味を持っている方の参加も歓迎します。

freebsd-zope

Zope

これは、Zope 環境を FreeBSD へ移植することを議論する、技術的なメーリングリストです。 活発に Zope を FreeBSD に移植する作業を行なっている人に向けたもので、
問題を提起したり、新しい解決法を議論することを目的としています。
技術的な議論に興味を持っている方の参加も歓迎します。

freebsd-virtualization

FreeBSD によりサポートされているさまざまな仮想化技術についての議論

FreeBSD によりサポートされているさまざまな仮想化技術に関する議論。
新しい機能および基本的な機能の実装に焦点を当てる一方で、
仮想化技術の使用の際に問題が起きた場合の手助けや議論のフォーラムでもあります。

freebsd-wip-status

FreeBSD の進行中のプロジェクトの状況

このメーリングリストは、開発者が FreeBSD に関連したプロジェクトの立ち上げや進捗状況のアナウンスに利用するためのものです。
メールはモデレータ制です。 "To:" として最も適切な FreeBSD のメーリングリストを入れ、
このメーリングリストを "BCC:" に入れることが推奨されます。
このメーリングリスト上では、議論が許されていないため、 このように送信することで、
進捗状況の議論を別の適切なメーリングリストで議論できるようになります。

どのようなメールが適切かについては、 アーカイブで確認してください。

このメーリングリストへのメッセージの編集ダイジェスト版が、 進捗状況レポート

として、数ヶ月おきに FreeBSD ウェブサイトに公開されます。
過去のレポートもアーカイブされています。

freebsd-wireless

802.11 スタック、 ツールおよびデバイスドライバの開発に関する議論

FreeBSD のワイヤレスに関するメーリングリストです。
バグ、新しい機能およびメンテナンスについての議論を含む、 802.11 スタック (sys/net80211)、
デバイスドライバおよびツールの開発に焦点が当てられています。

freebsd-xen

FreeBSD の Xen™ への移植 - 実装および利用についての議論

このメーリングリストは、FreeBSD の Xen™ への移植に焦点が当てられています。
このメーリングリストのトラフィックは小さいので、
技術的な議論およびデザインの詳細と管理上の問題の両方についてのフォーラムとして期待されています。

C.2.4. メーリングリストのフィルタリング

FreeBSD のメーリングリストは、スパム、
ウィルスおよび他の不要なメールを配布してしまわないよう、
いくつかの方法でフィルタリングを行なっています。 この節で説明するフィルタリングは、
メーリングリストを守るために使われている方法のすべてというわけではありません。

メーリングリストでは、以下の添付ファイルを送ることができます。 以下の一覧以外の MIME content type の添付ファイルを含むファイルは、 メーリングリストに流れる前に取り除かれます。

- 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

他の MIME content type の添付を許可するメーリングリストもありますが、
上の一覧に含まれるものであれば、 ほとんどのメーリングリストで適用できます。

HTML と plain テキストを両方含むメールでは、 HTML の部分が削除されます。 HTML のみを含むメールは、plain テキストに変換されます。

C.3. Usenet ニュースグループ

2 つの FreeBSD 用のニュースグループに加え、他にも FreeBSD の議論をしたり FreeBSD に関連するユーザがいるニュースグループがたくさんあります。

C.3.1. BSD 用のニュースグループ

- comp.unix.bsd.freebsd.announce
- comp.unix.bsd.freebsd.misc
- de.comp.os.unix.bsd (ドイツ)
- fr.comp.os.bsd (フランス)

C.3.2. 関連する他の UNIX® のニュースグループ

- comp.unix
- comp.unix.questions
- comp.unix.admin
- comp.unix.programmer
- comp.unix.shell
- comp.unix.misc
- comp.unix.bsd

C.3.3. X Window システム

- comp.windows.x

C.4. オフィシャルミラー

Central Servers, Armenia, Australia, {mirrors-austria}, Czech Republic, Denmark, Finland, France, Germany, Hong Kong, Ireland, Japan, Latvia, Lithuania, Netherlands, Norway, Russia, Slovenia, South Africa, Spain, Sweden, Switzerland, Taiwan, United Kingdom, United States of America.

Central Servers

- https://www.FreeBSD.org/

Armenia

- http://www1.am.FreeBSD.org/ (IPv6)

Australia

- http://www.au.FreeBSD.org/
- http://www2.au.FreeBSD.org/

{mirrors-austria}

- http://www1.am.FreeBSD.org/ (IPv6)

Czech Republic

- http://www.cz.FreeBSD.org/ (IPv6)

Denmark

- http://www.dk.FreeBSD.org/ (IPv6)

Finland

- http://www.fi.FreeBSD.org/

France

- http://www1.fr.FreeBSD.org/

Germany

- http://www.de.FreeBSD.org/

Hong Kong

- http://www.hk.FreeBSD.org/

Ireland

- http://www.ie.FreeBSD.org/

Japan

- http://www.jp.FreeBSD.org/www.FreeBSD.org/ (IPv6)

Latvia

- http://www.lv.FreeBSD.org/

Lithuania

- http://www.lt.FreeBSD.org/

Netherlands

- http://www.nl.FreeBSD.org/

Norway

- http://www.no.FreeBSD.org/

Russia

- http://www.ru.FreeBSD.org/ (IPv6)

Slovenia

- http://www.si.FreeBSD.org/

South Africa

- http://www.za.FreeBSD.org/

Spain

- http://www.es.FreeBSD.org/
- http://www2.es.FreeBSD.org/

Sweden

- http://www.se.FreeBSD.org/

Switzerland

- http://www.ch.FreeBSD.org/ (IPv6)
- http://www2.ch.FreeBSD.org/ (IPv6)

Taiwan

- http://www.tw.FreeBSD.org/
- http://www2.tw.FreeBSD.org/
- http://www4.tw.FreeBSD.org/
- http://www5.tw.FreeBSD.org/ (IPv6)

United Kingdom

- http://www1.uk.FreeBSD.org
- http://www3.uk.FreeBSD.org/

United States of America

- http://www5.us.FreeBSD.org/ (IPv6)

付録 D: PGP 公開鍵

FreeBSD.org オフィサの PGP 公開鍵を以下に示します。 これらの公開鍵は、署名を検証したり、オフィサに暗号メールを送る必要がある場合に使用できます。 すべての FreeBSD 公開鍵の一覧は、 PGP Keys にあります。 また、完全なキーリングは pgpkeys.txt からダウンロードできます。

D.1. オフィサ

D.1.1. セキュリティオフィサチーム <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+gbE5Hg6dvfttt RXD
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. コアチーム書記 <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. ports 管理チーム書記 <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-----
```