FreeBSD 2.X、3.X、4.X についての FAQ (よくある質問とその答え)

概要

この文書は FreeBSD システム・バージョン 2.X、3.X、4.X についての FAQ です。特に断わりがない限り、どの項目も FreeBSD 2.0.5 以降のものを想定しています。 <XXX> のついている項目はまだ作業中のものです。 この FreeBSD ドキュメンテーションプロジェクトに協力したいと思われる方は、 FreeBSD documentation project メーリングリスト まで (英語で) 電子メールを送ってください。 この文書の最新バージョンは、いつでも日本国内版 FreeBSD World Wide Web サーバや FreeBSD World Wide Web サーバで見ることができます。 また、ひとつの巨大な HTML ファイルとして HTTP でダウンロードすることもできます。 プレーンテキスト、PostScript、PDF、およびその他の形式のものは FreeBSD FTP サーバに置かれています。 また、FAQ の検索も可能です。



2005 年 6 月現在、HTML 版以外の日本語 FAQ は用意されていません。

日本語版の作成は FreeBSD 日本語ドキュメンテーションプロジェクトがオリジナルの英語版をもとにして行なっています。 FreeBSD FAQ 日本語訳および、 FreeBSD FAQ 日本語版のみに関連することは、 日本語ドキュメンテーションプロジェクト <doc-jp@jp.FreeBSD.org> において日本語で議論されています。 必要に応じて日本語ドキュメンテーションプロジェクトから、FreeBSD Documentation Project に対してフィードバックを行ないますので、 英語が得意でない方は日本語ドキュメンテーションプロジェクト <doc-jp@jp.FreeBSD.org> まで日本語でコメントをお寄せください。

また、この FreeBSD FAQ とは別に、日本の FreeBSD ユーザ有志によって FreeBSD users-jp メーリングリスト <FreeBSD-users-jp@jp.FreeBSD.org> やニュースグループ fj.os.bsd.freebsd などへの投稿をもとに作成された QandA が公開されています。特に日本語環境など日本固有の話題が充実していますので、 こちらも合わせてご覧ください。

目次

まえがき . . . 3
1. インストール . . . 14
2. ハードウェアコンパチビリティ . . . 30
3. トラブルシューティング . . . 43
4. 商用アプリケーション . . . 58
5. ユーザアプリケーション . . . 61
6. カーネルコンフィグレーション . . . 65
7. システム管理 . . . 67
8. X Window System と仮想コンソール . . . 94
9. ネットワーキング . . . 107
10. PPP . . . 117
11. シリアル接続 . . . 133
12. その他の質問 . . . 142
13. まじめな FreeBSD ハッカーだけの話題 . . . 152
14. 謝辞 . . . 162
15. FreeBSD FAQ 日本語化について . . . 163
有用な書籍 . . . 165

まえがき

FreeBSD 2.X-4.X FAQ へようこそ!

Usenet の FAQ がそうであるように、 この文書も FreeBSD オペレーティングシステムに関して頻繁に尋ねられる質問を網羅することを目的としています (もちろんそれに対する答えも!)。 FAQ は本来バンド幅を減らし、 同じ質問が何度も繰り返されるのを避けるために作られたものですが、最近は有用な情報源と見なされるようになってきました。

この FAQ をできる限り有用なものにしようと、 あらゆる努力がはらわれています。もし何かしらの改善案が浮かんだら、ぜひ FAQ 管理者 <faq@FreeBSD.org> までメールを送ってください。

FreeBSD って何?

FreeBSD とは一言で言えば、カリフォルニア大学バークレイ校から リリースされた "4.4BSD-Lite" と "4.4BSD-Lite2" による 強化の一部に由来する、 i386 および Alpha/AXP 系のプラットフォーム向けの UN*X ライクなオペレーティングシステムです。 間接的には同じバークレイ校の "Net/2" を William Jolitz が i386 系に移植した "386BSD" も基にしていますが、 386BSD のコードはほとんど残っていません。 FreeBSD についての詳細と、何ができるかについては FreeBSD のホームページ を参照してください。

FreeBSD は企業やインターネットサービスプロバイダ、研究者、コンピュータ専門家、学生、家庭のユーザなどにより、業務や教育、
娯楽に用いられています。これらに関しては FreeBSD ギャラリーをご覧ください。

FreeBSD に関するより詳しい情報は FreeBSD ハンドブックを参照してください。

FreeBSD が目指しているもの

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、 これからも多少の代償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残らずすべての人々に、 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、 多少制限が科されています。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です。 ただし GPL ソフトウェアを商用で利用する場合、 さらに複雑になるのは避けられません。そのため、それらのソフトウェアを、より制限の少ない BSD 著作権に従ったソフトウェアで置き換える努力を、 可能な限り日々続けています。



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。

どうして FreeBSD と呼ばれているのですか?

- 無料 (free) で使うことができる (商利用も含む)。
- オペレーティングシステムの完全なソースコードが自由 (freely) に手に入り、商利用・非商利用にかかわらず、最低限の制限で他の仕事への利用、配布、導入が可能。
- 改良やバグフィックスがある場合、誰でも (free) そのコードを提出でき、ソースツリーに加えることができます (いくつかの簡単な条件には従ってもらいます)。

母国語が英語でない読者のために、ここでは "free" という単語が二つの意味で用いられていることを指摘しておくと分かりやすいかも知れません。ひとつは「無料である」ということ、もうひとつは「自分のやりたいようにできる」ということです。FreeBSD のコードでできないいくつかのこと (自分が書いたものだと偽るなど) を除けば、あなたは自分のやりたいことをやることが可能なのです。

FreeBSD の最新バージョンは?

4.3 が最新の *STABLE* バージョンで、2001 年 4 月にリリースされました。また、これは最新の *RELEASE* バージョンでもあります。

簡単に言ってしまうと、*-STABLE* は最新の *-CURRENT* のスナップショットのすばらしい新機能の数々よりも、安定性と変更回数の少なさを好む ISP や、他の企業のユーザをターゲットにしています。リリースはこの二種類のブランチで行なわれますが、(*-STABLE* と比較すると多少) 不安定な動作があるということを許容できるなら、必要となるのは *-CURRENT* の方だけでしょう。

各リリースは数カ月毎にしか行なわれません。多くの人々が FreeBSD のソースをそのリリースよりも最新の状態に維持している (FreeBSD-current と FreeBSD-stable に関する質問も参照してください) のですが、ソースというのは常に改変され続けているため、そうすることは一種の慣例になっています。

FreeBSD-CURRENTって何?

FreeBSD-CURRENT はオペレーティングシステムの開発バージョンで、やがて 5.0-RELEASE となります。よってこれは、そこに携わっている開発者や、どんな障害をも乗り越えていけるタフな愛好家たちにとってのみ興味の対象となるものです。-CURRENT の使用に際しての詳細は FreeBSD ハンドブック の 関連するセクション を参照してください。

オペレーティングシステムに馴染みがない場合や、それが一時的に発生している問題なのか、それとも本質的な問題かを見極める能力がない場合は、FreeBSD-CURRENT を使うべきではありません。このブランチは時々急激に拡張されたり、システムが構築できない状態になることもしょっちゅうあります。FreeBSD-CURRENT を使う人は問題を分析し、「小さな欠陥」ではなく、明らかに間違いであると思われるものだけを報告できるものと想定されています。「make world したら group 関係でエラーがでました」のような質問は、-CURRENT メーリングリストでは軽蔑の眼差しであしらわれることもあります。

毎日、その時点の -CURRENT と -STABLE のコードを元に snapshot が作成されています。現在は、その snapshot の配布も利用可能です。それぞれの snapshot には以下のような目的があります。

- インストールプログラムの最新版のテスト。
- 試してみたいけれど、 基礎的な所から毎日変わるようなものを追いかける時間もバンド幅も無い、という人にも -CURRENT や -STABLE を使えるようにする。また、そのような人たちのシステム移行のための手っ取り早い方法を提供する。
- あとでとんでもないことをしてしまった時のために、問題となるコードの特定の参照基準点を保存しておく。 (通常は CVS がこういうハプニングのような恐ろしい事態を防止して いるんですけどね :)
- テストが必要な新しい機能を、 できる限り多くの隠れテスターに試してもらう。

どんな目的であれ、-CURRENT snapshot が "製品レベルの品質" であるとの考えに基づく要求は行わないでください。 安定性やテスト十分性にこだわる人は、完全なリリース、あるいは -STABLE snapshot から離れてはいけません。

スナップショットリリースは、5.0-CURRENT が ftp://current.FreeBSD.org/pub/FreeBSD/ から、4-STABLE が releng4.FreeBSD.org から直接入手可能です。 また、3-STABLE スナップショットは、この文章の執筆時点 (2000 年 5 月) で作成されていません。

スナップショットリリースは、 現在、開発や保守作業が行なわれているすべてのブランチにおいて、平均して一日一回作成されます。

FreeBSD-STABLE のコンセプトは何ですか?

FreeBSD 2.0.5 がリリースされた後、私たちは FreeBSD の開発を 2 系統に分割することにしました。一つは -STABLE というブランチで、バグの修正はしっかりテストされ、機能の強化は少しずつしか行われません (急な変更や実験的機能を望まない、インターネットサービスプロバイダや営利企業向け)。 もう一方のブランチは -CURRENT で、2.0 がリリースされて以来 5.0-RELEASE (そしてその後も) へ向けて脈々と続いているものです。 ASCII で描いた簡単な図がわかりやすいかは自信がありませんが、 こんな感じになります。

```
               2.0
                |
                |
                |  [2.1-STABLE]
*BRANCH*       2.0.5 -> 2.1 -> 2.1.5 -> 2.1.6 -> 2.1.7.1  [2.1-STABLE 終了]
                |                          (1997/03)
                |
                |
                |  [2.2-STABLE]
*BRANCH*       2.2.1 -> 2.2.2-RELEASE -> 2.2.5 -> 2.2.6 -> 2.2.7 -> 2.2.8 [終了]
                |         (1997/03)     (1997/10)   (1998/04)   (1998/07)   (1998/12)
                |
                |
              3.0-SNAPs  (1997 年第一四半期開始)
                |
                |
              3.0-RELEASE (1998/10)
                |
                |  [3.0-STABLE]
```

```
*BRANCH*      3.1-RELEASE  (1999/02) -> 3.2 -> 3.3 -> 3.4 -> 3.5 -> 3.5.1
                |                    (1999/05) (1999/09) (1999/12) (2000/06) (2000/07)
                |  [4.0-STABLE]
*BRANCH*        4.0  (2000/03) ->4.1 -> 4.1.1 -> 4.2 -> 4.3 -> ... 将来の 4.x リリース
...
                |
                |             (2000/07)   (2000/09)   (2000/11)
                |
                \|/
                +
        [5.0-CURRENT として継続中]
```

-CURRENT ブランチは 5.0 とその先へ向けてゆっくりと進化を続けています。 従来あった 2.2-STABLE ブランチは 2.2.8 のリリースをもって終了しました。 3-STABLE がそれに代わり、2000 年 7 月に 3.5.1-RELEASE (最後の 3.X リリース) がリリースされました。 2000 年 3 月 (3.5 の公開前になりますが) には、3-STABLE ブランチはほぼ、4-STABLE ブランチによって置き換えられました。 4.3-RELEASE は 2001 年 4 月にリリースされました。 4-STABLE は現在 -STABLE ブランチで活発に開発が続けられていますが、3-STABLE へのバグの修正 (ほとんどがセキュリティ関連のもの) もまだ行なわれています。 3.X ブランチは 2000 年の夏には公式に開発が終了する予定です。 現在の "current branch" は 5.0-CURRENT であり、 最初の 5.0 系列のリリース予定はまだ決定していません。

FreeBSD のリリースはいつ作られるのですか?

FreeBSD コアチームは原則的に、 新しい機能やバグフィックスが充分集まり、リリースの安定性を損なうことが無いよう、
さまざまな変更が十分に安定しているという条件を満たしている場合にのみ、 新しいバージョンの FreeBSD をリリースします。 たとえこの用心深さが新しい機能が使えるようになることを待ち望んでいるユーザを欲求不満にさせるとしても、 多くのユーザはこのことを FreeBSD の最も良い所の一つだと考えています。

リリースの作成は、平均的に言っておよそ 4 ヶ月ごとに行なわれます。

もう少し刺激が欲しい (あるいは待ち遠しい) 方々向けには、毎日バイナリスナップショットが作成されています。 上記を参照してください。

FreeBSD は PC 用だけしかないの?

FreeBSD 3.x 以降は x86 アーキテクチャと同様、 DEC Alpha でも動作します。 また、SPARC、PowerPC、IA64 への移植という興味深い話もあります。

異なるアーキテクチャのマシンを 持っていて、ゆっくり待てないという場合には次の URL を参照してください。

NetBSD または OpenBSD。

FreeBSD の責任者はいったい誰?

プロジェクトの全体的な方向性や、 誰にソースツリーにコードの書き込み権限を与えるか、

などといった FreeBSD プロジェクトに関する重要な意思決定は、 9 名からなるコアチーム (core team) によってなされます。 ソースツリーを直接変更できる人はもっと多く、 200 名以上のソースツリー管理者 (committer) がいます。

しかし、メーリングリストで先行して議論される、
通常の変更ではないものの議論への参加には、一切制限はありません。

どこから FreeBSD を入手できますか?

FreeBSD のすべての主要なリリースは anonymous FTP 経由で FreeBSD FTP サイト から入手できます。

- 現在の 3.X-STABLE リリース、3.5.1-RELEASE は 3.5.1-RELEASE のディレクトリにあります。
- 現在の 4-STABLE リリース、4.3-RELEASE は 4.3-RELEASE のディレクトリにあります。
- 4.X Snapshot は、ほぼ一日に一回作成されています。
- 5.0 Snapshot リリースは -CURRENT ブランチ用に一日に一回作成されており、これらは純粋に最先端の開発者およびテスターのために提供されています。

また、FreeBSD は CD-ROM でも入手でき、次のところで注文できます。

BSDi
4041 Pike Lane, Suite F
Concord, CA
94520
USA

Orders: +1 800 786-9907
Questions: +1 925 674-0783
FAX: +1 925 674-0821
email: BSDi Orders address
WWW: BSDi Home pageOrders: +1 800 786-9907

オーストラリアでは、次のところに問い合わせてください。

Advanced Multimedia Distributors
Factory 1/1 Ovata Drive
Tullamarine, Melbourne
Victoria
Australia
Voice: +61 3 9338 6777

CDROM Support BBS
17 Irvine St
Peppermint Grove, WA
6011
Voice: +61 9 385-3793
Fax: +61 9 385-2360

イギリスの場合は次のところです。

The Public Domain & Shareware Library
Winscombe House, Beacon Rd
Crowborough
Sussex. TN6 1UL
Voice: +44 1892 663-298
Fax: +44 1892 667-473

FreeBSD のメーリングリストについて知りたいのですが?

完全な情報が FreeBSD ハンドブックのメーリングリストの節 にあります。

FreeBSD のニュースグループは何がありますか?

完全な情報が FreeBSD ハンドブックのニュースグループの節にあります。

FreeBSD の IRC (Internet Relay Chat) について何か情報はありますか?

あります。 以下のように、ほとんどの有名な IRC ネットワークには FreeBSD のチャットチャンネルがあります。

- EFNet の Channel #FreeBSD は FreeBSD 関係のフォーラムですが、
  そこで技術的サポートを期待してはいけません。
  そこにいる人たちはあなたをマニュアルページを読むとか、
  研究をするとかといった苦労から遠ざけようとします。
  まず第一に、これはチャットチャンネルであり、 そこにあるトピックスは恋人募集、スポーツ、
  核兵器といったようなものであり、 FreeBSD も同列に扱われています。 一応注意しましたからね!
  これは irc.chat.org のサーバー上にあります。
- EFNet の Channel *#FreeBSDhelp* は FreeBSD ユーザのヘルプ専用チャネルです。 参加者は *#FreeBSD* チャネルよりも親切に質問に答えてくれます。
- DALNET の Channel #FreeBSD はアメリカでは irc.dal.net、 ヨーロッパでは irc.eu.dal.net にあります。
- UNDERNET の Channel #FreeBSD はアメリカでは us.undernet.org、 ヨーロッパでは eu.undernet.org にあります。 ここはヘルプチャンネルです。
  ドキュメントを読める準備をしてから利用してください。
- HybNet の Channel #FreeBSD。 このチャンネルはヘルプチャンネルです。 サーバーのリストは HybNet のウェブサイト にあります。

それぞれのチャンネルは別個のもので、 互いに接続されていません。
チャットのスタイルも違っていますので、
自分のチャットのスタイルにあったものを見つけるために一つ一つ試すのもいいでしょう。
あらゆる種類の IRC トラフィックのため、失礼なことをいう若者たち (年輩の方は少数です)
のために機嫌を損ねたり、 手に負えなくなっても気にしてはいけません。

FreeBSD の本

FreeBSD documentation project メーリングリスト にコンタクトしてみてください (さらに参加すればもっとよいでしょう)。 このメーリングリストは FreeBSD 関連の文書に関する議論のためのものです。 FreeBSD に関する質問に対しては、 FreeBSD general questions メーリングリスト というメーリングリストがあります。

FreeBSD ハンドブックもあります。 これは現在作業中で、不完全だったり最新情報でないものが含まれていることに注意してください。

FreeBSD のガイド本の決定版は、 Greg Lehey 氏による "The Complete FreeBSD" です。 これは BSDi (以前の Walnut Creek CDROM) Books から出版されています。 現在は第三版になっていて、インストール、システム管理ガイド、プログラム設定のヘルプ、 マニュアルページまでの内容が 773 ページにわたって書かれています。 この本は (そして現在の FreeBSD リリースは) BSDi、 CheapBytes、または最寄りの書店で注文することができます。 ISBN コードは 1-57176-246-9 です (これ以外のコードの場合もあるかもしれません)。

また、FreeBSD は Berkeley 4.4BSD-Lite ベースなので、多くの 4.4BSD のマニュアルが FreeBSD にも応用できます。 O'Reilly and Associates が以下のマニュアルを出版しています。

これらの詳細な説明が WWW 経由で 4.4BSD books description から読むことができます。販売数が少ないためこれらのマニュアルは入手しにくいかもしれません。

4.4BSD のカーネル構成についてより徹底的に知りたいのなら、 [biblio-44kernel] なら間違いないでしょう。

システム管理についての良書が [biblio-nemeth3rd] です。



初版ではなく、紫色のカバーの第三版であるか確認してくだ さい。

この本は TCP/IP だけでなく DNS、NFS、SLIP/PPP、sendmail、INN/NNTP、印刷などの基礎を扱っています。 高価ですが、買う価値はあります。 第三版では、Solaris, HP/UX, FreeBSD および Linux を取り扱っています。

障害報告 (PR; Problem Report) データベースにアクセスする方法は?

ユーザからの変更要求がまとめられている Problem Report データベースは、 障害報告の web ベースのインタフェースを通して、 提出と問い合わせを行なうことができます。 また、send-pr(1) コマンドを使用して、 電子メール経由で障害報告や変更要求を提出することもできます。

プレインテキスト (ASCII) 版 や PostScript 版の FreeBSD 文書はないのでしょうか?

はい、もちろんあります。 数多くの異なるフォーマット、圧縮形式の文書が FreeBSD FTP サイトの /pub/FreeBSD/doc/ というディレクトリから入手可能です。

文書は、次のようなさまざまな観点から分類されています。

- faq や handbook といった文書名による分類。
- 文書の言語とエンコーディングによる分類。これは FreeBSD システムの /usr/shared/locale にある locale 名に基づいています。 現在利用可能な言語、エンコーディングは以下のとおりです。

名前	意味
en_US.ISO8859-1	英語 (米国)
de_DE.ISO_8859-1	ドイツ語
es_ES.ISO8859-1	スペイン語
fr_FR.ISO8859-1	フランス語
ja_JP.eucJP	日本語 (EUC エンコーディング)
ru_RU.KOI8-R	ロシア語 (KOI8-R エンコーディング)
zh_TW.Big5	中国語 (Big5 エンコーディング)



言語によっては準備されていない文書も存在します。

- 文書の形式による分類。 文書は数多くの異なる出力形式を用意し、可能な限り柔軟な対応ができるようにしています。 現在、利用可能な文書形式は以下のとおりです。

文書形式	意味
html-split	サイズの小さい、 リンクされた複数の HTML ファイル
html	文書全体を含んだ、単一の大きなファイル
pdb	iSilo で利用可能な Palm Pilot データベース形式
pdf	Adobe 社の PDF (Portable Document Format) 形式
ps	Postscript 形式
rtf	Microsoft 社のリッチテキスト形式
txt	プレインテキスト形式

- 圧縮と package 形式による分類。 現在利用されているのは次の 3 種類です。

  a. html-split 形式の場合、 ファイルはまず、tar(1) を使ってまとめられ、 まとめられた .tar ファイルは次に解説する方式で圧縮されます。

  b. その他の形式の場合、ファイルは book.format (たとえば book.pdb、 book.html など) という単一のファイルです。

  上にあげたファイルは 3 種類の方式のいずれかで圧縮されます。

方式	説明
zip	Zip 形式。 FreeBSD で圧縮を元に戻すには、まず archivers/unzip の port をインストールする必要があります。

方式	説明
gz	GNU Zip 形式。圧縮を元に戻すには、 FreeBSD に含まれる gunzip(1) を使います。
bz2	BZip2 形式。 他の形式に比べて普及していませんが、 一般的にファイルサイズが小さくなります。 圧縮を元に戻すには、 archivers/bzip2 port をインストールしてください。

Postscript 版のハンドブックが BZip2 形式で圧縮されている場合、ファイル名は handbook/ ディレクトリの中の book.xml.bz2 になります。

c. さまざまな形式に整形された文書は、以下に述べるように FreeBSD の package としても提供されています。

ダウンロードする文書と圧縮形式を選択したら、 文書を FreeBSD *package* としてダウンロードするかどうか決めなければなりません。

package としてダウンロードしてインストールする場合には、 文書を pkg_add(1) や pkg_delete(1) といった、普通の FreeBSD package 管理システムを用いた管理が可能であるという利点があります。

文書の package をダウンロードしてインストールすることに決めたら、 まずはダウンロードするファイル名を知る必要があります。 文書の package は、packages というディレクトリに置かれています。 そしてそれぞれの package ファイルは、 文書名.言語 .エンコーディング.形式.tgz というような名前になっています。

たとえば、FAQ の英語版で PDF 形式のものは、 faq.en_US.ISO8859-1.pdf.tgz というファイル名です。

ファイル名がわかったら、 次のようなコマンドで英語版の PDF 形式 FAQ の package をインストールすることができます。

```
# pkg_add ftp://ftp.FreeBSD.org/pub/FreeBSD/doc/packages/faq.en_US.ISO8859-1.pdf.tgz
```

インストールの終了後は pkg_info(1) を使い、 ファイルがどこにインストールされたかを調べることができます。

```
# pkg_info -f faq.en_US.ISO8859-1.pdf
Information for faq.en_US.ISO8859-1.pdf:

Packing list:
        Package name: faq.en_US.ISO8859-1.pdf
        CWD to /usr/shared/doc/en_US.ISO8859-1/books/faq
File: book.pdf
        CWD to .
File: +COMMENT (ignored)
File: +DESC (ignored)
```

ご覧になるとわかるとおり、book.pdf は /usr/shared/doc/en_US.ISO8859-1/books/faq にインストールされます。

package を利用しない場合は、自分で圧縮されたファイルをダウンロードして元に戻し、適切な場所にそれをコピーする必要があります。

たとえば、分割された HTML 版の FAQ で、gzip(1) で圧縮されているものは en_US.ISO8859-1/books/faq/book.html-split.tar.gz というファイルです。
これをダウンロードして圧縮を元に戻すには、次のようにする必要があるでしょう。

```
# fetch ftp://ftp.freebsd.org/pub/FreeBSD/doc/en_US.ISO8859-1/books/faq/book.html-split.tar.gz
# gzip -d book.html-split.tar.gz
# tar xvf book.html-split.tar
```

こうすると、複数の .html ファイルが作成されます。中心となっているのは index.html という名前のファイルで、目次や前書き、文書の他の部分へのリンクが含まれています。
これらのファイルは、必要に応じて他の場所にコピーしても構いません。

FreeBSD のウェブサイトのミラーサイトになりたいです!

承知しました! ウェブページをミラーするにはいくつかの手段があります。

- CVSup を使います。CVSup を使って CVSup サーバに接続することで、整形されたファイルを取ってくることができます。

  ウェブページを取得する場合は、/usr/shared/examples/cvsup/www-supfile にある supfile の例を参考にしてください。

- FTP を使ってミラーリングします。あなたの好きな FTP ミラーリングツールを使って、FTP サーバに置いてある web サイトのコピーをダウンロードすることができます。タウンロードは単純に ftp://ftp.FreeBSD.org/pub/FreeBSD/FreeBSD-CURRENT/www から始めてください。

この文書を他の言語に翻訳したいのですが?

報酬は支払えませんが、文書の翻訳を提出してくださる方には、フリーの CD、T シャツの手配や、ハンドブックにある貢献者一覧への登録を行ないたいと思います。翻訳作業をはじめる前に、FreeBSD documentation project メーリングリストへ連絡するようにお願いします。
翻訳作業を手伝うという人が現われるかも知れませんし。
既に翻訳チームがあって、あなたの参加を歓迎してくれるかも知れません。

その他の情報

以下のニュースグループには FreeBSD ユーザに直接関係のある議論が行われてます。

- comp.unix.bsd.freebsd.announce (moderated)
- comp.unix.bsd.freebsd.misc
- comp.unix.bsd.misc

Web 上のリソース:

- FreeBSD のホームページ

- ラップトップ PC を持っている方は、 迷うことなく日本の細川 達己氏の Mobile Computing のページを見ましょう。
- SMP (Symmetric MultiProcessing) に関する情報は、 SMP サポートページをご覧ください。
- FreeBSD のマルチメディアアプリケーションに関する情報は、マルチメディアのページをご覧ください。 特に Bt848 ビデオキャプチャチップに興味のある方は、リンクをたどってみてください。

FreeBSD ハンドブックには、 実に完成された参考図書の一覧があり、買うべき本をさがしている方は読む価値があります。

Chapter 1. インストール

1.1. FreeBSD を入手するには、どのファイルをダウンロードすれば良いのでしょうか?

FreeBSD 3.1-RELEASE 以前では、 インストールの際に必要なのは floppies/boot.flp と名前のついた一つのフロッピーディスクイメージだけでした。 しかし FreeBSD 3.1-RELEASE 以降、幅広い種類のハードウェアサポートが基本システムに追加され、
そのサポートが必要とする容量を補うため、 3.X と 4.X の系列では新たに、 floppies/kernel.flp および floppies/mfsroot.flp という、二つのフロッピーディスクイメージを使うようになりました。
これらのイメージをフロッピーディスクに書き込むには、 `fdimage` や dd(1) といったツールが必要となります。

(DOS ファイルシステムからのインストールなどで)
あなた自身が手動で配布ファイルをダウンロードする場合には、
以下の配布ファイルをダウンロードすることをおすすめします。

- bin/
- manpages/
- compat*/
- doc/
- src/ssys.*

この手順の完全な説明と、一般的なインストール時の問題については FreeBSD ハンドブックのインストールの節 を参照してください。

1.2. ブートフロッピーイメージが一枚のフロッピーディスクに納まらないみたい!

3.5 インチ (1.44MB) のフロッピーディスクには、 1474560 バイトのデータを格納できます。
ブートイメージはちょうど 1474560 バイトの大きさです。

ブートフロッピーディスクを準備する際のよくある間違いには、 以下のものがあります。

- FTP によってフロッピーイメージをダウンロードする際に、 バイナリ *(binary)* モードにしていなかった。

  FTP クライアントの中には、 転送モードのデフォルトをアスキー *(ascii)* モードにして、クライアント側システムの慣習にあうよう、 すべての行末の文字を変更するものがあります。
  この場合は常に、ブートイメージが壊れたものになります。
  ダウンロードしたブートイメージのサイズをチェックしてください。 サーバ上のものと正確に一致しなければ、 ダウンロードの処理を疑いましょう。

これを回避するには、サーバに接続してイメージのダウンロードを開始する前に FTP のコマンドプロンプトで `binary` とタイプします。

- ブートイメージを DOS の `copy` コマンド (または GUI の同等のツール) でフロッピーディスクへ転送した。

  `copy` のようなプログラムは、
  直接起動するように作成されたブートイメージをうまく処理できません。
  イメージにはフロッピーディスクの完全な中身がトラック単位で格納されており、
  フロッピーディスク上に通常のファイルとして
  格納されるように想定されているわけではありません。FreeBSD のインストールに記述されているように、低レベルのツール (たとえば `fdimage` や `rawrite`) を使用して "そのままの (raw)" の状態でフロッピーディスクに 転送する必要があります。

1.3. FreeBSD のインストールについての説明書はどこにありますか?

インストールの説明書はFreeBSD ハンドブックのインストールの章にあります。

1.4. FreeBSD を動作させるには何が必要ですか?

386 以上の PC、5MB 以上の RAM、そして最低 60MB のハードディスク容量が必要となります。ローエンドの MDA カードでも動作しますが、X11R6 を使うには VGA かそれ以上のビデオカードが必要となります。

ハードウェアコンパチビリティ もご覧ください。

1.5. 4MB しかメモリがないのですが、インストールできますか?

4MB のシステムにインストールできた最後の FreeBSD は FreeBSD 2.1.7 でした。2.2 を含むより新しいバージョンの FreeBSD は新規のインストールに最低 5MB は必要になります。

ただし、インストールプログラムが 4MB では動作しないだけで、3.0 を含む FreeBSD のすべてのバージョンは 4MB の RAM で動作可能です。インストールする時だけさらに 4MB 追加しておき、システムがセットアップされて動作するようになった後、また 4MB を取り出して元に戻すこともできます。あるいは 4MB より多くメモリを搭載したシステムにディスクを持っていき、
そのマシンでインストールした後にディスクを戻すこともできます。

また、FreeBSD 2.1.7 であっても、4MB ではインストールできない場合があります。正確には、640KB のベースメモリ + 3MB の拡張メモリでは、インストールはできません。もしマシンのマザーボードが 640KB から 1MB の領域で「失われた」メモリを再マップできる場合は、FreeBSD 2.1.7 をインストールできるかもしれません。

BIOS のセットアップ画面で、"remap" のオプションを探して有効 (enable) にしてみてください。また、ROM shadowing を無効 (disable) にする必要もあります。

簡単なやり方としては、インストールする時だけあと 4MB 追加しておく方法があります。
必要なオプションだけを選択してカスタムカーネルを構築し、 また 4MB
を取り出してもとに戻せばいいのです。

また、2.0.5 をインストールして、 それから 2.1.7 のインストーラの "upgrade" オプションでシステムを 2.1.7 へアップグレード するというやり方もあります。

インストールしたあとでカスタムカーネルの構築をした場合には、 4MB でも動作します。 2MB で起動に成功した人もいます (でもそのシステムは、 ほとんど使いものになりませんでした :-))。

1.6. 自分用のインストールフロッピーを作るには?

現在はカスタムインストールフロッピーディスク「だけ」を作る方法はありません。
カスタムインストールフロッピーディスクイメージを含む、 release
環境全体を新たに作る必要があります。

カスタムの release 環境をつくるには、 ここの指示にしたがってください。

1.7. 同じマシンで Windows 95/98 と共存できますか?

まず Windows 95/98 をインストールしてから、そのあとで FreeBSD
をインストールしてください。FreeBSD のブートマネージャが Win95 と FreeBSD
のブート管理をしてくれるようになります。 Windows 95/98
を後にインストールした場合はひどいことに、
問い合わせることもなくブートマネージャを上書きしてしまいます。
そうなってしまった場合は次の節をご覧ください。

1.8. Windows 95/98 がブートマネージャを潰しちゃった! どうやって戻すの?

ブートマネージャの再インストールの方法として、 FreeBSD
では以下に示す三通りの方法が用意されています。

- DOS を起動し、FreeBSD の配布物の中にある tools/ ディレクトリへ移動し、 bootinst.exe を探してください。 そして次のように実行します。

  ```
  ...\TOOLS> bootinst.exe boot.bin
  ```

  こうすることで、 ブートマネージャが再インストールされます。

- FreeBSD のブートフロッピーディスクから起動し、 「カスタム」インストールメニューを選択し、続いて「パーティション」を選択します。 ブートマネージャがインストールされていたドライブ (多分最初のもの) を選択し、 パーティションエディタにたどり着いたら、 (何も変更せず) そのまま (W)rite を指定します。 確認のメッセージが出ますので「はい(Y)」と答え、ブートマネージャ選択の画面で確実に "Boot Manager" を選択します。
  これでブートマネージャがディスクに再び書き込まれます。
  インストールメニューから抜けて再起動すると、 ハードディスクは元通りになります。

- FreeBSD 起動フロッピー (もしくは CD-ROM) から起動し、 "Fixit" メニューを選択します。 Fixit フロッピーか CD-ROM #2 ("live" ファイルシステムオプション) の好きな方を選択して fixit シェルに入ります。 そして、次のコマンドを実行してください。

```
Fixit# fdisk -B -b /boot/boot0 起動デバイス
```

起動デバイス の部分は、たとえば ad0 (一番目の IDE ディスク)、 ad4 (セカンダリ IDE コントローラの一番目の IDE ディスク)、 da0 (一番目の SCSI ディスク) などといった、実際の起動デバイスを表しています。

1.9. IBM Thinkpad の A、T、X シリーズのいずれかを持っています。 FreeBSD をインストールしたら起動しなくなってしまいました。 どうすればいいですか?

これらのマシンに使われている初期のリビジョンの IBM BIOS にはバグがあり、FreeBSD のパーティションをディスクサスペンド用の FAT 領域だと誤認します。 そのため、BIOS が FreeBSD のパーティションを 検出したところでシステムがハング (停止) してしまいます。

IBM によれば、以下のモデル/BIOS リリース番号には修正が含まれています。

モデル	**BIOS** リビジョン番号
T20	IYET49WW 以降
T21	KZET22WW 以降
A20p	IVET62WW 以降
A20m	IWET54WW 以降
A21p	KYET27WW 以降
A21m	KXET24WW 以降
A21e	KUET30WW

それより新しいリビジョンの BIOS にまたバグが入り込んだかもしれないという報告がありました。Jacques Vidrine は mobile@freebsd.org メーリングリストにあてた メッセージ で、これ以降の IBM の laptop で FreeBSD が正常に起動しない場合におそらくうまく行く、BIOS をアップグレードまたはダウングレードできる手順を説明しています。

もし問題のある BIOS を使っていてアップグレードが選べない場合、 FreeBSD をインストールしてから FreeBSD が使っているパーティション ID を変更し、 変更されたパーティション ID を正しく扱うことのできる 新しい起動ブロックをインストールすることで解決することができます。

それにはまず、 セルフテスト画面を通過する状態にまでマシンを回復させる必要があります。 そのためには、マシンがプライマリディスクから FreeBSD パーティションを見つけないようにして起動しなければなりません。

たとえば、一度ハードディスクを外してしまって、そのディスクを古い ThinkPad (ThinkPad 600 など) やデスクトップ PC に適切な変換ケーブルで接続します。 その後 FreeBSD のパーティションを削除し、ハードディスクを元の ThinkPad に戻します。 こうすることで ThinkPad は起動可能な状態に戻るはずです。

マシンがちゃんと動くようになったら、 以下の復旧手順に従って FreeBSD をインストールすることができます。

1. http://people.freebsd.org/~bmah/ThinkPad/ から boot1 と boot2 をダウンロードします。これらのファイルは、 あとで必要になった時、取り出せる場所に置いておきます。
2. ThinkPad に普通に FreeBSD をインストールします。 ただし、`Dangerously Dedicated` モードを使ってはいけません。 また、インストールが終わっても再起動してはいけません。
3. "緊急ホログラフィックシェル (Emergency Holographic Shell)" (`ALT` + `F4`) に切り替えるか、"fixit" シェルを起動します。
4. fdisk(8) を使って FreeBSD のパーティション ID を `165` から `166` に 変更します (これは OpenBSD で使われているものです)。
5. boot1 と boot2 のファイルをローカルファイルシステムに持って来ます。
6. disklabel(8) を使って boot1 と boot2 を FreeBSD のスライスに書き込みます。

   ```
   # disklabel -B -b boot1 -s boot2 ad0sn
   ```

   n は、 あなたが FreeBSD をインストールしたスライスの番号です。
7. 再起動します。起動プロンプトは `OpenBSD` と示しますが、実際には、それで FreeBSD が起動します。

この方法で FreeBSD と OpenBSD をデュアルブートする方法は、読者への練習問題としましょう。

1.10. 不良ブロックのあるディスクにインストールできますか?

FreeBSD 3.0 以前のシステムでは、 不良ブロックを自動的に再マッピングする `bad144` というユーティリティが含まれていましたが、 現在の IDE ドライブはドライブ自身がこの機能を備えているため、 `bad144` は FreeBSD ソースツリーから削除されました。 FreeBSD 3.0 かそれ以降をインストールしたいと思っているなら、比較的新しいディスクドライブを購入することを強くおすすめします。
新しいドライブを購入する気がなければ、FreeBSD 2.x を利用するべきです。

現在の IDE ドライブで不良ブロックによるエラーが発生した場合、まもなくドライブが故障する可能性があります (それはそのドライブ内蔵の再マッピング機能では不良ブロックが修正できなくなったということであり、 ディスクがひどく壊れていることを意味します)。 新しいハードディスクドライブに交換しましょう。

不良ブロックのある SCSI ドライブの場合は、 この回答を参照してください。

1.11. インストーラから起動したら変なことになりました!

インストーラから起動しようとしたときに、マシンが固まってし
まうとか自然と再起動してしまうといった現象であれば、 次の三つの項目を確認してください。

1. 新品の、フォーマットしたての、 エラーのないフロッピーディスクを使っていますか?
   (三年間もベッドの下に放置されていた雑誌の付録みたいなやつではなくて、
   買ってきたばかりの新品を使ってください)
2. フロッピーイメージをバイナリモードでダウンロードしましたか?
   (困った顔をしないでください。私たちの中で一番優秀な人でさえ、
   少なくとも一回はバイナリファイルを ASCII
   モードで思いがけずダウンロードしたことがあるのです!)
3. Windows95 あるいは Windows98 を使用しているなら、 ありのままの本物の DOS で `fdimage` か `rawrite` を実行しましたか? これらの OS はディスク作成プログラムのような、
   ハードウェアに直接書き込みを行なうプログラムに干渉する可能性があります。 GUI の中の DOS
   シェル内部で動作している場合でも、 この問題は発生します。

また、Netscape でブートイメージをダウンロードする場合も問題があることが報告されていますので、
できれば別の FTP クライアントを使うのがよいでしょう。

1.12. ATAPI CD-ROM から起動したのですが、インストールプログラムは CD-ROM が見つかりませんと言ってきます。 CD-ROM はどこに行ってしまったのでしょうか?

この問題は通常、CD-ROM ドライブの設定ミスによって発生します。 大部分の PC の CD-ROM ドライブは、 セカンダリ側の IDE コントローラのスレーブデバイスとして接続され、
マスタデバイスがない状態で出荷されています。 この接続方法は ATAPI 規格違反なので、 Windows は規格どおりに動いたり、動かなかったりしますが、 BIOS は起動時に規格違反を無視します。 そのため BIOS は起動時に CD-ROM を見つけられますが、 FreeBSD は CD-ROM を見つけられず、
インストールを完了できないのです。

CD-ROM が 接続されている IDE コントローラのマスタデバイスとなるように設定するか、
もしくはマスタ、 スレーブの両方にデバイスが接続されているようにシステムを再構成してください。

1.13. あれれ? テープからインストールできません!

FreeBSD 2.1.7R をテープからインストールする場合、 tar ブロックサイズを 10 (5120 バイト) にしたテープを作る必要があります。 デフォルト の tar ブロックサイズは 20 (10240 バイト) で、
このデフォルトサイズで作られたテープでは FreeBSD 2.1.7R をインストールすることはできません。
もしこうしたテープを使うと、 レコードサイズが大きすぎるというエラーが起きることになります。

1.14. PLIP 経由で二つ FreeBSD box を接続したいのですが

Laplink パラレルケーブルを用意して、 両方の PC のカーネルに lpt

ドライバが組み込まれていることを確認してください。

```
% dmesg | grep lp
lpt0 at 0x378-0x37f irq 7 on isa
lpt0: Interrupt-driven port
lp0: TCP/IP capable interface
```

パラレルインタフェースに Laplink パラレルケーブルを接続します。

root になって、両方で lp0 のネットワークインタフェースパラメータを設定します。 たとえば、ホスト max と moritz を接続したい場合、

```
              max <-----> moritz
IP Address      10.0.0.1        10.0.0.2
```

max 側で次のようにして、

```
# ifconfig lp0 10.0.0.1 10.0.0.2
```

moritz 側で同様に次のようにします。

```
# ifconfig lp0 10.0.0.2 10.0.0.1
```

以上です! lp(4) と lpt(4) のマニュアルページも参照してください。

また、 /etc/hosts にホストの追加もしましょう。

```
127.0.0.1               localhost.my.domain localhost
10.0.0.1                max.my.domain max
10.0.0.2                moritz.my.domain moritz
```

動作確認は次のようにします。

max 側:

```
% 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
```

```
moritz              max              UH          4   127592       lp0
```

```
% ping -c 4 moritz
PING moritz (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

--- moritz 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
```

1.15. ラップトップ PC に PLIP 経由でインストールできますか?

次のようにして、二つのコンピュータを Laplink パラレルケーブルで接続してください。

表 *1.* ネットワーク接続用のパラレルケーブルの結線

A-name	A 側	B 側	説明	ポート / ビット
.... 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	-

また、 Mobile Computing についてのページもご覧ください。

1.16. ハードディスクドライブには、どのジオメトリを使うべきでしょうか?



ここでディスクの「ジオメトリ」とは、ディスクのシリンダ、ヘッダ、トラック当りのセクタの数を意味しています - 便宜上、C/H/S とすることにします。これはディスクのどの領域で読み書きを 行なうかを PC の BIOS が決定する手段となります。

これについてはある理由のために、誤解されている点が多いようです。 まず最初に、FreeBSD はディスクブロックで動作しているため、 SCSI ドライブの "物理的" なジオメトリという言い方は、まったく見当違いのものです。事実、 セクタの密度はディスクによってまちまちであるため、物理的なジオメトリというものは存在しません。 製造者が "本当の" 物理的なジオメトリと公表しているものは通常、
彼らが検査して得た最小の使用不可容量の結果のジオメトリのことです。 IDE の場合、FreeBSD は C/H/S で動作しますが、 最近のドライブはすべて、これを内部で参照するブロックに変換しています。

問題はとなるのは論理的なジオメトリです。 これは BIOS がそのディスクのジオメトリについて調べた際に取得されるものであり、
その後のディスクへのアクセスに使用します。 FreeBSD は起動時に BIOS を使用するため、これを正しく取得することは非常に重要なことなのです。
実際に、ディスク上に複数のオペレーティングシステムがある場合は、
ジオメトリはどこからでも同じように解釈される必要があります。
そうしないと、起動時に深刻な問題が発生します。

SCSI ディスクでは、 使用するジオメトリはコントローラの拡張 BIOS トランスレーション (">1GB の DOS ディスクドライブのサポート" とも呼ばれます) が有効になっているかどうかによります。無効になっている場合、N シリンダ、64 ヘッド、 32 セクタ/トラックを使用しますが、 ここで `N` は MB 単位のディスク容量です。 たとえば、2GB ディスクは見かけ上 2048 シリンダ、64 ヘッド、 32 セクタ/トラックとなります。

それが「有効」になっており (MS-DOS ではこの方法で、ある制限を回避する場合もあります)、ディスク容量が 1GB を越える場合は、M シリンダ、 63 セクタ/トラック (64 「ではなく」)、 255 ヘッドを使用します。 `M` は MB 単位のディスク容量を 7.844238(!) で割った値となります。ということで、2GB ディスクの例では、 261 シリンダ、63 セクタ/トラック、255 ヘッドとなります。(訳注: 以上は Adaptec 社と NCR 社製の SCSI アダプタの場合です。 SCSI アダプタによって変換の数値が変わってくるのでマニュアルを 参照してください)。

これについてよく分からない場合や FreeBSD がインストール中に正しくジオメトリを取得できない場合、 これを回避するもっとも簡単な方法は、ディスクに小さな DOS パーティションを作ることです。
そうすると正しいジオメトリが取得されるはずです (そして、 残しておきたくないとか、ネットワークカードのプログラミング用に使いたい場合などには、 いつでもパーティションエディタで DOS パーティションを削除することができます)。

もう一つの方法として、FreeBSD と一緒に配布されているフリーで使えるユーティリティに pfdisk.exe (FreeBSD CD-ROM の tools ディレクトリや、他のさまざまな FTP サイトにあります)と呼ばれるものがあり、 ディスク上の他のオペレーティングシステムが使用しているジオメトリを調べるのに役立ちます。 このジオメトリ情報は、パーティションエディタに入力することができます。

1.17. ディスクの分割の仕方で何か制限はありますか?

はい。 BIOS がカーネルを起動できるようにルートパーティションが 1024 シリンダ以内にあることを確認する必要があります (これは FreeBSD ではなく PC の BIOS の制限です)。

SCSI ドライブでは、通常はルートパーティションが最初の 1024MB に収まっていることが前提となります (または拡張 BIOS トランスレーションが有効になっている場合は最初の 4096MB - 他の質問をご覧ください)。IDE でそれに相当する値は 504MB となります (訳注: E-IDE 対応の BIOS 搭載マシンの場合は IDE の 504MB という制限はありません)。

1.18. 大容量ディスクを持っていますが、ディスクマネージャは使えますか?

FreeBSD は Ontrack Disk Manager を認識し、これを考慮にいれます。 他のディスクマネージャはサポートしません。

ディスク全体を FreeBSD で使いたい場合、 ディスクマネージャは必要ありません。 BIOS が扱える容量 (通常 504MB) いっぱいでディスクの設定を行なうと、 FreeBSD は実際の容量を算出するはずです。 MFM コントローラ付きの古いディスクを使っている場合は、 FreeBSD に使用するシリンダ数を詳細に指定する必要があります。

FreeBSD と他のオペレーティングシステムが入っているディスクを使用したい場合は、 ディスクマネージャなしでもできるでしょう。 FreeBSD の起動パーティションと他のオペレーティングシステム用のスライスが、 最初の 1024 シリンダ内に収まっている事を確認するだけです。 気になる方は、起動パーティションを 20 メガバイトぐらいにして大きめにするとよいでしょう。

1.19. FreeBSD の起動時に Missing Operating System と表示されます

これは FreeBSD や DOS、 そのほかの OS がディスク領域ジオメトリのとらえ方で衝突しあっていることから起こる典型的な例です。 こうなったら FreeBSD をインストールし直す以外にはありませんが、 他のところで説明した手順にしたがってやれば、 ほぼ間違いなくうまくいくはずです。

1.20. ブートマネージャの F? プロンプトが表示されません。

これはすでに前に質問されている問題のもう一つの症状です。 BIOS のジオメトリと FreeBSD のジオメトリ設定が一致していないのです! コントローラや BIOS がシリンダの変換 (>1GB ドライブのサポートとも呼ばれます) をサポートしていたら、 その設定を無効化して FreeBSD をインストールし直してみてください。

1.21. ソースを全部インストールする必要はありますか?

一般的には「いいえ」です。 しかし最低でも、base ソースキット (これにはこの FAQ

で述べられているファイルのいくつかが含まれています) と、 sys (kernel) ソースキット (これにはカーネルのソースが含まれています) をインストールする事を強くおすすめします。
通常、何かの実行にソースが必要になる事はありません。
しかし、カーネルをコンフィグレーションするためのプログラム config(8) を実行する時は例外です。
カーネルのソースをインストールしなくてもよい例として、
どこか別の場所からカーネルのソースを読み込み専用で NFS マウントすることができます。また、そこから新しいバイナリを作成できるようにもなっています (カーネルソースの制限があるので、直接 /usr/src をマウントする事はおすすめできません。 それよりもどこか別のディレクトリにマウントして、ソースツリーの複製ができるように適切にシンボリックリンクを張ってください)。

ソースをネットワーク上に持ち、 そこからシステムをビルドするようにしておけば、 FreeBSD の将来のリリースへのアップグレードがずっと簡単になります。

実際にソースのサブセットを選択するには、
システムインストールツールの「配布ファイル」メニューにある、「カスタム」メニューを使用します。

1.22. カーネルは必ず作り直さなくちゃならないんですか?

カーネルを新しく作り直すのは元々、 FreeBSD のインストール時に必須の作業でした。
でも最近のリリースでは、 とてもユーザフレンドリなカーネル設定ツールの恩恵を受けています。
FreeBSD の起動プロンプト (boot:) で -c とタイプすればビジュアルな設定画面になり、ほとんどの一般的な ISA カードについてのカーネルの設定をすることができるのです。

今でも、 必要なデバイスドライバだけを組み込んだカーネルを作ることはよい事とされています。
ほんのちょっとだけメモリを節約できますからね。 でもほとんどのシステムでは、もはやどうしてもやらなくちゃならないことではないのです。

1.23. DES と MD5、どちらのパスワードを使うべきなのでしょうか? また、ユーザがどちらを使うことになるか指定する方法はありますか?

FreeBSD の標準のパスワードフォーマットは *MD5* を使ったものです。 これは *DES* アルゴリズムに基づいた手法を用いる UNIX の伝統的なパスワードフォーマットより安全 (secure) だと信じられているものです。 DES パスワードは あなたが FreeBSD のパスワードファイルを、安全性に劣るパスワードフォーマットを利用している古い OS と共有しなければならなくなったときのために 利用可能になっています (これは利用するためには、sysinstall から "crypto" 配布物のインストール 選ぶか、ソースから build しているなら、 crypto のソースがインストールされている必要があります)。
新しいパスワードにどちらのパスワードフォーマットを使うかは /etc/login.conf の中の "passwd_format" という login ケーパビリティで制御されます。このケーパビリティは "des" (利用できるなら) か "md5" のどちらかの値を取ります。 login ケーパビリティの詳細については login.conf(5) を 参照してください。

1.24. ブートフロッピーで起動すると、 **Probing Devices...** の画面でハングアップします。

IDE Zip か Jaz ドライブが接続されていたら、 それを取り外してもう一度試してみましょう。
ブートフロッピーはこの種のドライブを誤認してしまうのです。
システムがインストールされた後は、そのドライブを再度接続することができます。
うまくいけばこの問題は将来のリリースで解決されるでしょう。

1.25. インストール終了後にシステムを再起動すると、 **panic: cant mount root** のエラーとなります。

このエラーはディスクデバイスについて、
起動ブロックとカーネルの認識が混乱しているために起こります。 このエラーは通常、 2 台の IDE ディスクがそれぞれ別の IDE コントローラのマスターに一つずつ接続されているシステムにおいて、 FreeBSD がセカンダリ IDE コントローラに接続されたディスクにインストールされている場合に発生します。 起動ブロックは FreeBSD が wd1 (2 台目の BIOS ディスク) にインストール されていると認識するのに対し、 カーネルはセカンダリ IDE の 1 台目のハードディスクである wd2 にインストールされていると認識するのです。 デバイス検出後で、 カーネルは起動ブロックが起動ディスクだと認識したディスクである wd1 をマウントしようとします。 しかし、実際には起動ディスクは wd2 なので失敗してしまうのです。

この問題を解決するには、以下のどれか一つを行ってください。

1. FreeBSD 3.3 以降を利用している場合には、 システムを再起動して、`Booting kernel in 10 seconds; hit [Enter] to interrupt` が表示されている間に `Enter` キーを押します。 すると、ブートローダに移行します。

   そうしたら、`set root_disk_unit="disk_number"` と入力します。 FreeBSD が最初の IDE コントローラのマスターに接続されたドライブにインストールされていれば、 *disk_number* は `0` です。 また、 最初の IDE コントローラのスレーブなら `1`、 二番目の IDE コントローラのマスターなら 2、 二番目の IDE コントローラのスレーブなら 3 になります。

   その後、`boot` と入力します。 システムはきちんと再起動するはずです。

   この変更を恒久的なものにする (つまり、 再起動や電源を入れる度にこの操作をする必要がないようにする) には、 /boot/loader.conf.local に `root_disk_unit="disk_number"` という行を追加してください。

2. FreeBSD 3.2 以前を利用している場合は、 Boot: プロンプトで `1:wd(2,a)kernel` と入力してエンターキーを押します。 システムが起動したら、 `echo "1:wd(2,a)kernel" > /boot.config` というコマンドを実行してこれをデフォルトのブート文字列とします。

3. FreeBSD のディスクをプライマリ IDE コントローラに接続して、 ハードディスクが連続したドライブ番号で認識されるようにします。

4. カーネルのコンフィグレーションファイルで wd の行を以下のように変更し、 カーネルの再構築を行って、 新しいカーネルをインストールします。

```
controller      wdc0    at isa? port "IO_WD1" bio irq 14 vector wdintr
disk            wd0     at wdc0 drive 0
# disk            wd1     at wdc0 drive 1 # この行をコメントアウト

controller      wdc1    at isa? port "IO_WD2" bio irq 15 vector wdintr
disk            wd1     at wdc1 drive 0 # wd2 から wd1 へ変更
disk            wd2     at wdc1 drive 1 # wd3 から wd2 へ変更
```

ディスクの接続を変更して元の設定に戻したい場合は、ディスクを
お望みの設定の通りの接続に戻してから再起動します。 システムは正常に起動するはずです。

1.26. メモリの大きさの制限は?

認識できるメモリの上限は、4GB です。 この構成は試験済みで、 詳細は wcarchive's configuration をご覧ください。 このようにたくさんのメモリをマシンに導入しようという場合には、注意が必要です。ECC 機能をサポートし、なおかつ 容量性負荷 (訳注: 多くのメモリ素子は容量性負荷として働きますが、
メモリバス上に容量性負荷が増えると信号の伝達が遅れ、誤動作の原因となります) を低減させるため、18 チップ構成のメモリモジュールより 9 チップ構成のメモリモジュールを選択することが、おそらく望ましいでしょう。

1.27. ffs ファイルシステムの大きさの制限は?

ffs ファイルシステムの場合、 論理的な最大の上限は 8 TB (2G ブロック)、デフォルトのブロックサイズを 8K とすると 16 TBとなります。 実際問題として、1 TB のソフトウェアの限界がありますが、 修正すれば 4 TB のファイルシステムが可能です (実際に存在します)。

一つの ffs のファイルの最大のサイズは、ブロックサイズが 4K の場合で 約 1G ブロック (4 TB)です。

表 2. 最大ファイルサイズ

fs ブロックサイズ	**2.2.7-stable**	**3.0-current**	動作確認済みのサイズ	動作するはずのサイズ
4K	4T-1	4T-1	4T-1	>4T
8K	>32G	8T-1	>32G	32T-1
16K	>128G	16T-1	>128G	32T-1
32K	>512G	32T-1	>512G	64T-1
64K	>2048G	64T-1	>2048G	128T-1

fs ブロックサイズが 4K の場合は三重間接ブロックが使用され、いずれの場合でも三重間接ブロックを使用して表現できる最大の fs ブロック番号 (およそ 1K^3 + 1K^2 + 1K) に制限されるはずなのですが、 実際は fs ブロック番号の (間違った) 上限 1G-1 で制限されます。 fs ブロック番号の制限は 2G-1 となるはずです。2G-1 付近に fs ブロック番号のバグが多少ありますが、fs ブロックサイズが 4K の場合は、ここまでのブロック番号には到達しません。

ブロックサイズが 8K 以上の場合、いずれの場合も fs ブロック番号の上限 2G-1 で制限されるはずですが、 実際は fs ブロック番号の上限 1G-1 で制限されます。 例外的に -STABLE では三重間接ブロックまでは到達しないため、 制限は二重間接ブロックで表現できる最大の fs ブロック番号 (およそ (blocksize/4)^2 + (blocksize/4)) となります。 -CURRENT ではこの制限を超えると問題を引き起こすかもしれません。 正しい制限値である 2G-1 ブロックを使用すると明らかに問題が出ます。

1.28. フロッピーに 1 TB のファイルを格納するには?

わたしのところでは、 フロッピーにいくつかの実際のファイルを保存しています :-)。
最大のファイルサイズは最大のディスクサイズとはあまり関係はありません。 最大のディスクサイズは 1 TB です。 ファイルサイズがディスクサイズより大きくなりうるというのは仕様です。

以下の例は、32K のディスク容量 (3 つの間接ブロックと 1 つのデータブロック) を使って、小さなルートパーティションに 8T-1 の大きさのファイルを作成します。 ここでの dd コマンドは大きなファイルが扱えるものが必要です。

```
% cat foo
df .
dd if=/dev/zero of=z bs=1 seek=`echo 2^43 - 2 | bc` count=1
ls -l z
du z
df .
% sh foo
Filesystem  1024-blocks     Used    Avail Capacity  Mounted on
/dev/da0         64479    27702    31619    47%    /
1+0 records in
1+0 records out
1 bytes transferred in 0.000187 secs (5346 bytes/sec)
-rw-r--r--  1 bde  bin  8796093022207 Sep  7 16:04 z
32      z
Filesystem  1024-blocks     Used    Avail Capacity  Mounted on
/dev/da0         64479    27734    31587    47%    /
```

1.29. 新しいカーネルをコンパイルしたら、起動時に **archsw.readin.failed** というエラーメッセージが表示されるようになってしまいました。

ローダがスタートする前の | が表示されているときに何かキーを押すことで、起動のセカンドステージから直接、起動するカーネルを指定して起動することができます。
特に、カーネルのソースを更新し、*make world* しないで新しいカーネルだけインストールした場合にこの症状が現われます。
こういう操作は動作が保証されません。きちんと make world してください。

1.30. 3.X から 4.X にアップグレードするにはどうしたら良いのですか?

アップグレードには、 バイナリスナップショットを使うことを強くおすすめします。 4-STABLE スナップショットは releng4.FreeBSD.org から入手可能です。

ソースを使ってアップグレードする場合は、詳細について FreeBSD ハンドブックを参照するようにしてください。



ソースを使ったアップグレードは、
慣れていないユーザにはまったくおすすめできません。 3.X から 4.X への場合は特にそうです。 ソースを使ったアップグレードを試す前に、手順を注意深く読むように心がけてください。

1.31. セキュリティプロファイル (security profiles) とは何ですか?

"セキュリティプロファイル"とは、特定の
プログラムやその他の設定を有効にしたり無効にすることで、求める
比率で安全と便利さを実現しようとする構成の選択肢の集まりの
ことです。セキュリティプロファイルが厳しいほど、デフォルトで
有効になるプログラムが減ります。これは、動かさなければならない
もの以外は、何も動かしてはいけないというセキュリティの 基本的原則の一つです。

セキュリティプロファイルは、単にデフォルトの設定である ということに気をつけてください。FreeBSD をインストールした あとに /etc/rc.conf に適切な行を編集したり追加すれば、どのプログラムでも有効にしたり無効にしたりできます。 後者について詳しいことは rc.conf(5) のマニュアルを ご覧ください。

以下に、各セキュリティプロファイルが何を行うかを説明した
表を掲載します。列はセキュリティプロファイルの選択肢で、行は
有効または無効になるプログラムや機能です。

表 3. 指定できるセキュリティプロファイル

	Extreme	High	Moderate	Low
inetd(8)	NO	NO	YES	YES
sendmail(8)	NO	YES	YES	YES
sshd(8)	NO	YES	YES	YES

	Extreme	High	Moderate	Low
portmap(8)	NO	NO	おそらく (インストール時に、すでにマシンを NFS クライアントまたはサーバとして設定していると、ポートマッパが有効になります。)	YES
NFS server	NO	NO	YES	YES
securelevel(8)	YES (2) (securelevel を設定するセキュリティプロファイル (Extreme または High) を選択する場合、その影響を承知していなければなりません。init(8) のマニュアルを読み、セキュリティレベルの意味について特に注意を払ってください。そうしないと、後で深刻な問題が起きるかもしれません。)	YES (1)	NO	NO



セキュリティプロファイルは魔法の薬ではありません。 High に設定したら、適当なメーリングリストを読んだり、良質なパスワードやパスフレーズを用いたり、セキュリティについてのよい習慣を守ったりしなくていいわけではありません。求めるセキュリティと便利さの比率を手軽に設定してくれるだけです。



セキュリティプロファイルの機構は、FreeBSD を最初にインストールする時に使うことを想定しています。すでに FreeBSD がインストールされているなら、単に求める機能を有効にしたり無効にしたりする方が、おそらく効率がよいでしょう。もし、本当にセキュリティプロファイルを使いたいのであれば、sysinstall(8) を再実行すれば 設定できます。

Chapter 2. ハードウェアコンパチビリティ

2.1. FreeBSD は、どんなハードディスクドライブをサポートしているのですか?

FreeBSD は、EIDE と SCSI ハードディスクドライブをサポートしています (互換コントローラも含みます。次の節参照)。また独自の "Western Digital" インタフェースを使用しているすべてのドライブ (MFM、RLL、ESDI、もちろん IDE) もサポートしています。独自仕様のインタフェースを使用する ESDI コントローラでは動作しないものがあり、 WD1002/3/6/7 とその互換インタフェースと衝突します。

2.2. どの SCSI コントローラをサポートしているのですか?

FreeBSD ハンドブックに記されている完全なリストを参照してください。

2.3. どんな CD-ROM ドライブをサポートしているのですか?

サポートされている SCSI コントローラに接続できる SCSI ドライブは、すべてサポートされています。

また、以下の専用 CD-ROM インタフェースもサポートしています。

- ミツミ LU002 (8bit)、LU005 (16bit) および FX001D (16bit 2倍速)。
- ソニー CDU 31/33A
- Sound Blaster 非 SCSI タイプの CD-ROM
- 松下/Panasonic CD-ROM
- ATAPI 互換の IDE CD-ROM

SCSI でないカードはすべて、SCSI ドライブよりも極めて動作速度が 遅いことが知られており、ATAPI CD-ROM には動作しないものもあるようです。

BSDi の FreeBSD 2.2 CD-ROM からは CD からの直接起動が サポートされています。

2.4. FreeBSD は、どの CD-RW ドライブに対応していますか?

FreeBSD は ATAPI 互換の IDE CD-R または CD-RW ドライブで あれば対応しています。FreeBSD バージョン 4.0 以降については、 burncd(8) のマニュアルをご覧ください。それ以前の バージョンの FreeBSD では、 /usr/shared/examples/atapi にある例を ご覧ください。

また、FreeBSD は SCSI の CD-R または CD-RW ドライブにも 対応しています。ports または packages から `cdrecord` コマンドをインストールして、カーネルに pass デバイスが組み込まれていることを確認してください。

2.5. ZIP ドライブをサポートしていますか?

もちろん、 FreeBSD は SCSI ZIP ドライブ (外付け) をサポートしています。 ZIP ドライブは SCSI ID を 5 か 6 に設定した状態でなら使用できますが、 もし SCSI ホストアダプタの BIOS がサポートしてさえいれば ZIP ドライブから起動させることもできます。 どのホストアダプタが SCSI ID を 0 や 1 以外に設定したデバイスから 起動できるのかはわかりません。そうしたい場合は、アダプタのドキュメントを参照しなければなりません。

ATAPI (IDE) ZIP ドライブは、FreeBSD 2.2.6 以降のバージョンでサポートされています。

バージョン 3.0 以降の FreeBSD では、 パラレルポート接続の ZIP ドライブをサポートしています。最近のバージョンの FreeBSD をお使いでしたら、 カーネルコンフィグレーションファイルに scbus0、da0、 ppbus0、 vp0 の各ドライバが記述されていることを確認してください。 (GENERIC カーネルには vp0 を除くすべてのドライバが含まれています)。 これらすべてのドライバがあれば、パラレルポートのドライブは /dev/da0s4 となります。 ディスクは `mount /dev/da0s4 /mnt` とするか `mount_msdos /dev/da0s4 /mnt` (DOS ディスクの場合) とすることでマウントできます。

それからリムーバブルドライブに関する注意および、 「フォーマット」に関する注意についても確認しておいてください。

2.6. では、JAZ や EZ、それからその他のリムーバブルドライブはサポートしていますか?

FreeBSD では、IDE バージョンの EZ ドライブを除くすべての SCSI デバイスは、 SCSI のディスクと同等に扱われます。 また IDE EZ は IDE ドライブと同等となります。

システム稼働中のメディア交換について FreeBSD がどれほどうまく動くか定かではありません。もちろんメディアを入れ替える前にそのドライブのマウントを解除しなければいけないでしょうし、FreeBSD がそれらを認識するには、起動時に外部ユニットにも電源が投入されていることを確認しなければいけないでしょう。

「フォーマット」に関する注意も参照のこと。

2.7. どのマルチポートシリアルカードをサポートしていますか?

一覧は その他のデバイスの節にあります。

無名のカードにもうまく動くものがあり、 特に AST 互換といわれているものに多く見られます。

カード設定の詳細な情報は、sio(4) のマニュアルページを参照してください。

2.8. USB キーボードを持っているのですが、FreeBSD で使えますか?

USB デバイスは FreeBSD 3.1 からサポートされましたが、 実装は FreeBSD 3.2

であってもまだ完全ではないため、 必ずしも安定して動作するとは限りません。 もし、それでも USB キーボードを使ってみたいという人は、 以下の手順を試してみてください。

1. FreeBSD 3.2 か、それ以降を使います。
2. 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、 カーネルを再構築します。

```
device  uhci
device  ohci
device  usb
device  ukbd
options KBD_INSTALL_CDEV
```

   FreeBSD 4.0 より前のバージョンでは、 代わりに次のようにします。

```
controller      uhci0
controller      ohci0
controller      usb0
controller      ukbd0
options         KBD_INSTALL_CDEV
```

3. /dev ディレクトリに移動し、 次のようにしてデバイスノードを作成します。

```
# cd /dev
# ./MAKEDEV kbd0 kbd1
```

4. /etc/rc.conf を編集し、 以下の行を追加します。

```
usbd_enable="YES"
usbd_flags=""
```

システムを再起動させた後、 AT、USB 両方のキーボードが接続されていれば、 AT キーボードは /dev/kbd0 に、 USB キーボードは /dev/kbd1になります。 一方、USB キーボードだけが接続されているなら、 /dev/ukbd0 となります。

USB キーボードをコンソールで利用するには、
それをコンソールドライバに対して明示的に指定する必要があります。
システムの初期化の際に、次に示すようなコマンドを実行してください。

```
# kbdcontrol -k /dev/kbd1 < /dev/ttyv0 > /dev/null
```

ただし、USB キーボードしか接続されていない場合、それは /dev/kbd0 としてアクセスされますので、
コマンドは次のようにしなければなりません。ご注意ください。

```
# kbdcontrol -k /dev/kbd0 < /dev/ttyv0 > /dev/null
```

上のコマンドは、/etc/rc.i386 に追加すると良いでしょう。

この設定を一度行なっていれば、 X 環境でも特に他の設定なしに USB キーボードが利用できます。

USB キーボードの活線挿抜（ホットプラグ機能）は、 まだおそらくきちんと動作しないと思われます。
トラブルを避けるためにも、キーボードはシステムを起動させる前に接続しておき、
シャットダウンするまではずさないようにした方が良いでしょう。

詳細については、ukbd:(4) のマニュアルページを参照してください。

2.9. 珍しいバスマウスを持っているのですが、どのように設定すればいいのですか?

FreeBSD は Microsoft、Logitech、 ATI 等のメーカーから出ているバスマウスと InPort バスマウスをサポートしています。FreeBSD 2.X の場合、 バスマウスのデバイスドライバは GENERIC カーネルに標準で含まれますが、 FreeBSD 3.X 以降では標準で含まれていません。もしバスマウスのデバイス
ドライバを含むカーネルを自分で構築する場合には、
カーネルコンフィグレーションファイルに以下の行が含まれていることを確認してください。

それは FreeBSD 3.0 を含む、それ以前のリリースの場合は次のとおり、

```
device mse0 at isa? port 0x23c tty irq5 vector mseintr
```

FreeBSD 3.X では、次のとおりです。

```
device mse0 at isa? port 0x23c tty irq5
```

そして FreeBSD 4.X とそれ以降では、次のようになります。

```
device mse0 at isa? port 0x23c irq5
```

通常バスマウスには専用のインタフェースカードが附属しています。
インタフェースカードによってはポートアドレスや割り込み番号を上記の
設定以外に変更できるかもしれません。詳しくはバスマウスのマニュアルと mse(4)
のマニュアルページを参照してください。

2.10. PS/2 マウス (「マウスポートマウス」、「キーボードマウス」) を 使うにはどのように設定すればいいのですか?

あなたが 2.2.5 以降のバージョン FreeBSD を使っているのなら、 必要なドライバ psm はカーネルに含まれていて有効になっています。 カーネルは起動時に PS/2 マウスを検出するでしょう。

あなたの使っている FreeBSD が比較的新しいけれど前のバージョン (2.1.x 以降) のものなら、インストールの時に、単にカーネルのコンフィグレーションのメニュー上で PS/2 マウスを有効化するだけです、あるいは後で `boot:` プロンプト上で `-c` を指定することでもメニューは現れます。 デフォルトでは無効に設定されていますので、明示的に有効化してあげないといけません。

あなたの使っている FreeBSD が比較的古いものなら、カーネルコンフィグレーションファイルに以下の行を加えて カーネルを再コンパイルする必要があります。

それは FreeBSD 3.0 を含む、それ以前のリリースでは次のとおり、

```
device psm0 at isa? port "IO_KBD" conflicts tty irq 12 vector psmintr
```

FreeBSD 3.1 を含む、それ以降のリリースでは次のとおり、

```
device psm0 at isa? tty irq 12
```

FreeBSD 4.0 とそれ以降のリリースでは次のとおりです。

```
device psm0 at atkbdc? irq 12
```

カーネルの再構築についてよく知らないのであれば、
カーネルのコンフィグレーションを参照してください。

起動時にカーネルが psm0 を検出したら、 psm0 のエントリが /dev の中にあることを確認してください。それには、以下のようにします。

```
# cd /dev; sh MAKEDEV psm0
```

これは `root` でログインしているときに行なってください。

2.11. X Window System 以外の環境でマウスを使うことは可能ですか?

もしデフォルトのコンソールドライバである syscons を使っているのであれば、テキストコンソール上でマウスを使って、 テキストのカットアンドペーストができます。

マウスデーモンである moused を起動し、 仮想コンソールでマウスポインタを有効にしてください。

```
# moused -p /dev/xxxx -t yyyy
# vidcontrol -m on
```

ここで *xxxx* はマウスのデバイス名、 *yyyy* はマウスのプロトコルタイプです。
サポートされているプロトコルタイプについては moused(8) のマニュアルページを参照してください。

システムを起動する時に自動的に moused を起動したい場合には、次のようにします。 FreeBSD 2.2.1 では以下の変数を /etc/sysconfig で設定してください。

```
mousedtype="yyyy"
mousedport="xxxx"
mousedflags=""
```

FreeBSD 2.2.2 以降のバージョンでは /etc/rc.conf で以下のように設定します。

```
moused_type="yyyy"
moused_port="xxxx"
moused_flags=""
```

FreeBSD 3.1 とそれ以降で PS/2 マウスを利用する場合は、 `moused_enable="YES"` を /etc/rc.conf に書き加えるだけです。

また、起動時にすべての仮想端末で、 標準のコンソールに加えマウスデーモンも使えるようにしたい、という場合には、以下の行を /etc/rc.conf に加えます。

```
allscreens_flags="-m on"
```

FreeBSD 2.2.6 以降の場合で 比較的新しいシリアルマウスを使っているならば、マウスデーモンはマウスのプロトコルタイプを自動判別できます。
自動判別を試みるには、プロトコルタイプとして `auto` を指定します。

マウスデーモンを実行中は、マウスデーモンと他のプログラム (たとえば X Window System) の間でマウスへのアクセスを調整しなければなりません。 この問題については X とマウスをご覧ください。

2.12. マウスを使って、テキストコンソールでカットアンドペーストするにはどうしたらよいのですか?

マウスデーモンを起動 (前の質問に対する答えを参照してください) したあと、 ボタン 1 (左ボタン) を押しながらマウスを動かして範囲を指定します。 ボタン 2 (中ボタン) またはボタン 3 (右ボタン) をクリックするとテキスト カーソルの位置に選択した範囲のテキストがペーストされます。

FreeBSD 2.2.6 以降では、ボタン 2 をクリックするとペーストされ、ボタン 3 をクリックした場合に既存の選択範囲が現在のマウスポインタの位置まで「延長または短縮」されます。もしマウスに中ボタンがないなら、 moused のオプションを使って中ボタンのエミュレーションをするか、他のボタンを中ボタンとして使う事ができます。 詳しくは mused(8) のマニュアルページを参照してください。

2.13. USB マウスを持っているのですが、FreeBSD で使えますか?

USB デバイスは FreeBSD 3.1 からサポートされましたが、 実装は FreeBSD 3.2 であってもまだ完全ではないため、 必ずしも安定して動作するとは限りません。 もし、それでも USB マウスを使ってみたいという人は、 以下の手順を試してみてください。

1. FreeBSD 3.2 か、それ以降を使います。
2. 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、 カーネルを再構築します。

```
device  uhci
device  ohci
device  usb
device  ums
```

FreeBSD 4.0 より前のバージョンでは、 代わりに次のようにします。

```
controller      uhci0
controller      ohci0
controller      usb0
device          ums0
```

3. /dev ディレクトリに移動し、 次のようにしてデバイスノードを作成します。

```
# cd /dev
# ./MAKEDEV ums0
```

4. /etc/rc.conf を編集し、 以下の行を追加します。

```
moused_enable="YES"
moused_type="auto"
moused_port="/dev/ums0"
moused_flags=""
usbd_enable="YES"
usbd_flags=""
```

moused の設定の詳細については、 前項も参照してください。

5. X のセッションで USB マウスを使うには、 XF86Config を編集する必要があります。 XFree86 3.3.2、もしくはそれ以降を利用している場合は、 *Pointer* セクションが次のようになっていることを確認してください。

```
Device          "/dev/sysmouse"
Protocol        "Auto"
```

それより前のバージョンの XFree86 を利用している場合は、 *Pointer* セクションが次のようになっていることを確認してください。

```
Device          "/dev/sysmouse"
Protocol        "SysMouse"
```

X 環境でのマウスの利用については、 他の項も参照してください。

USB マウスの活線挿抜 (ホットプラグ機能) は、 まだおそらくきちんと動作しないと思われます。
トラブルを避けるためにも、マウスはシステムを起動させる前に接続しておき、
シャットダウンするまではずさないようにした方が良いでしょう。

2.14. わたしのマウスにはホイール機能や便利なボタンがついているのですが、 これは FreeBSD でも使えるのですか?

答えは残念ながら「場合によります」です。
こうしたマウスの付加的な機能は大抵の場合、特殊なドライバを必要とします。
マウスのデバイスドライバやユーザのプログラムが
そのマウスに対する固有のサポートをしていない場合には、 標準的な 2 ボタン/3 ボタンマウスのように振舞います。

X ウィンドウシステムの環境でのホイールの使い方については、 X とホイールの項をご覧ください。

2.15. わたしのマウスはきちんと動いてくれないようです。マウスカーソルが画面中をとびまわります。このマウスにはホイールがついていて、 接続は PS/2 ポートです。

FreeBSD 3.2 およびそれ以前の PS/2 マウスドライバ psm には、 Logitech モデル M-S48 とその OEM のホイールマウスで不具合が発生します。 以下のパッチを /sys/i386/isa/psm.c に適用して、カーネルを再構築してください。

```
Index: psm.c
===================================================================
```

```
RCS file: /src/CVS/src/sys/i386/isa/Attic/psm.c,v
retrieving revision 1.60.2.1
retrieving revision 1.60.2.2
diff -u -r1.60.2.1 -r1.60.2.2
--- psm.c   1999/06/03 12:41:13 1.60.2.1
+++ psm.c   1999/07/12 13:40:52 1.60.2.2
@@ -959,14 +959,28 @@
    sc->mode.packetsize = vendortype[i].packetsize;

    /* set mouse parameters */
+#if 0
+    /*
+     * A version of Logitech FirstMouse+ won't report wheel movement,
+     * if SET_DEFAULTS is sent...  Don't use this command.
+     * This fix was found by Takashi Nishida.
+     */
    i = send_aux_command(sc->kbdc, PSMC_SET_DEFAULTS);
    if (verbose >= 2)
printf("psm%d: SET_DEFAULTS return code:%04x\n", unit, i);
+#endif
    if (sc->config & PSM_CONFIG_RESOLUTION) {
        sc->mode.resolution
    = set_mouse_resolution(sc->kbdc,
-           (sc->config & PSM_CONFIG_RESOLUTION) - 1);
+                  (sc->config & PSM_CONFIG_RESOLUTION) - 1);
+    } else if (sc->mode.resolution >= 0) {
+   sc->mode.resolution
+       = set_mouse_resolution(sc->kbdc, sc->dflt_mode.resolution);
+    }
+    if (sc->mode.rate > 0) {
+   sc->mode.rate = set_mouse_sampling_rate(sc->kbdc, sc->dflt_mode.rate);
    }
+    set_mouse_scaling(sc->kbdc, 1);

    /* request a data packet and extract sync. bits */
    if (get_mouse_status(sc->kbdc, stat, 1, 3) < 3) {
```

FreeBSD 3.2 より新しいリリースではきちんと動作するはずです。

2.16. ラップトップ PC のマウス/トラックボール/タッチパッドは使えますか?

前の質問に対する答えと、 モバイルコンピューティングのページをご覧ください。

2.17. どんなテープドライブをサポートしていますか?

FreeBSD は SCSI と QIC-36 (QIC-02 インタフェース付き) をサポートしています。 これらには 8-mm (Exabyte と呼ばれています) や DAT ドライブも含まれています。

初期の 8-mm ドライブの中には SCSI-2 とまったく互換性を持たないものがあります。 これらは FreeBSD 上では動作しません。

2.18. どんなテープチェンジャーをサポートしていますか?

FreeBSD 2.2 は ch(4) デバイスと chio(1) コマンドを使用した SCSI チェンジャーをサポートしています。実際のチェンジャーの制御方法の詳細は、chio(1) のマニュアルページを参照してください。

使用している製品が AMANDA のようにチェンジャーに対応済みのものでない場合は、次のことについて留意してください。
それらの製品は任意のポイント間のテープの移動を制御するだけなので、
テープがどのスロットに入っているか、現在ドライブにあるテープが
どのスロットに戻るべきかを把握しておく必要があります。

2.19. どんなサウンドカードをサポートしていますか?

FreeBSD は SoundBlaster、SoundBlaster Pro、SoundBlaster 16、 Pro Audio Spectrum 16、AdLib それから Gravis UltraSound サウンドカードを サポートしています。MPU-401 やその互換カードも機能に制限はあるものの
サポートされています。マイクロソフトサウンドシステムのスペックに準拠 したカードも、pcm ドライバでサポートされています。



これらはサウンドについてのみの話です! これらのドライバは CD-ROM、SCSI、カード上にあるジョイスティックをサポートしていません (SoundBlaster は例外です)。SoundBlaster SCSI インタフェースと非 SCSI CD-ROM はサポートしていますが、そのデバイスからは起動できません。

2.20. pcm ドライバで es1370 から音が出ないのはどうにかなりませんか?

マシンを起動するごとに以下のコマンドを実行してください。

```
# mixer pcm 100 vol 100 cd 100
```

2.21. どんなネットワークカードをサポートしていますか?

より完全な一覧についてはイーサネットカードの節を参照してください。

2.22. 数値演算コプロセッサを持っていませんが、何かまずいでしょうか?



これらは 386/486SX/486SLC を持っている場合に影響します - ほかのマシンでは CPU に内蔵されています。

一般にこれらは問題とはなりません。 しかし、数値演算エミュレーションコードのパフォーマンスか、正確さのいずれかを選択する状況があります (詳しくは FP エミュレーションについての節をご覧ください)。 とくに、X 上で弧を描く際にとても遅くなることでしょう。数値演算コプロセッサを購入されることを強くおすすめします。 とても役立つことでしょう。



他の数値演算コプロセッサよりも優れたコプロセッサもあります。
これは言いにくいことなのですが、Intel を買うために躍起になる人もいないでしょう。
それが FreeBSD 上で動くという確信がないのなら、クローンにご用心を。

2.23. FreeBSD がサポートするデバイスは他にもあるんでしょうか?

FreeBSD ハンドブックに記されている、 サポートされている他のデバイスの一覧を参照してください。

2.24. パワーマネージメント機能付きのラップトップ PC を持っているのですが...。

FreeBSD は一部のマシンの APM をサポートしています。 LINT カーネルコンフィグファイル の APM の部分をご覧ください。 さらに詳しいことは apm(4) に載っています。

2.25. Micron システムが起動時に固まってしまいます。

特定の Micron 製のマザーボードの中には、PCI BIOS が規格通りに 実装されていないために FreeBSD の起動に失敗するものがあります。 その BIOS は、PCI デバイスをあるアドレスで設定したと報告するにも 関わらず、実際にはそうしていないのです。

この問題を回避するには、BIOS の "Plug and Play Operating System" を無効に設定してください。また、より詳しい情報は http://cesdis.gsfc.nasa.gov/linux/drivers/vortex.html#micron を参照してください。

2.26. 新しい Adaptec コントローラを持っているのですが、FreeBSD が検出できないようです。

新しい AIC789x シリーズの Adaptec チップは、3.0 でデビューした CAM SCSI フレームワークでサポートされています。 2.2-STABLE のパッチは ftp://ftp.FreeBSD.org/pub/FreeBSD/development/cam/ にあります。 CAM システムが入っている高機能ブートフロッピーは http://people.FreeBSD.org/~abial/cam-boot/ にあります。 どちらの場合にしても、作業を始める前に README をお読みください。

2.27. 内蔵の Plug & Play モデムを持っているのですが、FreeBSD が検出できないようです。

モデムの PnP ID を シリアルドライバの PnP ID リストに追加する必要があるでしょう。 Plug & Play

サポートを有効にするには、 controller pnp0 をコンフィグレーション ファイルに付け加え、新しいカーネルをコンパイルしてシステムを再起動してください。
カーネルは、検出したすべてのデバイスの PnP ID を表示します。 モデムの欄にある PnP ID を /sys/i386/isa/sio.c の 2777 行目くらいにあるテーブルに書き入れてください。
テーブルを見つけるには、構造体 siopnp_ids[] の文字列 SUP1310 を探します。
カーネルを作り直したらインストールし、システムを再起動してください。
そうすれば、モデムが検出されるはずです。

起動時のコンフィグレーションの際に、pnp コマンドを使用して PnP の設定をマニュアルで行なわなければならないかもしれません。
その場合、モデムを検出させるためのコマンドは

```
pnp 1 0 enable os irq0 3 drq0 0 port0 0x2f8
```

のようになります。

2.28. シリアルコンソールで boot: プロンプトを表示するにはどうすればいい?

1. options COMCONSOLE を指定してカーネルを構築してください。
2. そして /boot.config を作成して -P とだけ書き入れてください。
3. その後、キーボードをシステムから抜きます。

/usr/src/sys/i386/boot/biosboot/README.serial に、 これに関する情報が書かれています。

2.29. なぜ Micron コンピュータで 3Com PCI ネットワークカードが動かないのでしょう?

特定の Micron 製のマザーボードの中には、PCI BIOS が規格通りに 実装されていないために FreeBSD の起動に失敗するものがあります。 その BIOS は、PCI デバイスをあるアドレスで設定したと報告するにも 関わらず、実際にはそうしていないのです。

この問題を回避するには、BIOS の "Plug and Play Operating System" を無効に設定してください。また、より詳しい情報は
http://cesdis.gsfc.nasa.gov/linux/drivers/vortex.html#micron を参照してください。

2.30. 対称型マルチプロセシング (SMP) をサポートしていますか?

SMP は、3.0-STABLE とそれ以降のリリースでのみサポートされています。 *GENERIC* カーネルでは SMP は有効化されていませんので、 SMP を有効化するにはカーネルを再構築する必要があります。
/sys/i386/conf/LINT を見て、
カーネルコンフィグファイルにどのオプションを追加すれば良いのか確かめてください。

2.31. ASUS K7V

マザーボードのシステムでブートフロッピーを使うと、システムがハングアップします。 対応策はありませんか?

BIOS セットアップで "起動時のウィルス保護機能" を無効化してください。

Chapter 3. トラブルシューティング

3.1. ハードディスクに不良ブロックがあります!

SCSI ディスクの場合は自動的に再マップする機能があるはずです。
しかし、理解し難い理由から多くのドライブがこの機能が無効化 されて出荷されています...。

これを有効化するには、 最初のデバイスのモードページを変更する必要があります。
これは次のコマンドを実行することで、FreeBSD 上で行なうことができます (root 権限で行ないます)。

```
# scsi -f /dev/rsd0c -m 1 -e -P 3
```

そして、AWRE と ARRE の値を 0 から 1 へ変更します

```
AWRE (Auto Write Reallocation Enbld):  1
ARRE (Auto Read Reallocation Enbld):  1
```

以下は、Ted Mittelstaedt 氏から寄せられたものです。

IDE ドライブの場合は通常、不良ブロックは潜在的な障害の兆候です。 最近の IDE ドライブは、内部の不良ブロック再マッピング機能を有効にした状態で 出荷されています。また、今日の IDE ハードディスクメーカは、 出荷以降に不良ブロックが発生することに関して保証を提供していて、不良ブロックのあるディスクドライブを交換するサービスを行なっています。

もし、不良ブロックのある IDE ディスクドライブを復旧しようと思うなら、 IDE ドライブメーカが提供する IDE 診断プログラムをダウンロードして、そのドライブに使ってみてください。この種のプログラムは大抵、
ドライブの制御部分に対して不良ブロックを再走査し、
不良ブロックを使用不能にするようにセットすることができます。

ESDI、RLL および MFM ドライブの場合、 不良ブロックはドライブの正常な部分であり、一般的に言って障害を表すものではありません。 PC では、ディスクドライブコントローラカードと BIOS が不良ブロックの使用不能化の作業を行ないます。 DOS など、ディスクアクセスに BIOS を経由する OS にとっては有効に働きますが、FreeBSD のディスクドライバは BIOS を利用しません。そのため、代替として bad144 という機構が存在します。 bad144 は、wd ドライバでだけ (つまり FreeBSD 4.0 ではサポートされていない)動作し、SCSI ドライバに利用することは できません。bad144 は、検出された不良セクタをスペシャルファイルに記録するという機能を持っています。

bad144 を利用する上で、注意しなければならない点が一つあります。
それは、不良ブロックスペシャルファイルは、 ディスクの最終トラックに置かれるということです。
このファイルには、ディスクの先頭の付近、 /kernel ファイルが位置しているであろう部分で発生した不良セクタが記録されています。
したがって、このファイルは BIOS コールを使ってカーネルファイルを読み込む起動プログラムが、アクセス可能でなければなりません。 これはつまり、bad144 を利用するディスクは 1024 シリンダ、16 ヘッド、63 セクタを超えてはならないということを意味し、 bad144 を利用したディスクが実質 500MB を超えられないことになります。

bad144 を使うには、FreeBSD のインストール時に表示される fdisk 画面で "Bad Block" 走査を ON に設定するだけです。 これは、FreeBSD 2.2.7 以降で機能します。 ディスクは、1024 シリンダ以内でなければなりません。 ディスクドライブは事前に少なくとも 4 時間、ディスクが温度によって膨張し、 トラックに曲がりが出るまで回転させることをお薦めします (訳注: 温度変化に対する膨張によって、
ディスクが微小変形することにより発生する不良セクタを確実に検出するためです)。

大容量の ESDI ドライブのように 1024 シリンダを超えるディスクの場合、 DOS 上でそのディスクが利用できるよう、 ESDI コントローラは特殊な変換モードを利用します。 fdisk の "set geometry" コマンドを使って "変換された (translated)" ジオメトリに切替えると、wd ドライバはこの変換モードを解釈できます。 その際、FreeBSD パーティションを作成するのに "dangerously dedicated" モードを利用してはいけません。
このモードは、そのようなジオメトリを無視するからです。 たとえ fdisk がオーバーライドされたジオメトリ情報を使ったとしても、
依然としてディスクの真の大きさを保持しているため、大きすぎる FreeBSD パーティションを作成しようとしてしまうでしょう。
ディスクジオメトリ情報が変換されたジオメトリ情報にかわっている場合は、
手動でブロック数を入力し、 パーティションを作成する必要があります。

大容量の ESDI ディスクを ESDI コントローラでセットアップするには、ちょっとしたトリックを使います。まず、DOS のディスクで起動して そのディスクを DOS パーティションとしてフォーマットします。 そして FreeBSD を起動し、インストーラの fdisk 画面で DOS パーティションのブロックサイズとブロック数を読みとり、メモしておきます。 ジオメトリ情報を DOS が利用しているものと同一に再設定し、 DOS パーティションを削除して "cooperative" FreeBSD パーティションを 先程記録したブロックサイズを使って作成してください。
そのパーティションを起動可能パーティションに設定し、不良ブロック走査を 有効にします。
実際のインストールでは、ファイルシステムが作成される前に bad144 が最初に実行されます (Alt-F2 を押すことで状況を確認できます)。 不良セクタファイルを作成中に何らかの障害が発生したなら、システムを再起動して、もう一度最初からやり直しになります。
おそらくディスクジオメトリ情報の設定を大きくしすぎているのでしょう (やり直しは、DOS によるフォーマットとパーティション確保を含みます)。

もし、不良ブロックの再マッピングを有効にしていて不良ブロックが見付かったら、
ドライブの交換を考えてください。不良ブロックは、時間とともに悪化するからです。

3.2. Bustek 742a EISA SCSI が認識されません。

この情報は 742a のためのものですが、他の Buslogic カードについても 同様のことが言えます。(Bustek = Buslogic)

742a カードには大きくわけて 2 つの「バージョン」が存在します。 ハードウェアリビジョンの A-G と H 以降です。リビジョンの 文字はカードの隅にあるアセンブリ番号の後ろにあります。 742a は二つの ROM チップを持っており、一つは BIOS チップで もう一つはファームウェアチップです。FreeBSD はあなたの 持っているものがどの BIOS バージョンかは問題ありませんが、ファームウェアバージョンについては問題となります。 Buslogic の技術サポート部門に連絡すれば、アップグレード版の ROM を送ってくれることでしょう。BIOS チップと ファームウェアチップはペアで出荷されます。
アダプタカードのハードウェアリビジョンにあわせた 最も新しいファームウェア ROM を使用しなければなりません。

リビジョン A-G のカードには、2.41/2.21 までの BIOS/ファームウェアのセットを使用することができます。 リビジョン H 以降のカードには、最新のものである 4.70/3.37 の BIOS/ファームウェアのセットを使用することができます。これらのファームウェアの違いは、 ファームウェア 3.37 が「ラウンドロビン方式」をサポートしているところからきています。

Buslogic のカードには、製造番号も刻印されています。古いハードウェアリビジョンのカードを持っている場合は、Buslogic の RMA 部門に問い合わせて製造番号を伝えると、新しいハードウェアリビジョンの カードに交換することもできます。もしカードが十分新しければ、彼らは 交換に応じてくれるでしょう。

FreeBSD 2.1 は ファームウェアリビジョン 2.21 以降のものをサポートしています。これよりも古いファームウェアリビジョンのものは、 Buslogic カードとして正常に認識されません。しかし、Adaptec 1540 として認識されるかもしれません。 初期の Buslogic のファームウェアは AHA1540 「互換」モードを 持っています。しかし、EISA カードにとってこれはよいことではありません。

古いハードウェアリビジョンのカードを持っていてファームウェア 2.21 を入手するのであれば、ジャンパ W1 の位置をデフォルトの A-B から B-C に合わせる必要があるでしょう。

3.3. HP Netserver 上のオンボード SCSI コントローラが認識されません。

基本的にこれは既知の問題です。HP Netserver マシンの EISA オンボード SCSI コントローラは EISA のスロット番号 11 を占有しますが、「本当の」EISA スロットはすべてそれよりも前のアドレスに配置されているのです。 残念ながら、 10 番以上の EISA スロットは PCI に割り当てられたアドレス空間と衝突し、FreeBSD の自動コンフィグレーションは、現状ではうまくこの状況を処理できていないのです。

ですから現時点での最良の方法は、カーネルオプションの `EISA_SLOTS` を 12 に変え、アドレス空間の衝突がないかの ようなふりをさせることです :) カーネルの再構築に記述されているようにしてカーネルを再構築してください。

もちろん、これはこのようなマシンにインストールする際に 「卵が先か、鶏が先か」といった問題を生み出すことになります。 この問題を回避するために、 ユーザコンフィグ *(UserConfig)* の中には特別な仕組みが組み込まれています。 このとき "visual" インタフェースは使用せず、 コマンドラインインタフェースを使用してください。単純に

```
eisa 12
quit
```

とプロンプト上から打ち込み、 後は普通にインストールを行なってください。とにかくカスタムカーネルのコンパイルとインストールを行なうことを おすすめします。

うまくいけば、将来のバージョンではこの問題が解決していることでしょう。



HP Netserver では危険覚悟の専用ディスクは使用できません。 詳細については この注意事項をご覧ください。

3.4. この CMD640 IDE コントローラはどこかおかしいようです。

それは壊れているのです。両方のチャンネルを同時に制御できないのです。

現在、このチップを使っているシステムを自動的に検出して、うまく動かすためのしくみが使えるようになっています。 くわしくは wd(4) のマニュアルページを参照してください。

CMD640 IDE コントローラを使っているシステムで FreeBSD 2.2.1 あるいは 2.2.2 を使い、かつセカンダリのチャネルを使いたいのであれば、 `options "CMD640"` を有効にしてカーネルを作り直してください。 FreeBSD 2.2.5 以降では、デフォルトでそうなっています。

3.5. ed1: timeout のようなメッセージがいつも出ます。

たぶん IRQ の衝突が原因でしょう (二つのボードが同じ IRQ を使用しているなど)。FreeBSD 2.0.5R 以前はこれに関して寛大で、 IRQ の衝突があってもネットワークドライバは機能していました。 しかし 2.0.5R 以降はもはや、IRQ の衝突に寛大ではありません。 `-c` オプションをつけて起動し、 ed0/de0/... のエントリをボードの設定に合わせてください。

ネットワークカードの BNC コネクタ (訳注: 10BASE-2 タイプのインタフェース) を使っている場合、デバイスのタイムアウトはターミネーションの不良によっても起きます。これをチェックするにはケーブルを外してターミネータを直接 NIC に接続します。そしてエラーメッセージが消えるかどうか 確認します。

NE2000 コンパチブルカードのなかには、 UTP ポートのリンクがなかったりケーブルが接続されていない場合に このエラーを出すものがあります。

3.6. CDROM をマウントしようとすると Incorrect super block と言われます。

mount(8) にマウントしたいデバイスのタイプを指定する必要があります。 デフォルトでは mount(8) はファイルシステムを `ufs` とみなします。CDROM のファイルシステムを マウントしたいのであれば `-t cd9660` と mount(8) にオプションをつけて明示する必要があります。 これはもちろん CDROM が ISO 9660 ファイルシステムである場合です。ほとんどの CDROM はこの形式です。1.1R の FreeBSD では (訳注: 2.1.5R、 2.2R でも同様です) 自動的に Rock Ridge 拡張 (長いファイル名への対応) をうまく解釈します。

CDROM のデバイス /dev/cd0c を /mnt にマウントしたい場合の例では、次のようにします。

```
# mount -t cd9660 /dev/cd0c /mnt
```

デバイスの名前はインタフェースによっては別の名前になっている かもしれないので注意してください (/dev/cd0c はこの場合の例です)。 オプション `-t cd9660` によって `mount_cd9660` コマンドが実行されることに注意してください。 このため例は次のようにすることもできます。

```
# mount_cd9660 /dev/cd0c /mnt
```

3.7. CDROM をマウントしようとすると Device not configured と言われます。

これは 一般的に CDROM ドライブの中に CDROM が入っていないか、ドライブがバス上に見えないことを意味します。ドライブに CDROM を入れるか、IDE (ATAPI) であれば master/slave の状態をチェックしてください。 また、CDROM ドライブに CDROM を入れてから認識するまでには数秒かかりますので、 少し待ってみてください。

SCSI CDROM ではバスリセットへの応答時間が遅いために、 失敗することがあるかもしれません。 SCSI CDROM を持っている場合は、 カーネルコンフィグレーションファイルに以下の行を加えて再コンパイルして試してみてください。

```
options "SCSI_DELAY=15"
```



現在の GENERIC カーネルでは上の設定はデフォルトになっています。問題がある場合は SCSI_DELAY の数値を増やしてみてください。

3.8. CDROM をマウントすると、ファイル名中の英数字以外の 文字が、? と表示されてしまいます。

もっともありそうなのは、その CDROM が "Joliet" 拡張を利用してファイルおよびディレクトリに関する情報を保存しているということです。この拡張は、 すべてのファイル名を Unicode の 2 バイト文字で保存するように 規定しています。現在、FreeBSD カーネルに汎用的な Unicode インタフェースを導入する作業が行われていますが、 まだ完了していません。したがって、CD9660 ドライバはファイル名の文字を解読できません。

一時的な解決策として、FreeBSD 4.3R 以降では、CD9660 ドライバに特別な仕掛けを施して、ユーザーがその場で適切な 変換表を読み込めるようにしました。一般的なエンコーディングに 対応したいくつかのモジュールが sysutils/cd9660_unicode port で提供されています。



この記述は古くなっています。英語版の記述をご覧ください。

3.9. 私のプリンタはとてつもなく遅いのです。どうしたらよいのでしょう?

パラレルインタフェースで、問題はとんでもなく遅いだけであるなら、 プリンタポートを "polled" モードに設定してみてください。

```
# lptcontrol -p
```

HP の新しいプリンタには、割り込みモードで使えないものがあるようです (完全にわかったわけではありませんが)。 タイミングの問題のように思われます。

3.10. わたしのプログラムは時々 Signal 11 のエラーで止まってしまいます。

Signal 11 エラーはオペレーティングシステムが許可を与えていないメモリにアクセスしようとしたときに発生します。
このようなことがランダムな間隔で起っているようなら、 注意深く調査していった方が良いです。

この手の問題はたいていの場合、以下のどちらかです。

1. その問題が特定の、あなたが自分で開発したアプリケーションでのみ起っているなら、あなたのコードにバグがあるのでしょう。
2. それが FreeBSD のベースシステムの一部と関連する問題なら、コードにバグがあるということになります。 しかしほとんどの場合、普通の FAQ の読者がそのようなコードを使うようになるずっと前に、
   そういった問題は発見され、修正されているはずです (それが -current の役目なのですから)。

それが FreeBSD のバグでは「ない」という決定的なケースとして、その問題の発生がプログラムをコンパイルしているときであり、
コンパイル毎に毎回、コンパイラの挙動が変るというものがあります。

たとえば、あなたが "make buildworld" を実行していて、 コンパイラが ls.c から ls.o をコンパイルしようとしたときに コンパイルに失敗したとします。もう一度 "make buildworld" を実行したときに、まったく同じ場所でコンパイルが失敗したのなら、 それは build が壊れている (訳注: つまりソースにバグがある) と言うことです — ソースを更新してやりなおしてみてください。
もしコンパイルが別の場所でしくじっていたら、 それはハードウェアの問題です。

あなたのやるべき事は:

前者の場合は、 そのプログラムの間違ったアドレスへアクセスしようとしている部分を、 gdb 等のデバッガで見つけて修正します。

後者の場合は、 ハードウェアに問題がないことを確かめる必要があります。

その一般的な原因として :

1. ハードディスクが熱を持ちすぎているかも知れません:
   ケースのファンがちゃんと動いていてディスクを冷やしているか 確かめてください (たぶん、他の部品も過熱しています)。
2. CPU がオーバーヒートしています: CPU をオーバークロックしていませんか? さもなければ CPU ファンが死んでいるのかもしれません。 いずれにせよ、少なくとも問題解決の間ではハードウェアが動くべく指定された条件で動かしてください。
   クロックはデフォルトの設定に戻してください。

   もしあなたがクロックアップをしているのなら、 遅いシステムでも、システムが焼き付いて

買い換えなければならなくなるよりずっとマシだということを 覚えておいた方が良いでしょう。
大きいコミュニティでは特に、 あなたがそれが安全だと思っているかどうかは関係なく、
オーバークロックしたシステムに発生した問題には同情的ではありません。

3. 怪しいメモリ: もし複数の SIMM や DIMM を使っているならそれを全部抜いてから 各 SIMM や DIMM を別個に組み込んだシステムを立ち上げてることで どの DIMM/SIMM が怪しいのか、それとも組合わせが悪いのか と問題の幅が狭まります。

4. 楽観的すぎるマザーボードの設定: ほとんどの場合に標準設定で十分なタイミングを、 BIOS の設定やマザーボード上のジャンパピンを変えることで、
さまざまに変更することができます。しかし時には RAM の アクセスウェイトを低くしすぎたり "RAM Speed: Turbo" や その手の BIOS の設定でおかしな挙動が起こることがあります。 BIOS を標準の設定に戻すというのはいいアイディアですが、
その前にあなたの設定を書き留めておいた方がいいでしょう。

5. マザーボードへの電源が安定していない。 もし使っていない I/O ボードやハードディスク、 CDROM 等があるなら、一旦それらから電源ケーブルを抜き、
電源が小さな負荷ならなんとか動作するか確認しましょう。 あるいは別の電源を試してみましょう。
その時はなるべく、少し容量の大きいもので試しましょう (たとえば、今の電源容量が 250W だったら 300W のものを試します)。

SIG11 FAQ (下に示します) にはこれらの問題のすべてが 詳しく説明されています。Linux の視点に基づくものですが、 これも読んでおいた方がいいでしょう。そこではまた、
メモリのテストを行うソフトウェアや、
ハードウェアがなぜ問題のあるメモリを見逃してしまうかについても 議論されています。

最後に、これらがどれも助けにならなかったら、 FreeBSD のバグを発見した可能性があります。
以下の説明を読んで障害報告を送ってください。

詳細な FAQ は、the SIG11 problem FAQ にあります。

3.11. 起動の時に画面が真っ暗になって同期も取れません。

これは ATI Mach 64 ビデオカードの既知の問題です。 この問題はカードがアドレス 2e8 を使い、 4 番目のシリアルポートもここを使うということにあります。 sio(4) ドライバのバグ (仕様?) のため、 4 番目のシリアルポートがなくても、 通常このアドレスを使う sio3 (4 番目のポートにあたります) を無効にしても、ドライバはこのアドレスをさわります。

バグが修正されるまでは、次のようにして対処してください。

1. 起動プロンプトが出たら -c と入力します
(これによりカーネルはコンフィグレーションモードに入ります)。

2. sio0, sio1, sio2, sio3 (これらすべて) を無効にします。 これによって sio(4) ドライバは動作しなくなりますが、問題はありません。

3. exit と入力して起動を続行します。

もしシリアルポートを有効にしたいのであれば以下の変更を行なって
新しいカーネルを作る必要があります。 /usr/src/sys/i386/isa/sio.c の中で 1 ヵ所ある 0x2e8 という文字列を探し、 この文字列とその手前にあるコンマを削除します (後ろのコンマは残します)。
後は通常の手続きにしたがって新しいカーネルを作ります。

この対処を行なった後でもまだ X ウィンドウシステムはうまく動かないかもしれません。 その場合は、使用している XFree86 がすくなくとも XFree86 3.3.3 以降であることを確かめてください。それ以降のバージョンでは、 Mach64 カードやそれらのカードのためにつくられた X サーバの組込みをサポートします。

3.12. 128MB の RAM があるのですが、64MB しか認識しません。

FreeBSD がメモリのサイズを BIOS から取得する方法の制限により、 KB 単位で 16 ビット分までしか検出できません (すなわち最大 65535KB=64MB です。これより少ない場合もあります。 ある BIOS の場合はメモリサイズが 16MB に制限されます)。64MB 以上のメモリを積んでいる場合、 FreeBSD はそれを検出しようとします。しかしその試みは失敗するかもしれません。

この問題を回避するには、 以下に示すカーネルオプションを使用する必要があります。完全なメモリ情報を BIOS から取得する方法もありますが、起動ブロックに空きが無いため実装できません。 起動ブロックの問題が解決されれば、 いつか拡張 BIOS 機能を使用して完全なメモリ情報を取得できるようになるでしょう。
とりあえず現在は、カーネルオプションを使ってください。

```
options "MAXMEM=n"
```

n には、 キロバイト単位でメモリの量を指定します。128MB の場合は、131072 となります。

3.13. FreeBSD 2.0 が kmem_map too small! と言ってパニックします。



メッセージは、mb_map too small! の場合もあります。

このパニックは、ネットワークバッファ (特に mbuf クラスタ) の仮想メモリが無くなったことを示します。 以下のオプションをカーネルコンフィグファイルに追加して mbuf クラスタに使用できる仮想メモリの量を増やしてください。

```
options "NMBCLUSTERS=n"
```

n には、 同時に使用したい TCP コネクションの数に応じて 512 から 4096 までの数値を指定できます。とりあえず 2048 を試してみるのをおすすめします。 これでパニックは完全の予防できるはずです。mbuf クラスタの割り当て、使用状況については、 netstat -m で知ることができます (netstat(1) をご覧ください)。 NMBCLUSTERS のデフォルト値は 512 + MAXUSERS * 16 です。

3.14. 新しいカーネルで再起動すると CMAP busy panic となってパニックを起こしてしまいます。

ファイル /var/db/kvm_*.db において範囲外のデータを検出するためのロジックは失敗することがあり、こうした矛盾のあるファイルを使用することでパニックを引き起こすことがあります。

これが起こったなら、シングルユーザで再起動した後に、 以下のコマンドを実行してください。

```
# rm /var/db/kvm_*.db
```

3.15. ahc0: brkadrint, Illegal Host Access at seqaddr 0x0 というエラーが出ます

これは Ultrastor SCSI Host Adapter と衝突しています。

起動時に kernel configuration メニューに入り、 問題を起こしている uha0 を disable にしましょう。

3.16. sendmail が mail loops back to myself というメッセージを出すのですが。

この事は、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。たとえば:

    553 relay.domain.net config error: mail loops back to myself
    554 <user@domain.net>... Local configuration error

    のような物ですが、どのようにしたらこの問題を解決できますか?

    これは、たとえば domain.net のようなドメイン宛てのメールを MX record 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/sendmail.cw の中に domain.net を
    追加してください。もしくは、/etc/sendmail.cf の中に
    "Cw domain.net" を追加してください。
```

もはや現在の sendmail FAQ は sendmail release とは一緒には保守されていません。しかし次のネットニュースに定期的に投稿されてます。 comp.mail.sendmail、 comp.mail.misc、 comp.mail.smail、 comp.answers、 news.answers。 また、メール経由でコピーを入手する場合は mail-server@rtfm.mit.edu 宛まで本文に `send usenet/news.answers/mail/sendmail-faq` と書いて送ります。

3.17. リモートマシン上のフルスクリーンアプリケーションがうまく動かない

リモートマシンのターミナルタイプが FreeBSD のコンソールで必要とされている `cons25` 以外のものです。

この問題を解決しうる方法はいろいろあります:

- リモートマシンにログインした後、 そのリモートマシンが `ansi` か `sco` のターミナルタイプを知っているなら、 shell 変数の TERM にそれらのいずれかを設定します。

- FreeBSD のコンソール側で screen のような VT100 エミュレータを使用します。 screen は一つのターミナルの中で複数のセッションを並列動作させることができますし、本来の機能も優れています。 各々の screen のウィンドウは VT100 ターミナルのように振る舞うので、 リモート側で設定されるべき TERM 変数は `vt100` となります。
- リモートマシンのターミナルデータベースに `cons25` のエントリをインストールします。このインストール方法はリモートマシンのオペレーティングシステムに依存します。リモートのシステムのシステム管理マニュアルが役に立つことでしょう。
- FreeBSD 側で X サーバを起動して、 リモートマシンに `xterm` や `rxvt` のような X ベースのターミナルエミュレータを使ってログインします。 (訳注: 日本語が必要な場合は `kterm` 等を利用します) リモートホストの TERM 変数は `xterm` もしくは `vt100` (訳注: もしくは `kterm`) に設定します。

3.18. 私のマシンで calcru: negative time... と表示されるのですが

これは、割り込みに関連するさまざまな不具合によって発生します。
あるいは、あるデバイスが元々持っているバグが表面化したのかも知れません。
この症状を再現させる一つの方法として、パラレルポート上で、 TCP/IP を 大きな MTU で走らせるというものがあります。 グラフィックアクセラレータがこの症状を起こすことがありますが、その場合はまず、カードの割り込み設定を確認してください。

この問題の副作用として、 プロセスが "SIGXCPU exceeded cpu time limit" というメッセージとともに終了してしまう、というものがあります。

1998 年 11 月 29 日に公開された FreeBSD 3.0 以降で この問題が解決しないなら、次の sysctl 変数をセットしてください。

```
# sysctl -w kern.timecounter.method=1
```

これは、パフォーマンスへ強い影響を与えますが、
問題の発生に比べればおそらく気にならない程度でしょう。 もし、これでもまだ問題が残るようなら、カーネルオプションの `NTIMECOUNTER` を大きな値に増やしてください。 `NTIMECOUNTER=20` にまで増やしても解決しない場合は、 計時処理の信頼性が保てない程の割り込みが、そのマシン上で起こっていることを意味します。

3.19. pcm0 not found という表示を見たり カーネルコンフィグレーションファイルには device pcm0 と 書いてあるのにサウンドカードが pcm1 として 発見されたりします。

これは FreeBSD 3.x で PCI のサウンドカードを使っているときに 発生します。`pcm0` デバイスは ISA のカード専用に予約されているものです。このため、 あなたが PCI カードを持っているときはこのエラーが表示され、 カードは `pcm1` として検出されます。



この警告を、単にカーネルコンフィグファイルの当該行を `device pcm1`

> に変更することで 抑制することはできません。その時は pcm1 が ISA カードのために予約され、PCI のカードは pcm2 として (pcm1 not found の警告とともに) 検出されます。

PCI のサウンドカードを持っているのならば、以下のようにして snd0 デバイスのかわりに snd1 を作る必要があります。

```
# cd /dev
# ./MAKEDEV snd1
```

この状況は FreeBSD 4.x では生じません。多くの努力の結果より *PnP* 中心に作り替えられ、 現在、pcm0 デバイスは ISA カード専用に予約されたものではなくなりました。

3.20. プラグアンドプレイのカードが認識されなくなりました (または、unknown と認識されるようになりました)。

現在の FreeBSD 4.x はより *PnP* 中心に なっています。その副作用の影響で、FreeBSD 3.x で動いていた PnP デバイス (たとえばサウンドカードや内蔵モデム) の中には、動かなくなってしまったものもあります。

この挙動の原因は Peter Wemm が freebsd-questions メーリングリストに書いた、以下の 「FreeBSD 4.x にアップグレードしたところ内蔵モデムが 見つからなくなった」というメールで解説されています。(わかりやすくするために [] 内に コメントを加えました)。

PnP BIOS はあらかじめ、[モデムを] ポート空間に存在しているかのように設定します。 そのため [3.x では] 従来の手法に基づく ISA デバイスの検索により、モデムの存在を「発見」できます。

4.0 の ISA コードは、より PnP 中心になっています。 [3.x では] ISA デバイスの検索が「はぐれた」デバイスを発見して、 次に PNP デバイス ID のマッチが行なわれることでリソースの競合が発生し、 デバイスの検索に失敗する可能性があります。したがって、4.0 の ISA コードでは 二重に検索しないよう、プログラマブルなカードを最初に無効にしています。 これは、対応している PnP ハードウェアの PnP ID が、予めわかっている必要がある、ということを意味します。
ユーザがこの挙動にもっと手を入れられるようにすることが TODO リスト中にあげられています。

3.0 で動作していたデバイスを 4.0 でも動作するようにするには、 それの PnP ID を調べ、ISA デバイスの検索が PnP デバイスの識別に使っているリストにそれを追加する必要があります。デバイスの検索に使われる pnpinfo(8) を用いて、 PnP ID を得ることができます。たとえば、内蔵モデムに関する pnpinfo(8) の出力は、 以下のようになります。

```
# pnpinfo
Checking for Plug-n-Play devices...

Card assigned CSN #1
Vendor ID PMC2430 (0x3024a341), Serial Number 0xffffffff
PnP Version 1.0, Vendor Version 0
Device Description: Pace 56 Voice Internal Plug & Play Modem
```

```
Logical Device ID: PMC2430 0x3024a341 #0
        Device supports I/O Range Check
TAG Start DF
    I/O Range 0x3f8 .. 0x3f8, alignment 0x8, len 0x8
        [16-bit addr]
    IRQ: 4  - only one type (true/edge)
```

```
TAG End DF
End Tag

Successfully got 31 resources, 1 logical fdevs
-- card select # 0x0001

CSN PMC2430 (0x3024a341), Serial Number 0xffffffff

Logical device #0
IO:  0x03e8 0x03e8 0x03e8 0x03e8 0x03e8 0x03e8 0x03e8 0x03e8
IRQ 5 0
DMA 4 0
IO range check 0x00 activate 0x01
```

必要な情報は、出力の冒頭にある "Vendor ID" 行にあります。 かっこの中の 16 進数 (例の中では 0x3024a341) が PnP ID で、 直前の文字列 (PMC2430) はユニークな ASCII ID です。この情報はファイル /usr/src/sys/isa/sio.c に 追加する必要があります。

まず失敗したときに備えて sio.c の バックアップを取るべきです。障害報告を送るために修正パッチを作る時にも必要になるでしょう (send-pr しようとしていますよね?)。 sio.c を編集して以下の行を探してください。

```
static struct isa_pnp_id sio_ids[] = {
```

そしてあなたのデバイスのエントリを追加する正しい場所を探します。
エントリは以下のような形をしていて、pnpinfo(8) の 出力にある デバイスの説明の全部 (もし収まれば) か一部とともに行の右の方のコメント領域に書かれている ASCII ベンダ ID でソートされています。

```
{0x0f804f3f, NULL},     /* OZO800f - Zoom 2812 (56k Modem) */
{0x39804f3f, NULL},     /* OZO8039 - Zoom 56k flex */
{0x3024a341, NULL},     /* PMC2430 - Pace 56 Voice Internal Modem */
{0x1000eb49, NULL},     /* ROK0010 - Rockwell ? */
{0x5002734a, NULL},     /* RSS0250 - 5614Jx3(G) Internal Modem */
```

あなたのデバイスの16進数のベンダ ID を正しい場所に追加し、ファイルをセーブしてカーネルを作り直して再起動します。 あなたのデバイスは FreeBSD 3.x の時と同じように sio として見つかるようになっているはずです。

3.21. top や systat の 実行中に nlist failed というエラーがでます。

このエラーは、 実行しようとしたアプリケーションが あるカーネルシンボルを検索した結果、何らかの理由でその検索に失敗した、ということを意味しています。
これは、以下に示すいずれかの理由によるものです。

- カーネルとユーザランドが同期していない (つまり カーネルは新しいものを構築したが、`installworld` は行なっていない。 あるいはその逆) ので、シンボルテーブルがユーザアプリケーションの考えているものと異なっている。
  もしこのケースなら、一連のアップグレード手順に従ってアップグレードを行なってください (正しいやり方は /usr/src/UPDATING に書いてあります)。
- カーネルをロードするのに `/boot/loader` を使わず、 直接 boot2 (boot(8) 参照) からロードしている。
  もちろん `/boot/loader` を使わなくとも問題はないのですが、 `/boot/loader` は一般的に、ユーザアプリケーションからカーネルシンボルを
  アクセスできるようにするための機能を持っています。

3.22. ssh(1) や telnet(1) でコンピュータに接続するのに、どうしてこんなに時間がかかるのですか?

症状: TCP コネクションが確立してから、 クライアントソフトウェアがパスワードを尋ねてくるまで (telnet(1) の場合は、ログインプロンプトが表示されるまで) に長い時間がかかる、というもの。

問題: おそらく、サーバソフトウェアがクライアントの IP アドレスからホスト名を解決しようとして、遅れが生じている のでしょう。FreeBSD に付属する SSH や Telnet を含む多くの サーバソフトウェアは、この名前解決をおこないます。これは、管理者が後日参照するログファイルに、その他の情報と一緒に
ホスト名を記録できるようにするのが目的です。

対処法: もし、あなたのコンピュータ (クライアント) からどのサーバに接続する場合にも問題が起こるのであれば、
クライアントに問題があります。そして、誰かがあなたの コンピュータ (サーバ) に接続するときだけ問題が起こるのであれば、 そのサーバの問題です。

問題がクライアントにある場合、唯一の対処法は サーバがそのクライアントの名前を解決できるように DNS を修正することです。 症状がローカルネットワークで発生しているなら、サーバの設定に原因がありますので、このまま続きを読みましょう。
そうではなく、グローバルなインターネット環境で発生しているなら、 ISP に連絡して問題の修正をお願いしなければならない可能性が高いでしょう。

問題がサーバにあって、症状がローカルネットワークで
発生しているなら、ローカルのアドレス範囲にあるアドレスを、
それに対応するホスト名に解決する問合せを処理できるように、 サーバを設定する必要があります。詳しくは、hosts(5) および named(8) のマニュアルをご覧ください。グローバルなインターネット環境の場合は、
サーバのリゾルバが正しく動作していないのが原因かもしれません。 確認するには、他のホスト (たとえば `www.yahoo.com`) を引いてみてください。
うまくいかなければ、あなたのコンピュータの問題です。

3.23. file: table is full という メッセージが繰り返し dmesg にあらわれます。

このエラーは、システムのファイル記述子を使い果たして
しまった時に発生します。メモリ中のファイルテーブルが一杯に なっているのです。

解決法:

手動で sysctl 変数 kern.maxfiles の限界値を調整します。

```
# sysctl -w kern.maxfiles=n
```

n は、システム要件に合わせてください。 オープンされたファイル、ソケットまたは fifo のそれぞれが
ファイル記述子を消費します。規模の大きなサーバは、
同時に実行されるサービスに応じて、いともたやすく何万もの ファイル記述子を要求します。

カーネルに設定されたデフォルトのファイル記述子の 数を決定するのは、次の

```
maxusers        32
```

カーネル設定ファイルの maxusers 行 です。kern.maxfiles はこの値に比例して 増加します。

現在設定されている kern.maxfiles の 値は、次のコマンドで調べることができます。

```
# sysctl kern.maxfiles
kern.maxfiles: 1064
```

3.24. laptop の時間が狂って、大きく進んだり遅れたりします。

laptop には二つ以上の時計が内蔵されていますが、FreeBSD が間違った方を選択して使用しています。

dmesg(8) を実行して Timecounter を含む行を確認してください。 最後に出力された行が FreeBSD が選択したもので、まず間違い なく TSC でしょう。

```
# dmesg | grep Timecounter
Timecounter "i8254"  frequency 1193182 Hz
Timecounter "TSC"  frequency 595573479 Hz
```

sysctl(3) 変数 kern.timecounter.hardware を確認すれば 裏付けがとれます。

```
# sysctl kern.timecounter.hardware
kern.timecounter.hardware: TSC
```

バッテリ駆動している時に、BIOS が CPU の速度を変えるために TSC クロックを変更したり、電力節約モードに入ることがあります。しかし、FreeBSD はそういった調整を関知しないので、 時間が早まったり遅れたりするようです。

上記の例では、i8254 クロックも利用できます。sysctl(3) 変数 kern.timecounter.hardware にその名称を書き込んで選択できます。

```
# sysctl -w kern.timecounter.hardware=i8254
kern.timecounter.hardware: TSC -> i8254
```

これで、laptop はより正確な時間を刻むでしょう。

この変更を起動時に自動で行うには、次の行を /etc/sysctl.conf に追加してください。

```
kern.timecounter.hardware=i8254
```

3.25. BIOS 画面が出た後、FreeBSD のブートローダが Read error と表示して止まって しまいます。

FreeBSD のブートローダがハードディスクのジオメトリを正しく 認識していないようです。FreeBSD のスライスを fdisk によって手動で作成したり変更したりする際に、ジオメトリを誤って指定してしまったのでしょう。

ハードディスクのジオメトリの正しい値は、マシンの BIOS から得られます。そのハードディスクのシリンダ、ヘッド、セクタの 数を探してください。

sysinstall(8) の fdisk において、 G を入力してハードディスクのジオメトリを 設定してください。

シリンダ、ヘッド、セクタの数を入力するダイアログが出てきます。 BIOS から得た値を斜線 (/) で区切って入力してください。

5000 シリンダ、250 ヘッド、60 セクタなら、 5000/250/60 と入力します。

リターンキーを押して値を設定してください。それから W を入力してハードディスクに新しいパーティションテーブルを書き込んでください。

3.26. 別のオペレーティングシステムが、ブートマネージャを壊してしまいました。どうすれば復旧できるでしょうか。

sysinstall(8) を立ち上げて Configure (設定)、Fdisk の順に選択してください。ブートマネージャが置かれていた ディスクを選択して、スペースキーを押してください。W を押して変更を ディスクに書き込んでください。どのブートローダをインストールするか尋ねられます。ここで選択すれば戻せます。

Chapter 4. 商用アプリケーション



この章はまだまだ情報が足りません。
情報を追加してくれるような企業を待ち望んでいます。 FreeBSD グループはここに載っている企業からの金銭的な支援を期待してはいませんので、奉仕作業の一つとして掲載しています (そして FreeBSD が係わる宣伝は、長い目で見ると FreeBSD に対してよい方向へ働くと思っています)。
私たちは商用ソフトウェアベンダに、ここで製品を宣伝してもらうことを望んでいます。詳しくは、商用ソフトウェアベンダ覧のページをご覧ください。

4.1. FreeBSD 用のオフィススイートはどこで入手できますか?

- BSDi は FreeBSD ネイティブ版の VistaSource ApplixWare 5 を提供しています。

  ApplixWare は、豪華で機能満載の FreeBSD 向けの商用オフィススイートで、ワードプロセッサ、表計算、プレゼンテーションソフトウェア、ベクタ描画ソフトウェア、その他のアプリケーションを揃えています。

  FreeBSD 版の ApplixWare の購入は こちらからどうぞ。

- Linux 版の StarOffice は FreeBSD で完璧に動作します。Linux 版の StarOffice をインストールするもっとも簡単な方法は、FreeBSD Ports コレクションを利用することです。また、オープンソースの OpenOffice も将来のバージョンで動作するでしょう。

4.2. FreeBSD 用の Motif はどうやったら手に入りますか

FreeBSD 用の廉価版 ELF Motif 2.1.20 (i386 版、Alpha 版) に関する情報はApps2go から手に入れることができます。

この製品には、「開発者版 (development edition)」 と、 より安価な「ランタイム版 (runtime edition)」 の二つの版があります。これらの製品は以下の物が含まれています。

- OSF/Motif manager、xmbind、panner、wsm。
- uil、mrm、xm、xmcxx、インクルードファイルや Imake ファイルといった開発者向けキット
- FreeBSD 3.0 以降で利用できる ELF 版スタティックライブラリ、 およびダイナミックライブラリ
- デモンストレーションプログラム

注文する際には FreeBSD 用の Motif であることをきちんと 確認してください (あなたの欲しいアーキテクチャを指定するのも 忘れないでください!)。NetBSD や OpenBSD 用の Motif もまた、 *Apps2go*から販売されています。現在、FTP による ダウンロードのみ利用可能です。

より詳しい情報は

Apps2go WWW page

問い合わせは

Sales または Support 電子メールアドレス。

もしくは

phone (817) 431 8775 or +1 817 431-8775

他の FreeBSD 用 Motif 2.1 (ELF 版、a.out 版) に関する情報は Metro Link から手に入れることができます。

この製品は以下の物が含まれています。

- OSF/Motif manager、xmbind、panner、wsm。
- uil、mrm、xm、xmcxx、インクルードファイルや Imake ファイルといった開発者向けキット
- スタティックライブラリ、およびダイナミックライブラリ。 (FreeBSD 3.0 以降で利用できる ELF 版か、 FreeBSD 2.2.8 以前で利用できる a.out 版を指定してください)
- デモンストレーションプログラム
- 整形済みのマニュアルページ

注文する際には FreeBSD 用の Motif であることをきちんと 確認してください。Linux 用の Motif も *Metro Link* から販売されています。現在、CDROM および FTP によるダウンロードが利用可能です。

FreeBSD 用の a.out 版 Motif 2.0 に関する情報は Xi Graphics から 手に入れることができます。

この製品には以下の物が含まれています。

- OSF/Motif manager、xmbind、panner、wsm。
- uil、mrm、xm、xmcxx、インクルードファイルや Imake ファイルといった開発者向けキット
- FreeBSD 2.2.8 以前のバージョンで利用できるスタティックライブラリ、およびダイナミックライブラリ
- デモンストレーションプログラム
- 整形済みのマニュアルページ

注文する際には FreeBSD 用の Motif であることをきちんと 確認してください。BSDI や Linux 用の Motif もまた、*Xi Graphics* から販売されています。現在フロッピーディスク 4枚組ですが、 将来的には CDE のように統合された CD に変わるでしょう。

4.3. FreeBSD 用の CDE はどうやったら手に入りますか

以前 Xi Graphics より FreeBSD 用の CDE が 販売されていましたが、現在は既に販売が終了しています。

KDE 多くの点で CDE と類似しているオープンソースの X11 デスクトップ環境です。 xfce の ルック & フィール (訳注: 外観や操作方法のこと) も気に入るかも知れません。 KDE、xfce は、いずれも FreeBSD Ports Collection に含まれています。

4.4. 高機能な商用 X サーバってあるんですか?

はい、Xi Graphics と Metro Link から、FreeBSD ほか Intel ベースのシステムで動作する Accelerated-X

という製品が販売されています。

Metro Link は、FreeBSD のパッケージ操作ツールを利用することで容易に設定が行なえるほか、数多くのビデオボードをサポートした高機能な X サーバを提供しています。配布はバイナリ形式のみで、FTP が利用可能です。もちろん、とても安価 ($39) に手に入れることができます。

また、Metro Link は ELF 版、a.out 版の FreeBSD 用 Motif も販売しています (前を参照)。

より詳しい情報は

Metro Link WWW page

問い合わせは

Sales または Support 電子メールアドレス

もしくは

phone (954) 938-0283 or +1 954 938-0283

Xi Graphics が提供している高性能な X サーバは楽に設定を行なえるほか、数多くのビデオボードをサポートしています。サーバはバイナリのみが含まれます。FreeBSD 用と Linux 用の統合されたフロッピーディスクに入っています。Xi Graphics は Laptop サポートに特化した高性能 X サーバも提供しています。

バージョン 5.0 の「互換デモ」が無料で入手できます。

また Xi Graphics は FreeBSD 用の Motif と CDE も販売しています (前を参照)。

より詳しい情報は

Xi Graphics WWW page

問い合せは

Sales または Support

もしくは

phone (800) 946 7433 or +1 303 298-7478.

4.5. FreeBSD 用のデータベースシステムはありますか?

もちろんです。FreeBSD のウェブサイトにある 商用ベンダー というセクションをご覧ください。

また、FreeBSD Ports Collection のデータベースのセクションも参考になるでしょう。

4.6. Oracle を FreeBSD 上で動かすことはできますか?

はい。Linux 版 Oracle を FreeBSD でセットアップするための方法は、次に示すページに詳しく書かれています。

- http://www.scc.nl/~marcel/howto-oracle.html
- http://www.lf.net/lf/pi/oracle/install-linux-oracle-on-freebsd

Chapter 5. ユーザアプリケーション

5.1. そういうユーザアプリケーションはどこにあるの?

FreeBSDに移植されたソフトウェアパッケージについては、FreeBSD Ports Collection のページをご覧ください。このリストには現在3400を越える項目があり、しかも毎日更新されています。このページをこまめに訪れるか、freebsd-announce メーリングリストを購読すると、新しく入った ports を定期的にチェックすることができます。

大部分の ports は 2.2 と 3.x および 4.x ブランチで利用できるはずです。多くは 2.1.x 系のシステムでも同様に動作するでしょう。FreeBSD のリリースが出る度に、そのリリースの時点での ports ツリーの スナップショットが撮られ、ports/ ディレクトリに 納められることになっています。

また、"package" という考えも採用されています。これは基本的には gzip で圧縮されたバイナリディストリビューションに、
インストール時に環境に合わせた作業が必要になった場合、行う機能を多少付け加えたものです。package を使えば、どのようなファイルが配布物として含まれているか、と言った細かい事柄にいちいち煩わされることなく、
簡単にインストールやアンインストールを繰り返すことができます。

インストールしたい package があるなら、/stand/sysinstallの、「インストール後の FreeBSD の設定を行う」の下にある package のインストールメニューを使うか、package のファイル名を指定して pkg_add(1) を使用してください。package のファイル名には、通常末尾に .tgz がついています。CDROM をご使用の方は、CD の packages/All ディレクトリからそれらのファイルを利用することができます。また、以下の場所から、FreeBSD の各種バージョンにあわせた package をダウンロードする こともできます。

2.2.8-RELEASE/2.2.8-STABLE 用

ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/i386/packages-2.2.8/

3.X-RELEASE/3.X-STABLE 用

ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/i386/packages-3-stable/

4.X-RELEASE/4-STABLE 用

ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/i386/packages-4-stable/

5.X-CURRENT 用

ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/i386/packages-5-current

お近くのミラーサイトもご利用ください。

新しい ports が続々と追加されている状態なので、すべての ports に 対応する package が存在するわけではないことを覚えておいてください。定期的に ftp.FreeBSD.org マスターサイトを訪れて、どのような package が利用できるのかチェックするのも良いでしょう。

5.2. なぜ /bin/sh はこんなに低機能なのですか? どうして bash や他のシェルを採用しないのでしょう?

それは、POSIX がそのようなシェルがあることを規定しているからです。

もっと込み入った回答:
多くのユーザは、多くのシステムで同じように動作できるシェルスクリプトを書く必要があります。
これが、POSIX でシェルやユーティリティコマンドが細く規定されている理由です。
ほとんどすべてのスクリプトは Bourne shell で書かれているのですが、
それは、数多くの重要なプログラミングインタフェイス (make(1)、 system(3)、popen(3)、や Perl や Tcl 等の類似の 高水準スクリプト言語) が、コマンドの解釈に Bourne shell を使うからです。 このように Bourne shell が極めて頻繁にかつ広範囲で使われているため、
素早く起動できて確実に動作し、メモリを少ししか消費しないということが 重要になります。

既存の実装は、 私たちに可能な限りこれらの多くの要求を同時に満足することができる最良のものです。
/bin/sh を小さいままに保つため、 私たちは他のシェルが持つ様々な便利な機能を提供していません。
Ports コレクションが bash や scsh、tcsh、zsh などの 多機能なシェルを含んでいるからです
(これらのシェルすべての メモリ使用状況は、ps -u の "VSZ" や "RSS" の行で、あなた自身が確認することができます)。

5.3. libc.so.3.0 はどこにありますか?

FreeBSD 2.1.x のシステムで 2.2 以降用の package を動かそうとしていますね?
前のセクションを読んで、システムに合った正しい port/package を入手してください。

5.4. Error: can't find libc.so.4.0 というメッセージが表示されるのですが。

何かの手違いで、4.X と 5.X のシステム用 package をダウンロードし、 FreeBSD 2.X、もしくは 3.X のシステムにインストールしてしまったのでしょう。 対応する正しいバージョンの package をダウンロードしてください。

5.5. 386/486SX のマシンで ghostscript を動かすとエラーがでます。

あなたのマシンには数値演算プロセッサが搭載されていませんね?
カーネルにコプロセッサの代わりとなる数値演算エミュレータを追加する必要があります。
以下のオプションをカーネルの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。

```
options GPL_MATH_EMULATE
```



このオプションを追加する場合、 MATH_EMULATE の行を削除してください。

5.6. SCO/iBCS2 のアプリケーションを実行すると、 socksys で落ちてしまいます。 (FreeBSD 3.0 とそれ以前のみ)

まず最初に /etc/sysconfig (または /etc/rc.conf, rc.conf(5) 参照) の最後のセクションを編集し、以下の変数を YES に直します。

```
# Set to YES if you want ibcs2 (SCO) emulation loaded at startup
ibcs2=NO
```

これでシステムの起動時に ibcs2 カーネルモジュールが読み込まるようになります。

次に /compat/ibcs2/dev/ を以下のように編集します。

```
lrwxr-xr-x  1 root  wheel         9 Oct 15 22:20 X0R@ -> /dev/null
lrwxr-xr-x  1 root  wheel         7 Oct 15 22:20 nfsd@ -> socksys
-rw-rw-r--  1 root  wheel         0 Oct 28 12:02 null
lrwxr-xr-x  1 root  wheel         9 Oct 15 22:20 socksys@ -> /dev/null
crw-rw-rw-  1 root  wheel   41,   1 Oct 15 22:14 spx
```

open や close の処理は、socksys から /dev/null (null(4) 参照) へシンボリックリンクを張ることで代用します。 残りの処理は、-CURRENT に入っているコードが担当しています。 これは以前のものより ずっとスッキリした方法です。

5.7. INN (インターネットニュース) の設定方法は?

inn の package や port をインストールしたあとに Dave Barr's INN Page を見てみましょう。初心者向けの INN FAQ があります。

5.8. どのバージョンの Microsoft FrontPage を手に入れる必要がありますか?

ルーク、ports を使うのだ! パッチ処理済みの Apache が ports ツリーから入手できます。

5.9. FreeBSD は Java をサポートしていますか?

はい。 http://www.FreeBSD.org/java/ をご覧ください。 日本語訳 もあります。

5.10. 3.x-STABLE を載せているマシンで port がコンパイルできないことがあります。それはどうしてですか?

もし、その時点の -CURRENT か -STABLE に比べてずっと古いバージョンの FreeBSD を利用しているなら、 http://www.FreeBSD.org/ports/ にある ports アップグレードキットが必要です。

最新の FreeBSD を利用しているのに発生する場合はおそらく、 -CURRENT では正常なのに -STABLE ではうまく動かなくなるような変更がその port に対して行なわれ、受理されてしまっているのでしょう。 ports コレクションは -CURRENT と -STABLE、 両方のブランチで動かなければならないものですので、 もしそれを発見したら send-pr(1) コマンドを使ってバグレポートの提出をお願いします。

5.11. ld.so はどこにありますか?

3.1-R 以降などの Elf 化されたマシンで Netscape Navigator などの aout 形式のアプリケーションを動かすときには、 /usr/libexec/ld.so と aout ライブラリのファイルが必要です。 それらは配布物の `compat22` に納められています。 /stand/sysinstall や compat22 サブディレクトリ内の install.sh を使って `compat22` をインストールしてください。 合わせて 3.1-R と 3.2-R の ERRATA もお読みください。

5.12. ソースコードを更新しました。さて、インストール済みの ports を更新するにはどうすればよいでしょうか?

残念ながら、インストール済みの ports を更新する簡単な 方法はありません。`pkg_version` コマンドを用いて ports ツリー中の新しいバージョンに更新する スクリプトを次のように生成することができます。

```
# pkg_version -c > /tmp/myscript
```

出力されたスクリプトを使う前に、手で 編集しなければなりません。現在のバージョンの `pkg_version` では、スクリプトの先頭に `exit` を挿入して強制しています。

スクリプトの出力には、更新された packages に依存する packages が記載されているので、保存しておきましょう。これらも やはり更新する必要があるかもしれません。通常、更新が 必要となるのは、共有ライブラリのバージョンが変化し、 そのライブラリを利用している ports が新しいライブラリを用いるために 再構築する必要がある場合です。

システムが常時稼動しているならば、 /etc/periodic.conf に `weekly_status_pkg_enable="YES"` を設定して、periodic(8) システムによって毎週更新が必要な ports の一覧を生成できます。

Chapter 6. カーネルコンフィグレーション

6.1. カーネルをカスタマイズしたいんですが、難しいですか?

全然難しくありません。 カーネルの再構築を調べてください。



うまく動作するカーネルができたら、 日付入りのカーネルのスナップショットを kernel.YYMMDD のように作成することをおすすめします。 こうしておけば、次にカーネルの構築をやってうまくいかなくなってしまっても、 kernel.GENERIC にわざわざ戻る必要がなくなります。 これは、GENERIC カーネルでサポートされないデバイスから起動している場合は、 特に重要です。

6.2. _hw_float が無いので、カーネルのコンパイルがうまくいきません。

推測ですが、数値演算コプロセッサを持ってないからと思って、 npx0 (npx(4) 参照) をカーネルコンフィグファイルから削除してしまったのではないでしょうか? npx0 は必須です。 コプロセッサがなくても、npx0 デバイスは削除してはいけません。

6.3. わたしのカーネルはどうしてこんなに大きい (10MB 以上) のでしょうか?

これはデバッグモードでカーネルを構築していることが原因です。
デバッグモードで構築されたカーネルは、 デバッグに用いられる膨大なシンボル情報を含んでいるため、 カーネルのサイズが非常に大きくなります。 ただし FreeBSD 3.0 とそれ以降のシステムの場合は カーネルのサイズは小さくなりますし、
デバッグカーネルを実行する時のパフォーマンスの低下もありません。
また、そのカーネルはシステムがパニックした場合に有用です。

しかし、容量の小さなディスクでシステムを運用していたり、
単にデバッグカーネルを実行したくない場合は、
以下の両方が当てはまっているかどうか確認してください。

- カーネルコンフィグファイルに以下の行が書かれていないこと。

```
makeoptions DEBUG=-g
```

- `config` を実行する際、 `-g` オプションを付けていないこと。

上に書かれた指定は両方ともカーネルをデバッグモードで構築するためのものです。
上の手順を従っている限り、カーネルを普通に構築してサイズの小さなカーネルを得ることができます。
その場合のカーネルサイズは、およそ 1.5MB から 2MB 程度になります。

6.4. マルチポートシリアルのコードで割り込みが衝突しています。

マルチポートシリアルを最初のポートだけ検出され、どうやったらいいでしょうか?

サポートするコードを含んだカーネルをコンパイルしようとすると、残りのポートは割り込みの競合のためスキップされたと言われます。

ここでの問題は、FreeBSD にはハードウェアまたはソフトウェアの競合により、カーネルがクラッシュするのを防ぐコードが含まれているという点です。
解決するには、最初のポートにだけ IRQ の設定を書き、残りは IRQ の設定を削除します。
以下に例を示します。

```
# Multiport high-speed serial line - 16550 UARTS
#
device sio2 at isa? port 0x2a0 tty irq 5 flags 0x501 vector siointr
device sio3 at isa? port 0x2a8 tty flags 0x501 vector siointr
device sio4 at isa? port 0x2b0 tty flags 0x501 vector siointr
device sio5 at isa? port 0x2b8 tty flags 0x501 vector siointr
```

6.5. カーネルを構築にいつも失敗します。 GENERIC カーネルも構築できません。

さまざまな理由が考えられます。以下、順に列記します。

- あなたは新しい `make buildkernel` や `make installkernel` ターゲットを使わず、現在走っているシステムを構築した時と異なるソースツリーを 構築しようとしている (たとえば、4.0-RELEASE のシステム上で 4.3-RELEASE を構築しようとしている) のではないでしょうか? もしシステムをアップグレードしようとしているのなら、 /usr/src/UPDATING ファイルを "共通項目 (COMMON ITEMS)" 節に注意しながら最後までお読みください。
- あなたは新しい `make buildkernel` や `make installkernel` ターゲットを 使っているのにも関わらず、`make buildworld` を行なっていないのではないでしょうか? `make buildkernel` ターゲットは、 `make buildworld` ターゲットによって作られるファイルに依存しています そのため、`make buildkernel` が正常に終了するためには `make buildworld` ターゲットが正常に完了している必要があります。
- 構築しようとしているのが FreeBSD-STABLE だったとしても、あなたが入手したソースツリーが何らかの理由で書き換わったり、壊れてしまっているのかも知れません。 FreeBSD-STABLE はほとんどの場合、きちんと構築できるようになっていますが、確実に構築可能であることが保証されているのはリリース版だけです。一度ソースツリーを再取得して、問題が解決しないかどうか試してみてください。
また、あるサーバから取得した時に問題が発生したら、別のサーバを試すのも効果があるかも知れません。

Chapter 7. システム管理

7.1. システムスタートアップファイルはどこにあるのですか?

FreeBSD 2.0.5R から 2.2.1R までは、 プライマリコンフィグレーションファイルは /etc/sysconfig にあります。 オプションはすべてこのファイルで設定され、他の /etc/rc (rc(8) 参照) および /etc/netstart といった ファイルはこれを読み込むだけです。

ファイル /etc/sysconfig を見て、システムに適合するように変更してください。 このファイルには、それぞれの場所に何を書けばいいのかを表すコメントがたくさん書かれています。

FreeBSD 2.2.2 から 3.0 までのシステムでは、 /etc/sysconfig は、 より分りやすい名前の rc.conf(5) に改名され、それに従って書式もいくぶん改められています。 /etc/netstart も /etc/rc.network に改名され、 全部のファイルを cp /usr/src/etc/rc* /etc で一度にコピーすることが出来るようになります。

FreeBSD 3.1 とそれ以降では、 /etc/rc.conf が /etc/defaults/rc.conf に移動しました。 このファイルを編集してはいけません! 代わりに、 /etc/defaults/rc.conf の中で変えたいエントリの行を /etc/rc.conf にコピーし、 そこで変更するようにしてください。

たとえば named を起動したいとしましょう。 FreeBSD 3.1 かそれ以降のシステムで FreeBSD 付属の DNS サーバを起動するには、次のようにするだけです。

```
# echo named_enable="YES" >>
            /etc/rc.conf
```

FreeBSD 3.1 かそれ以降でローカルサービスを起動するためには、 /usr/local/etc/rc.d ディレクトリにシェルスクリプトを置きます。 シェルスクリプトは起動可能に設定し、ファイル名が .sh で終わっていなければなりません。 FreeBSD 3.0 とそれ以前のリリースでは、 /etc/rc.local を編集する必要があります。

ファイル /etc/rc.serial はシリアルポートの初期化 (たとえばポートの設定を固定したり等々) のためにあります。

ファイル /etc/rc.i386 は iBCS2 エミュレーションのような Intel アーキテクチャ固有の設定や、 PC システムコンソール設定のためにあります。

7.2. 簡単にユーザを追加するにはどうすればいいのですか?

adduser(8) コマンドを使用してください。 また、pw(8) コマンドを用いることで、さらに細かい操作が可能です。

ユーザを削除するには rmuser(8) コマンドを使用してください。 繰り返しになりますが、pw(8) でも構いません。

7.3. 新しいリムーバブルドライブを持っていますが、どうやって使うの?

そのリムーバブルドライブが ZIP であれ EZ drive であれ
(あるいはもしそういう風に使いたいのなら、フロッピーであれ)、
またハードディスクであれ、一旦システムにインストールされて認識され、
カートリッジ、フロッピー等々が挿入されていれば、 ことはどのデバイスでも全く同じように進みます。

(このセクションはMark Mayo's ZIP FAQ に基づいています)

ZIP ドライブやフロッピーで、すでに DOS のファイルシステムで
フォーマットしてある場合、次のコマンドを使うことができます。 これはフロッピーの場合です。

```
# mount -t msdos /dev/fd0c /floppy
```

出荷時の設定の ZIP ディスクではこうです。

```
# mount -t msdos /dev/da2s4 /zip
```

その他のディスクに関しては、fdisk(8) や /stand/sysinstall を使って、
どのようにレイアウトされているか確かめてください。

以降は ZIP ドライブが 3 番目の SCSI ディスクで、 da2 と認識されている場合の例です。

他人と共有しなければならないフロッピーやリムーバブルディスク でなければ、BSD
ファイルシステムを載せてしまうのが良い考えでしょう。
ロングファイル名もサポートされ、パフォーマンスは少なくとも 2
倍は向上しますし、おまけにずっと安定しています。 まず最初に、DOS レベルでのパーティション /
ファイルシステムを無効にしておく必要があります。使用するのは fdisk でも /stand/sysinstall
でも結構です。 複数のオペレーティングシステムを入れることを考慮する
必要がないような容量の小さなドライブの場合は、 次のように FAT パーティションテーブル (スライス)
全体を飛ばして、BSD のパーティション設定を行うだけで良いでしょう。

```
# dd if=/dev/zero of=/dev/rda2 count=2
# disklabel -Brw da2 auto
```

複数の BSD パーティションをつくる場合、 disklabel か /stand/sysinstall を使います。
固定ディスク上にスワップ領域を加える場合、 そういうことをしたいと思うのはもっともですが、 ZIP
のようなリムーバブルドライブの上ではそういう考えは不適切 でしょう。

最後に、新しいファイルシステムをつくります。ディスク全体を使用する ZIP
ドライブの場合は、以下のようにします。

```
# newfs /dev/rda2c
```

次にマウントします。

```
# mount /dev/da2c /zip
```

また、次のような行を /etc/fstab (fstab(5) 参照) に入れておくのも良い考えでしょう。 `mount /zip` と入力するだけでマウントできるようになります。

```
/dev/da2c /zip ffs rw,noauto 0 0
```

7.4. 自分の crontab ファイルを編集した後 root: not found のようなメッセージが延々と表示されるのですが、これはなぜですか?

これは通常、システム crontab (/etc/crontab) を編集し、crontab(1) を使ってインストールした場合に起こります。

```
# crontab /etc/crontab
```

この方法は正しくありません。 システム crontab のフォーマットは crontab(1) が更新する各ユーザの crontab とは異なります (フォーマットの相違点の詳細は crontab(5) で説明されています)。

もしこのような操作をしてしまったなら、 あらたな crontab は誤ったフォーマットの /etc/crontab のコピーになってしまっているからです。 以下のコマンドで削除してください。

```
# crontab -r
```

今度 /etc/crontab を編集する時は、 その変更を cron(8) に伝えるような操作をしてはいけません。 cron(8) は、自動的にその変更を認識するからです。

もしあなたが何かを一日一回、あるいは一週間や一ヶ月に一回だけ実行させたいなら、シェルスクリプトを /usr/local/etc/periodic に追加し、 periodic(8) コマンドにシステムの cron スケジュールから 他の定期的なシステムのタスクとともに実行させたほうが良いかもしれません。

このエラーの実際の原因は、システム crontab にはどのユーザ権限でコマンドを実行するかを指定する余分なフィールドがあることによるものです。 FreeBSD に添付されている標準のシステム crontab には、 すべてのエントリに `root` が書かれています。 この crontab が `root` ユーザの crontab (システム crontab とは 異なります) として使われた場合、cron(8) は `root` を実行するコマンドの最初の単語だと認識しますが、そのようなコマンドは存在しないのです。

7.5. su(1) コマンドを実行して root になろうとすると、 su が you are not in the correct group to su root と警告します。

これは、セキュリティ上の機能です。su コマンドを実行して root (またはスーパーユーザ権限を持つ他のアカウント) になるには、wheel グループに所属していなければなりません。この機能がないと、システムにアカウントがあって root の パスワードを見つけさえすれば、誰でもスーパーユーザ権限でシステムにアクセスできてしまいます。この機能がある場合は、 必ずしもそうはなりません。wheel グループに 所属していなければ、su(1) がパスワードの入力すら 拒否するからです。

誰かが root に su できるように するには、その人を wheel グループに追加してください。

7.6. rc.conf やその他のスタートアップファイルを書き間違えてしまいました。しかもそのためファイルシステムがリードオンリーになってしまっていて 編集ができません。どうすればいいですか?

シェルのパス名を入力するプロンプトが表示されたときに、 単に ENTER を押し、mount / を実行してそルートファイルシステムを再マウントさせます。
また、お気に入りのエディタがあるファイルシステムを マウントするために mount -a -t ufs をする必要があるかも知れません。あなたのお気に入りのエディタが
ネットワークファイルシステム上にある場合は、
ネットワークファイルシステムをマウントする前にネットワークを 手動で設定するか、ed(1) のようなローカルファイルシステムにある エディタを使うかしなければなりません。

vi(1) や emacs(1) の様なフルスクリーンエディタを 使うつもりなら export TERM=cons25 とやってエディタが termcap(5) データベースから正しいデータを読み取れるようにしなければなりません。

これを行ったあとはいつもと同様、 /etc/rc.conf
を編集して間違いを訂正することができるようになります。
カーネル起動メッセージの直後に表示されたエラーメッセージには、
問題の起こったファイル内での行番号を表示されているはずです。

7.7. どのようにしたら DOS の拡張パーティションをマウントできますか?

DOS 拡張パーティションは、 すべての基本パーティションの後に認識されます。 たとえば、2台目のSCSIドライブの拡張パーティションに "E" パーティションがあるとしますと、 これは /dev に「スライス 5 」のスペシャルファイルを作る必要があり、 /dev/da1s5 としてマウントされます。

```
# cd /dev
# ./MAKEDEV da1s5
# mount -t msdos /dev/da1s5 /dos/e
```

7.8. 他のシステムのファイルシステムを FreeBSD でマウントすることはできますか?

Digital UNIX: UFS CDROM は直接 FreeBSD でマウントすることができます。 Digital UNIX やそれ以外のシステムのサポートする UFS のディスクパーティションをマウントすることはもっと複雑なことで、オペレーティングシステムのディスクパーティションの詳細に依存します。

Linux: 2.2 以降は ext2fs パーティションをサポートします。 詳しくは、mount_ext2fs(8) を見てください。

NT: FreeBSD 用の読みだしのみ可能な NTFS ドライバがあります。 詳しくは、Mark Ovens 氏によって書かれたチュートリアル http://ukug.uk.freebsd.org/~mark/ntfs_install.html をご覧ください。

この問題について他の情報があれば、他の人から感謝されるでしょう。

7.9. どのようにしたら FreeBSD を NT ローダーから起動させることができますか?

この手順は 2.2.x と (起動が 3 つのステージに分かれている) 3.x のシステムとで多少異なります。

FreeBSD のネイティブルートパーティションの最初のセクタをファイルにして DOS/NT パーティション上に置くという画期的なアイディアがあります。 ファイル名を c:\bootsect.bsd (c:\bootsect.dos からの発想です) としたとします。 c:\boot.iniファイルを次のように編集します。

```
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(1)\WINDOWS
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Windows NT"
C:\BOOTSECT.BSD="FreeBSD"
C:\="DOS"
```

この手順は、利用しているシステムが 2.2.x であり、DOS、NT、FreeBSD あるいはその他のオペレーティングシステムがすべて、同じディスクのそれぞれの fdisk パーティションにインストールされていることを想定しています。 この例は、DOS と NT を最初の fdisk パーティションにおき、 FreeBSD は 2 番目においたシステムで確認しています。 また、FreeBSD は MBR を使わずに、 ネイティブパーティションから起動するように設定してあります (訳注: FreeBSD のインストールで、ブートマネジャを使わずに標準 MBR を使う場合に相当します)。

(もし NTFS に変換してしまっているなら)DOS フォーマットのフロッピーディスクか FAT パーティションを /mnt に DOS マウントします。

```
# dd if=/dev/rda0a of=/mnt/bootsect.bsd bs=512 count=1
```

再起動して DOS か NT に切替えます。NTFS ユーザは bootsect.bsd や bootsect.lnx

をフロッピーディスクから C:\ へコピーします。 boot.ini のファイル属性 (パーミッション) の変更を以下のように行ないます。

```
> attrib -s -r c:\boot.ini
```

上の例の boot.ini で示したような正しいエントリを加え、 ファイル属性を元に戻します。

```
> attrib +s +r c:\boot.ini
```

FreeBSD が MBR から起動するようになっている場合、 それぞれのネイティブパーティションから起動するように設定した後で、 DOS から `fdisk` コマンドを実行して元に戻してください。

FreeBSD 3.X における手順は、これよりいくぶん簡単です。

FreeBSD が NT 起動パーティションとして同じディスクにインストールされている場合には、 /boot/boot1 を単純に C:\BOOTSECT.BSD へコピーします。 もし FreeBSD が異なったディスクにインストールされている場合には、 /boot/boot1 では動作しませんので、 /boot/boot0 が必要です。



ここで /boot/boot1 の代わりに /boot/boot0 をコピーするようなことをしてはいけません!
そうすると、パーティションテーブルを上書きしてしまい、
コンピュータが起動できなくなってしまいます。

/boot/boot0 をインストールするには、 sysinstall のブートマネージャを利用するかどうか尋ねられる画面で FreeBSD ブートマネージャを選択する必要があります。 /boot/boot0 のパーティションテーブル部分は NULL 文字で埋められているのですが、 sysinstall は /boot/boot0 を MBR にコピーする前にパーティションテーブルをきちんとコピーしてくれるからです。

FreeBSD ブートマネージャは最後に起動した OS を記録するために パーティションテーブルの最後に起動した OS のエントリにあるアクティブフラグをセットし、512 バイト全体を MBR に書き戻します。 これは /boot/boot0 を C:\BOOTSECT.BSD にコピーし、 エントリの一つにアクティブフラグをセットして空のパーティションテーブルを MBR に書き込むことと同じです。

7.10. FreeBSD と Linux を LILO から起動するには?

FreeBSD と Linux が同じディスクにインストールされている場合、 単に Linux 以外の OS を起動するための LILO のインストール手順に 従えばいいだけです。非常に簡単にではありますが、記してみましょう。

Linux を起動し、/etc/lilo.conf に以下の行を加えて ください。

```
other=/dev/hda2
      table=/dev/hda
```

```
        label=FreeBSD
```

(上記の手順は FreeBSD のスライスが Linux から /dev/hda2 という名前で見えていると仮定しています。あなたの設定にあわせてください) その後、`lilo` を `root` で実行すれば完了です。

FreeBSD が別のディスクにインストールされているのなら、 LILO のエントリに `loader=/boot/chain.b` を追加してください。たとえば、このようになります。

```
other=/dev/dab4
        table=/dev/dab
        loader=/boot/chain.b
        label=FreeBSD
```

場合によっては、二つ目のディスクを正しく起動するために FreeBSD ブートローダに BIOS ドライブ番号を指定する必要があるかもしれません。 たとえば、FreeBSD SCSI ディスクが BIOS によって BIOS ディスク 1 として認識されるのなら、 FreeBSD のブートローダのプロンプトで、次のように指定する必要があります。

```
Boot: 1:da(0,a)/kernel
```

FreeBSD 2.2.5 やそれ以降の版では、boot(8) を設定すれば 起動時に上記のことが自動的に行えます。

Linux+FreeBSD mini-HOWTO が FreeBSD と Linux とを相互に使えるようにするためのよい参考資料になるでしょう。

7.11. FreeBSD と Linux を BootEasy から起動するには?

LILO をマスターブートレコード (MBR) ではなく Linux の起動パーティションにインストールしてください。 これで BootEasy から LILO を起動できるようになります。

Windows95 と Linux を使用している場合は、 いずれにせよ後者の方がおすすめです。 Windows95 を再インストールする必要にかられたとき、 Linux を起動可能に戻す手続きが簡単ですむからです (Windows95 は偏屈なオペレーティングシステムで、 マスターブートレコード (MBR) から他のオペレーティングシステムを追い払ってしまうのです)。

7.12. 「危険覚悟の専用 (dangerously dedicated) ディスク」は健康に悪いの?

インストール作業中、 ハードディスクのパーティションを切る際に 2 つの方法を選ぶことができます。
デフォルトの方法では、fdisk のテーブルエントリ (FreeBSD ではスライスと呼ばれる) を使って、
自身のパーティションを使用する FreeBSD のスライスを、
同じマシンの他のオペレーティングシステムと互換性のある形にします。
それに付随して、ブートセレクタをインストールすれば、
ディスク上の使用可能なオペレーティングシステムを切り替えることができます。
もう一つの方法はディスクすべてを FreeBSD で使うというもので、
この場合ほかのオペレーティングシステムとの互換性を考慮しないことになります。

では、なぜこれが 「危険覚悟の」と言われるのでしょう? このモードのディスクが、通常の PC のユーティリティが有効な fdisk テーブルと見なす情報を持っていないからです。 ユーティリティの出来如何によりますが、 そのようなディスクを発見したとき、 警告を出すものもあります。また、もっと悪い場合、 確認も通告もなしに BSD のブートストラップにダメージを与えるものもあるでしょう。
さらには、「危険覚悟の」ディスクレイアウトは多数の BIOS、 AWARD (たとえば HP Netserver や Micronics システム、 他多数で使用されていた) や Symbios/NCR (人気のあるSCSI コントローラ 53C8xx 用) などを混乱させることが分かっています。 これは完全なリストではありません。 他にもまだまだあります。この混乱の兆候は、 起動時にシステムがロックするというだけでなく、 FreeBSD のブートストラップが自分自身を見つけられないために表示する "read error" というメッセージなどにも現れることでしょう。

そもそもいったいなぜこのモードがあるのでしょうか?
これはわずかに数キロバイトのディスク容量を節約するのみであり、
新規インストールで実際に問題を生ずるのです。 「危険覚悟の」モードの起源は新しい FreeBSD インストーラでの、 BIOS から見えるディスクの 「ジオメトリ」の値とディスク自身との整合性という、 もっとも一般的な問題のひとつを回避したいという要求が背景にあります。

「ジオメトリ」は時代遅れの概念ですが、 未だに PC BIOS とディスクへの相互作用の中核をなしています。 FreeBSD のインストーラがスライスを作る時、 ディスク上のスライスを BIOS が見つけられるように、 スライス位置をディスク上に記録します。それが誤っていれば、 起動できなくなってしまうでしょう。

「危険覚悟の」モードはこれを、 問題を単純にすることで回避しようとします。 状況によってはこれでうまくいきます。 しかし次善の策として使われているに過ぎません。 この問題を解決するもっと良い方法はいくらでもあるのです。

では、 インストール時に「危険覚悟の専用」モードが必要になる状況を回避するにはどうすればよいのでしょうか? まず BIOS が報告するディスクのジオメトリの値を覚えておくことからはじめましょう。 "boot:" プロンプトで “-v” を指定するか、ローダで "boot -v" と指定して、 起動時にカーネルにこの値を表示させることができます。 インストーラが起動する直前に、 カーネルがジオメトリ値のリストを表示するでしょう。 パニックを起こさないでください。 インストーラが起動するのを待ち、 逆スクロールでさかのぼって値を確認してください。 普通は BIOS ディスクユニット番号は、 FreeBSD がディスクを検出する順序と同様であり、 最初に IDE、次に SCSI となります。

ディスクをスライシングする際に、 FDISK の画面で表示されるディスクのジオメトリが正しいこと (BIOS の返す値と一致しているか) を確認してください。 万一異なっていたら “g” を押して修正してください。 ディスクにまったくなにもない場合や、 他のシステムから持ってきたディスクの場合はこれを行なう必要があるかもしれません。 これはそのディスクから起動させようとしている場合にのみ、 問題になることに注意してください。 FreeBSD はそのディスクをうまい具合いに他のディスクと区別してくれます。

ディスクのジオメトリについて BIOS と FreeBSD 間で一致させることができたら、この問題はほぼ解決したと思ってよいでしょう。
そしてもはや「危険覚悟の専用」モードは必要ありません。 しかし、まだ起動時に恐怖の "read error" メッセージが出るようであれば、 お祈りを捧げて新しいディスクを買いましょう。 もう失うものは何もありません。

「危険覚悟の専用ディスク」を通常の PC での使用法に戻すには、 原則として 2 つ方法があります。1

つは十分な NULL バイトを MBR に書き込んで、
きたるべきインストーラにディスクはまっさらだと思い込ませる方法です。 たとえば、こんな感じです。

```
# dd if=/dev/zero of=/dev/rda0 count=15
```

また、マニュアルには書かれていない DOS の「機能」

```
> fdisk /mbr
```

は、BSD ブートストラップを追い払ってくれる上に、
新しいマスターブートレコードをインストールしてくれます。

7.13. どのようにしたらスワップ領域を増やせますか?

スワップパーティションのサイズを増やすのが最良の方法ですが、
別のディスクを追加しなくて済むという利点のある方法があります。
経験から得た一般的な方法はメインメモリの 2倍程度のスワップ領域を
とるというものです。しかしごく小さなメインメモリしかない場合は、
それ以上のスワップを構成したいと思うでしょう。また、将来のメモリの
アップグレードに備え、後でスワップの構成を変更する必要がないように
十分なスワップを構成しておくことは良い考えです。

スワップを別のディスク上に追加することは、単純に同じディスク上
にスワップを追加する場合よりも高速に動作するようになります。
例に挙げれば、あるディスク上のソースをコンパイルしているとして、
スワップが別のディスク上に作られていれば、これらが同じディスク上
にある場合よりも断然速いです。SCSI ディスクの場合は特にそうだと言えます。

ディスクが複数ある場合、スワップパーティションを各ディスクに
作るように構成すると、使用中のディスク上にスワップを置いたとしても、
通常の場合は有益です。一般的に、システムにある高速なディスクには
スワップを作るようにすべきでしょう。 FreeBSD はデフォルトでインターリーブなスワップデバイスを
4つまで サポートします。複数のスワップパーティションを構成する際に、
普通はそれらを大体同じくらいの大きさにして作りたいところですが、
カーネルのコアダンプを取るのに都合が良いようにメインの
スワップパーティションを大きめにとる人もいます。
メインのスワップパーティションはカーネルのコアがとれるように
最低でも実メモリと同じ大きさにすべきでしょう。

IDE ドライブは同時に同じチャネル上の複数のドライブには アクセスできません (FreeBSD は mode 4 をサポートしていないので、 すべての IDE ディスク I/O は "programmed" です)。 IDE の場合であってもやはり、スワップを別のハードディスク上に 作成することをおすすめします。
ドライブは実に安いものです、心配するだけ無駄です。

NFS 越しにスワッピングさせる方法は、 スワップ用のローカルディスクが無い場合にのみ推奨されます。
NFS 越しのスワッピングは遅く、FreeBSD 4.x より前のリリースでは 効率が悪いのですが、4.0 以降ではそれなりに高速になります。
そうはいっても、利用できるネットワークの太さに制限されますし、 NFS

サーバに余計な負荷がかかります。

これは 64MBの vn-swap を作る例です (ここでは /usr/swap0 としますが、もちろん好きな名前を使うことができます)。

カーネルが次の行を含むコンフィグファイルから構成されているかを 確認します。GENERIC カーネルには、この行が含まれています。

```
pseudo-device   vn 1   #Vnode driver (turns a file into a device)
```

1. vn デバイスを作ります

   ```
   # cd /dev
   # sh ./MAKEDEV vn0
   ```

2. スワップファイルを作ります (/usr/swap0)

   ```
   # dd if=/dev/zero of=/usr/swap0 bs=1024k count=64
   ```

3. スワップファイルに適切なパーミッションを設定します

   ```
   # chmod 0600 /usr/swap0
   ```

4. /etc/rc.conf でスワップファイルを有効化させます

   ```
   swapfile="/usr/swap0"   # Set to name of swapfile if aux swapfile desired.
   ```

5. マシンを再起動します

スワップファイルをすぐに有効化させたいのなら以下のようにタイプします。

```
# vnconfig -e /dev/vn0b /usr/swap0 swap
```

7.14. プリンタのセットアップで問題があります

ハンドブックのプリンタの部分を参照してください。
探している問題のほとんどが書かれているはずです。 FreeBSD
ハンドブックの「プリンタの利用」をご覧ください。

プリンタによっては、印刷するのにホスト側にドライバが 必要です。これら "WinPrinters" と呼ばれるものは、 素の FreeBSD では使えません。DOS や Windows NT 4.0 で動作しないなら、そのプリンタはおそらく WinPrinter でしょう。 ただし、唯一の希望が残されています。ports/print/pnm2ppa の port が 対応しているかどうか確認してみてください。パッケージの説明にはこう書いてあります。

このソフトウェアは PPA (printer performance architecture) プロトコルの出力を行います。このプロトコル は HP の "Windows 専用" プリンタの一部に使われています。 そのなかには、HP Deskjet 820C シリーズ、HP DeskJet 720 シリーズ、および HP DeskJet 1000 シリーズがあります。(略)

WWW: http://pnm2ppa.sourceforge.net/

7.15. 私のシステムのキーボードマッピングは間違っています。

`kbdcontrol` プログラムは、 キーボードマップファイルを読み込むためのオプションを備えています。 /usr/shared/syscons/keymaps の下にたくさんのマップファイルがあります。 システムに関連のあるものを一つ選んで、ロードしてください。

```
# kbdcontrol -l uk.iso
```

/usr/shared/syscons/keymaps と拡張子 .kbd は、どちらも kbdcontrol(1) によって使用されます。

これは /etc/sysconfig (または rc.conf(5)) 中で設定することができます。 このファイル中にあるそれぞれのコメントを参照してください。

FreeBSD 2.0.5R やそれ以降の版では、 テキストフォントやキーボードマッピングに関係のあるものはすべて、 /usr/shared/examples/syscons の中におさめられています。

現在以下のマッピングがサポートされています。

- Belgian ISO-8859-1
- Brazilian 275 keyboard Codepage 850
- Brazilian 275 keyboard ISO-8859-1
- Danish Codepage 865
- Danish ISO-8859-1
- French ISO-8859-1
- German Codepage 850
- German ISO-8859-1
- Italian ISO-8859-1
- Japanese 106
- Japanese 106x
- Latin American
- Norwegian ISO-8859-1
- Polish ISO-8859-2 (programmer's)
- Russian Codepage 866 (alternative)
- Russian utf-8 (shift)

- Russian utf-8
- Spanish ISO-8859-1
- Swedish Codepage 850
- Swedish ISO-8859-1
- Swiss-German ISO-8859-1
- United Kingdom Codepage 850
- United Kingdom ISO-8859-1
- United States of America ISO-8859-1
- United States of America dvorak
- United States of America dvorakx

7.16. 起動時に、**unknown: <PNP0303> can't assign resources** というメッセージが表示されるのですが?

以下は、freebsd-current メーリングリストへの投稿からの 抜粋です。

Garrett Wollman <wollman@FreeBSD.org>, 2001 年 4 月 24 日 "can't assign resources" というメッセージは、 そのデバイスがレガシー ISA デバイスで、PnP を意識していないドライバがカーネルに組み込まれていることを示します。
これには、キーボードコントローラ、プログラム可能な 割り込み制御 IC やその他さまざまな標準的なデバイスが
あります。リソースが割り当てられないのは、既にそのアドレスを 使っているドライバがあるからです。

7.17. ユーザディスククォータが正常に動作していないようです。

1. "/" にはディスククォータを設定しないでください。
2. クォータファイルが置かれるファイルシステム上に クォータファイルを置くようにしてください。

Filesystem	Quota file
/usr	/usr/admin/quotas
/home	/home/admin/quotas
...	...

7.18. わたしの **ccd** は、 何が適合していない **(Inappropriate)** のでしょう?

次のような症状が現れます。

```
# ccdconfig -C
```

```
ccdconfig: ioctl (CCDIOCSET): /dev/ccd0c: Inappropriate file type or format
```

通常この現象はタイプを「未使用 (unused)」のまま放っておかれた c パーティションをつなげようとした場合に現れます。ccd ドライバは FS_BSDFFS タイプをベースとするパーティションを要求します。
つなげようとしているディスクのディスクラベルを編集して、 パーティションのタイプを 4.2BSD に変更してください。

7.19. どうしてわたしの ccd のディスクラベルを変更することができないのでしょう?

次のような症状が現れます。

```
# disklabel ccd0
(it prints something sensible here, so let's try to edit it)
# disklabel -e ccd0
(edit, save, quit)
disklabel: ioctl DIOCWDINFO: No disk label on disk;
use "disklabel -r" to install initial label
```

これは ccd から返されるディスクラベルが、 実はディスク上にはないまったくの偽の情報だからです。これを明示的に書き直すことで問題を解消できます、 それには、つぎのようにします。

```
# disklabel ccd0 > /tmp/disklabel.tmp
# disklabel -Rr ccd0 /tmp/disklabel.tmp
# disklabel -e ccd0
(this will work now)
```

7.20. FreeBSD は System V の IPC プリミティブをサポートしますか?

はい。 FreeBSD は System-V スタイルの IPC をサポートします。共有メモリ、メッセージ、セマフォが含まれます。 以下の行をカーネルコンフィグファイルに加えると、サポートが有効になります。

```
options    SYSVSHM         # enable shared memory
options    SYSVSEM         # enable for semaphores
options    SYSVMSG         # enable for messaging
```



FreeBSD 3.2 とそれ以降では、 これらのオプションがあらかじめ *GENERIC* カーネルに含まれていますので、 あなたのシステムにはすでに組み込まれています。

カーネルを再構築してインストールしてください。

7.21. UUCP でメールを配送するには sendmail をどう使えばよいのですか?

FreeBSD に付属している sendmail は、インターネットに直接つながっているサイトにあわせて設定してあります。 UUCP 経由で mail を交換したい場合には sendmail の設定ファイルを改めてインストールしなければなりません。

/etc/sendmail.cf を自分の手で改造するのは純粋主義者のやるような事です。 sendmail の version 8 は m4(1) のようなプリプロセッサを通して設定ファイルを生成する新しいアプローチを取っており、より抽象化されたレベルの設定ファイルを編集します。 /usr/src/usr.sbin/sendmail/cf ディレクトリの中にある設定ファイルを使用してください。

もしすべてのソースをインストールしていない場合には sendmail の設定ツールは、別の tar ファイルにまとめてあります。CD-ROM が mount されている場合には、次のようにしてください。

```
# cd /cdrom/src
# cat scontrib.?? | tar xzf - -C /usr/src contrib/sendmail
```

これはたった数 100Kbyte ですから心配ないでしょう。 cf ディレクトリにある README に、m4 での設定の基本的な説明があります。

UUCP での配送のためには、`mailertable` を使用すれば よいでしょう。これによって、sendmail が配送方式を決定するデータベースを 作成することができます。

まずはじめに、 .mc ファイルを作成しなければなりません。 /usr/src/usr.sbin/sendmail/cf/cf というディレクトリが、 これらのファイルを作成する場所です。既にいくつか例があると思います。これから作成するファイルの名前を foo.mc とすると、 sendmail.cf を求めているような形式に変換するには、 次のようにしてください。

```
# cd /usr/src/usr.sbin/sendmail/cf/cf
# make foo.cf
# cp foo.cf /etc/sendmail.cf
```

標準的な .mc ファイルは次のようになります。

```
include(`../m4/cf.m4')
VERSIONID(`Your version number')
OSTYPE(bsd4.4)

FEATURE(nodns)
FEATURE(nocanonify)
FEATURE(mailertable)

define(`UUCP_RELAY', your.uucp.relay)
define(`UUCP_MAX_SIZE', 200000)

MAILER(local)
```

```
MAILER(smtp)
MAILER(uucp)

Cw    your.alias.host.name
Cw    youruucpnodename.UUCP
```

nodns と nocanonify という指定をすることで、 mail の配送に DNS を使用しなくなります。 UUCP_RELAY という 行に関しては、 ある理由から必要ですがそれは聞かないでください。 .UUCP で終わる仮想ドメインを処理することのできるインターネット上での
ホスト名をここに書いてください。通常は、ISP の mail リレーホストを 書くことになると思います。

これが終了したら、次に /etc/mailertable というファイルが必要です。標準的な例は次のとおりです。

```
#
# makemap hash /etc/mailertable.db < /etc/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:
```

見れば分かるように、これは実在する設定のファイルです。はじめの 3
行はドメイン名で指定されたメールが default の経路で配送されずに、 「近道」するために UUCP で隣りのサイトに送るための特別な状況を 処理するものです。 次の行は Ethernet でつながっているローカルのドメインに対しては SMTP で送るための設定です。 最後に、UUCP での隣りのサイトが .UUCP で終わる仮想ドメインの書式で 指定されており、default の rule を uucp-neighbour! recipient で上書きするためのものです。一番最後の行はいつもドットを一つ書きます。
これは、ここまでの行でマッチしなかったすべてのホストにマッチし、
このサイトから世界に向けて出ていくための mail gateway に UUCP で配送するためのものです。 uucp-dom: に続けて書かれているノード名は、 uuname コマンドで指定することによって UUCP で直接配送される正しいノード名でなければなりません。

最後に、このファイルは使用する前に DBM データベースのファイルに
変換する必要があります。これを行なうコマンドラインは mailertable
の最初のコメントに書いてあります。mailertable を変更した時には、
必ずこのコマンドを実行してください。

最後のヒントです: もし特定のメール配送がうまく作動するかどうか 確かめたい場合には、sendmail の -bt オプションを 使用してください。このオプションによって sendmail は
アドレステストモードで起動します。 0
の後に配送したいアドレスを書いてください。最後の行に、実際に使用される mail agent、この mail agent で送られる送信先のホスト、そして (多分変換されている)
アドレスが表示されます。このモードを抜けるには Control-D を押してください。

```
% sendmail -bt
```

```
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 0 foo@interface-business.de
rewrite: ruleset  0   input: foo @ interface-business . de
...
rewrite: ruleset  0 returns: $# uucp-dom $@ if-bus $: foo \
< @ interface-business . de >
> ^D
```

7.22. ダイアルアップでインターネットに接続する環境でメールをセットアップするにはどうやるの?

静的に IP アドレスが割り当てられる場合は、デフォルトの状態を変更する必要はありません。割り当てられた名前をホストネームと するだけで、sendmail が後のことを引き受けてくれます。

ダイアルアップ ppp を インターネット接続に使用し、動的に IP アドレスが割り当てられる場合は、インターネットサービスプロバイダ (ISP) のメールサーバにメールボックスがあるはずです。 ISP のドメインが myISP.com で、あなたのユーザ名が user だと仮定します。 また、あなたが自分のマシンを bsd.home と呼んでおり、ISP が relay.myISP.com をメールリレーとして使用できると言っているとしましょう。

メールボックスからメールを取ってくるためには、回収 (retrieval) エージェントをインストールする必要があります。 Fetchmail は多種多様なプロトコルをサポートしているのでお勧めです。 ISP が使用しているのは、大抵 POP3 プロトコルです。 ユーザ ppp を使用している場合、 /etc/ppp/ppp.linkup に以下のように記述すると、インターネットと接続が完了した時点で自動的にメールを取得するようになります。

```
MYADDR:
      !bg su user -c fetchmail
```

ローカルでないアカウントにメールを配送するのに sendmail を使用している場合 (後述)、上に示したエントリの後に

```
      !bg su user -c "sendmail -q"
```

を記述します。これはネットワーク接続が確立したらすぐに sendmail に溜っている mailqueue を強制的に処理させるようにします。

この例では、user が bsd.home にアカウントを持ち、bsd.home 上の user のホームディレクトリに、以下のような .fetchmailrc ファイルがつくられていることを想定しています。

```
poll myISP.com protocol pop3 fetchall pass MySecret;
```

言うまでもなく、このファイルは user 以外のユーザが読むことが出来ないようにしなくてはなりません。 内容にパスワード MySecret

が含まれているからです。

正しい `from:` ヘッダをつけてメールを送るためには、 sendmail に `user@bsd.home` ではなく `user@myISP.com` を使用するよう教える必要があります。メールをより早く転送するために、すべてのメールを `relay.myISP.com` へ送るように sendmail に指示しておくのも良いでしょう。

上の要件を満たすには、以下のような .mc ファイルが適しています。

```
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(`myISP.com')dnl
FEATURE(allmasquerade)dnl
FEATURE(masquerade_envelope)dnl
FEATURE(nocanonify)dnl
FEATURE(nodns)dnl
define(`SMART_HOST', `relay.myISP.com')
Dmbsd.home
define(`confDOMAIN_NAME',`bsd.home')dnl
define(`confDELIVERY_MODE', `deferred')dnl
```

.mc ファイルから sendmail.cf への変換方法については、 前のセクションを参照してください. sendmail.cf を更新した後に sendmail をリスタートするのもお忘れなく。

7.23. この UID が 0 の toor という アカウントとは何ですか? 危険にさらされているのでしょうか?

心配無用です。`toor` は "代替の" スーパーユーザーアカウントです (toor は root を逆に綴ったものです)。以前は、bash(1) シェルがインストールされた時に作成されていましたが、現在は標準で作成されています。 このユーザーが作成されるのは、スーパーユーザが非標準のシェルを使う場合を想定しており、 `root` の標準のシェルを変更しなくてもよくなっています。 基本配布に含まれていないシェル (たとえば ports や packages からインストールされるシェル) は、デフォルトでは別のファイルシステムに存在する可能性のある /usr/local/bin に インストールされることが多いので、これは重要です。 `root` のシェルが /usr/local/bin にあり、 /usr (または、/usr/local/bin があるいずれかのファイルシステム) が何らかの理由でマウントされていないとすると、 `root` は問題を解決するためにログインすることができません (シングルユーザーモードで再起動すれば、シェルのパスの入力を促されるのですが)。

`toor` を日々の root の仕事を 非標準のシェルで行うために使い、`root` はシングルユーザーモードや緊急時のために、標準のシェルのままにしている人がいます。何もしなければ、パスワードを無効にしてあるので `toor` ではログインできません。 使いたいなら、`root` でログインして `toor` の パスワードを設定しましょう。

7.24. しまった! root のパスワードを忘れてしまった!

慌てないでください! 単にシステムを再起動し、 シングルユーザモードに移るために Boot: と表示されるプロンプトで boot -s と入力してください (FreeBSD の 3.2 より前のリリースでは -sとなります)。 どのシェルを使うのかという質問には、ENTER キーを押してください。# に移ることができるでしょう。 mount -u / と入力してルートファイルシステムの読み書きを再マウントし、 mount -a と入力して、すべてのファイルシステムをマウントし直した後、 passwd root と入力して root のパスワードを設定し直してください。 その後、exit と入力すれば、起動が続けられます。

7.25. Control-Alt-Delete でシステムが再起動しないようにするにはどうすればいい?

FreeBSD 2.2.7-RELEASE 以降で syscons (デフォルトのコンソールドライバ) を使用している場合には、次の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、インストールしてください。

```
options SC_DISABLE_REBOOT
```

FreeBSD 2.2.5-RELEASE 以降で PCVT コンソールドライバを使用している場合には、同様に次の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、インストールしてください。

```
options PCVT_CTRL_ALT_DEL
```

上にあげたものよりも古い FreeBSD の場合、 現在コンソールが使用しているキーマップを編集し、キーワード boot を nop に書き換えてください。 /usr/shared/syscons/keymaps/us.iso.kbd にあります。その変更を反映させようとして、 このキーマップのロードを明示的に行なうために、 /etc/rc.conf を実行すべきかもしれません。 もちろん他の国のキーマップを使っているのであれば、代わりにそのキーマップファイルを編集してください。

7.26. DOS のテキストファイルを UNIX のテキストファイルに整形するにはどうすればいい?

単に次の perl コマンドを実行してください。

```
% perl -i.bak -npe 's/\r\n/\n/g' file ...
```

file の部分には処理するファイルを指定してください。 整形後のファイルは元のファイル名で作成され、整形前のファイルはバックアップとして元の ファイル名の末尾に拡張子 .bak のつけられた名前で作成されます。

あるいは tr(1) コマンドを使うこともできます。

```
% tr -d '\r' < dos-text-file > unix-file
```

dos-text-file は DOS 形式のテストファイル、 *unix-file* には変換された出力が格納されます。 perl を使うよりほんのちょっぴり速くなります。

7.27. 名前で指定してプロセスにシグナルを送るにはどうすればいい?

killall(1) を使ってください。

7.28. su が not in root's ACL と言って私を悩ませるのはなぜ?

Kerberos の認証システムからくるエラーです。 この問題は致命的なものではなく、うっとおしいといったものです。 su に -K オプションをつけて起動するか、次の質問で説明されている方法で Kerberos をアンインストールしてください。

7.29. Kerberos をアンインストールするにはどうすればいいの?

システムから Kerberos を削除するには、 あなたの動かしているリリースの bin ディストリビューションを再インストールしてください。 もし CDROM を持っているのなら、 その CDROM をマウント (マウントポイントは /cdrom と仮定) して、 次のように入力してください。

```
# cd /cdrom/bin
# ./install.sh
```

7.30. 疑似ターミナルを追加するには?

telnet、ssh、X、screen をたくさん利用されている場合、疑似ターミナルが足りなくなっている可能性があります。 これを増やすには次のようにします。

1. 次の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

   ```
   pseudo-device pty 256
   ```

   新たにカーネルを作りインストールします。

2. 次のコマンドを実行して

```
# cd /dev
# ./MAKEDEV pty{1,2,3,4,5,6,7}
```

新たなターミナル用の 256 個のデバイスノードを作ります。

3. /etc/ttys を編集し 256 個のターミナルごとの定義を追加します。
既存のエントリーの形式にあわせる必要があるでしょう。 たとえばこんな感じです。

```
ttyqc none network
```

正規表現を使った指定は `tty[pqrsPQRS][0-9a-v]` となります。

4. 新しいカーネルでシステムを再起動すると完了です。

7.31. snd0 デバイスを作成することができません!

snd というデバイスは存在しません。 この名前は、FreeBSD サウンドドライバによって作成されるさまざまなデバイス、 mixer や sequencer、 dsp などを総称したものです。

これらのデバイスを作成するには、次のようにする必要があります。

```
# cd /dev
# sh MAKEDEV snd0
```

7.32. 再起動せずにもう一度 /etc/rc.conf を読み込んで /etc/rc を開始させるには?

シングルユーザモードに移行して、 マルチユーザモードに戻ってください。

コンソールで次のように実行します。

```
# shutdown now(注: -r や -h は付けません)
# return
# exit
```

7.33. 砂場 (sandbox) とは何ですか?

"砂場 (Sandbox)" とはセキュリティ用語の一つで、 次の二つの意味があります。

- 一つ目は、「仮想的な『防壁』で囲まれているプロセス」です。
その『防壁』は、そのプロセスに侵入した第三者が、
さらにシステムの広い範囲に影響を与えることを防ぐように設計されます。

このプロセスの振舞いは、『防壁』の中だけに制限される、と表現できます。
つまり、このプロセスにおいて、『防壁』を越えるようなコードの実行は
できないという意味です。そのため、コードの実行におけるセキュリティは
確かなものであると保証でき、実行の詳細な追跡を行なう必要はなくなります。

その『防壁』とは、たとえばユーザ ID がそれにあたるでしょう。 この定義は、security(7) や named(8) のマニュアルページで用いられています。

`ntalk` サービス (/etc/inetd.conf 参照のこと) を例にとってみます。 このサービスはかつて、実行時のユーザ ID として root を用いていましたが、現在では tty というユーザ ID で動作します。 ユーザ tty は、 ntalk を経由してシステムの侵入に成功した第三者が そのユーザ ID 以上の権限を得ることを、より一層困難にするために設計された砂場 (sandbox) なのです。

- 二つ目は「シミュレートされたマシンの内側で実行されるプロセス」のことで、
こちらはより中核的です。 普通に考えれば、あるプロセスに侵入することができる第三者は、
マシンのより広い範囲にも侵入できると信じるものなのですが、
この種のプロセスの場合、それは実際にはシミュレートされたマシンに
侵入しただけなので、現実のデータを変更することは何一つできません。

  これを実現するための最も広く用いられている方法は、
シミュレートされた環境をサブディレクトリに構築し、 そのディレクトリに chroot して、そのディレクトリで プロセスを実行すること (つまり、そのプロセスにとって / はシステムの実際のルートディレクトリ / ではなく、 chroot されたサブディレクトリを指す) です。

  広く用いられているもう一つの方法があります。 それは、既に存在しているファイルシステムを読み込み専用 (read-only) でマウントし、その上に、あるプロセスに対してそのファイルシステムが書き込み可能であるように見せるような、
もう一つのファイルシステムの層を用意するものです。すると、
そのプロセスはファイルを書き込むことができると認識し、
実際に書き込むことができるのもその特定のプロセスだけ -
システムにある他のプロセスは書き込めないのに対して -
であるという状況を実現することができます。

  この種の砂場 (sandbox) は、 その非常に透過的な性質を使って、ユーザ (もしくは侵入者) がその事実に気付かないように実現されます。

UNIX は、内部的に二つの砂場 (sandbox) を実装しています。
一つはプロセスレベルのもの、もう一つはユーザ ID レベルのものです。

UNIX プロセスはすべて、他の UNIX プロセスから完全に隔離されています。
どのプロセスも、他のプロセスのアドレス空間を変更することはできません。
これは、あるプロセスが他のプロセスのアドレス空間を上書きできるような、
クラッシュにつながる行為が容易に実現できる Windows とは全く異なるものです。

UNIX プロセスは、特定のユーザ ID が所有します。 もし、実行者のユーザ ID が `root` ユーザのものでなければ、 ユーザ ID は、他のユーザが所有するプロセスからそのプロセスを守る機能を果たすわけです。 また、そのユーザ ID は、ディスク上にあるデータを保護するのにも使われています。

7.34. セキュアレベル (securelevel) って何ですか?

セキュアレベルとはカーネルに実装されているセキュリティ機構の一つです。
簡単に言うと、カーネルはセキュアレベルが正の値の時に、
ある特定の操作を制限します。この制限は、たとえスーパユーザ (root のこと)
であっても例外ではありません。この文を書いている時点では、
セキュアレベル機構を使って以下のような操作を制限することができます。

- schg (system immutable flag) のようなファイルフラグの変更
- /dev/mem および /dev/kmem 経由でのカーネルメモリへの書き込み
- カーネルモジュールのロード
- ipfirewall(4) ルールの変更

稼働中のシステムでセキュアレベルの状態をチェックするには、 次のコマンドを実行します。

```
# sysctl kern.securelevel
```

出力には、sysctl(8) 変数 (今の場合は kern.securelevel) と数字が現れます。
数字が現在のセキュアレベルの値です。これがもし正の値なら、
何らかのセキュアレベルによる制限が有効になっています。

システム稼働中にセキュアレベルを下げることはできません。
これは、それを可能にするとセキュアレベルの意味がなくなってしまうからです。
セキュアレベルが正の値でないことを要求する操作 (たとえば installworld や日付の変更など)
を行なう必要がある場合は、/etc/rc.conf にあるセキュアレベルの設定 (kern_securelevel と
kern_securelevel_enable という変数) を変更して再起動する必要があります。

セキュアレベルに関する詳しい情報や、各レベルで実現される機能に関しては init(8)
のマニュアルページを参照してください。



セキュアレベルは万能というわけではなく、
弱点も数多く存在します。また、場合によっては、
セキュリティを低下させてしまうこともあります。

最も大きな問題の一つに、セキュアレベルの機能を有効にするには、
起動処理でセキュアレベルが設定されるまでに使われるすべてのファイルを
保護する必要があるということがあります。
もし攻撃者が、システムがセキュアレベルを設定する前にコードを実行することができ
るとしたら、セキュアレベルによる保護は無意味になってしまいます
(起動時には低いセキュアレベルでしか実行できない処理を行なう必要があるため、
セキュアレベルの設定は、起動処理の最後の方で行なわれます)。
起動処理で使われるすべてのファイルを保護することは技術的に不可能です。
もしそうできたとしても、システムの保守はまさに悪夢となるでしょう。
設定ファイル一つ書き換えるのにも、
シングルユーザモードに切替えなければならなくなるのですから。

以上で説明した内容やその他の点については、
メーリングリストでも良く話題にのぼります。議論のようすを

このページから検索してみてください。 セキュアレベルは、
いずれより粒度の細かい機構にとって代わるだろうと考えている人々もいますが、
その点についてはまだ不透明なままです。

どうか注意するようにしてください。

7.35. フロッピーや CDROM や他のリムーバブルメディアのマウントを一般ユーザーに許可するには?

一般ユーザーでもデバイスをマウントできるようにすることができます。 手順は次のとおりです。

1. root になって、 sysctl 変数である vfs.usermount を 1 に設定します。

   ```
   # sysctl -w vfs.usermount=1
   ```

2. root になって、
   リムーバブルメディアに関連するブロックデバイスに適切なパーミッションを設定します。

   例として、最初のフロッピーデバイスをユーザーがマウントできるようにするには、
   次のようにします。

   ```
   # chmod 666 /dev/fd0
   ```

   operator グループに所属するユーザが CDROM ドライブをマウントできるようにするには
   以下のようにします。

   ```
   # chgrp operator /dev/cd0c
   # chmod 640 /dev/cd0c
   ```

3. 最後に vfs.usermount=1 という行を /etc/sysctl.conf ファイルに追加し、
   ブート時にセットされるようにしておきます。

これで、すべてのユーザは フロッピー /dev/fd0 を
自身の所有するディレクトリへマウントすることができます。

```
% mkdir ~/my-mount-point
% mount -t msdos /dev/fd0 ~/my-mount-point
```

これで、operator グループに所属するユーザは CDROM /dev/cd0c を
自身の所有するディレクトリへマウントすることができます。

```
% mkdir ~/my-mount-point
```

```
% mount -t msdos /dev/cd0c ~/my-mount-point
```

デバイスのアンマウントは簡単です。

```
% umount ~/my-mount-point
```

しかし、 vfs.usermount を有効にすることは、セキュリティ上よいことではありません。 MSDOS 形式のメディアにアクセスには、Ports コレクションにある パッケージ mtools を使用した方がよいでしょう。

7.36. システムを新しい巨大ディスクへ移すにはどうするのですか?

一番良いのは新しいディスクに OS を再インストールして、それからユーザデータを移すことです。特にあなたが -stable を複数のリリースを跨いで追い掛けている場合にはこの方法をおすすめします。 あなたは boot0cfg(8) を使うことで booteasy を両方の ディスクにインストールでき、新しい配置で満足している間デュアルブートができます。これを行ったあとデータを移す方法を探すなら次の段落は読み飛ばしてください。

何もないディスクへインストールしないことに決めたならば /stand/sysinstall、なり fdisk(8) と disklabel(8) なりを使って新しいディスクにパーティションとディスクラベルを作らなければなりません。 また boot0cfg(8) で booteasy を両方のディスクに インストールして、コピーの作業が終わったあとに古いシステムからでも新しいディスクからでも起動できるようにしておく必要があります。この作業の詳細は formatting-media tutorial を見てください。

新しいディスクの立ち上げが終わってデータの移動を待つばかりになりました。しかし悲しいかな、無闇やたらとコピーすればいいというものではありません。デバイスファイル (/dev) やシンボリックリンクなどは失敗の元になります。これらを理解するツール、すなわち dump(8) や tar(1) 等を使う必要があります。データの移転はシングルユーザで行うことをお勧めしますが、 絶対と言うわけではありません。

あなたは dump(8) と restore(8) 以外のもので root ファイルシステムを移行してはなりません。 tar(1) コマンドでもたぶんうまく行くでしょうが、 やらないほうがいいでしょう。パーティション一つをもう一つのからのパーティションに移すときは dump(8) と restore(8) 使うべきです。パーティションのデータを新しいパーティションに移すのに dump を使うやり方は以下の通りです。

1. 新しいパーティションに newfs をかける。
2. それを暫定的なマウントポイントにマウントする。
3. そのディレクトリに cd。
4. 古いパーティションを dump し、 その出力をパイプで新しい方へ。

たとえば root を /dev/ad1s1a へ、暫定的なマウントポイントを /mnt として移そうとすると以下のようになります。

```
# newfs /dev/ad1s1a
# mount /dev/ad1s1a
# cd /mnt
# dump 0uaf - / | restore xf -
```

もしパーティションの構成を変えようと思っているなら -
つまり一つだったものを二つにしたり二つだったものをくっつけたり
しようとしているなら、自前であるディレクトリ以下のすべてを
新しい場所へ移す必要が出てくるかも知れません。dump(8) は
ファイルシステムに働くのでこの目的には使えません。この場合は tar(1) を使います。一般に /old から /new への移動は tar(1) で 以下のようにします。

```
# (cd /old; tar cf - .) | (cd /new; tar xpf -)
```

/old に他のファイルシステムが マウントされていて、そのデータの移動までは考えてないならば 最初の tar(1) に 'l' フラグを追加します。

```
# (cd /old; tar clf - .) | (cd /new; tar xpf -).
```

tar(1) のかわりに cpio(1) や pax(1), cpdup (ports/sysutils/cpdup) 等を 使っても構いません。

7.37. システムを最新の -STABLE にアップデートしようとしたのですが -RC や -BETA になってしまいました! 何が起こったのですか?

短い答え: ただの名前です。RC は "リリース候補 (Release Candidate)" に由来するもので、リリースが間近であることを意味します。 また、FreeBSD における -BETA は通常、リリース前のコードフリーズ期間に入っているという意味になります。

長い答え: FreeBSD はそのリリースを 2 ヶ所あるうちの 一方から派生させます。3.0-RELEASE や 4.0-RELEASE の様な (0 のマイナー番号を持つ) メジャーリリースは、一般に -CURRENT と呼ばれる開発版の流れから分岐させられてできます。3.1-RELEASE や 4.2-RELEASE などのマイナーリリースはアクティブな -STABLE ブランチ (枝) の スナップショットでした。 4.3-RELEASE からは、リリース毎にブランチが作成されるように なりました。ものすごく保守的な開発速度 (主にセキュリティ 勧告のみ) を求めている人は、このブランチを追跡すると よいでしょう。

リリースを作る時になるとそれを分岐させるブランチは
特定のプロセスへ突入します。そのプロセスの一つは コードフリーズ (コードの凍結) です。コードフリーズが 始まると、そのブランチの名前がリリースになろうとしていることを反映するものに変えられます。たとえば、4.0-STABLE と 呼ばれていたブランチは名前が 4.1-BETA へと変えられ、コードフリーズとリリース前のテストが 始まったことを示します。バグの修正はリリースの一部としてコミットされます。 ソースコードがリリースの形を取ったなら名前が 4.1-RC へと 変えられ、それからリリースが作られることを示します。 ひとたび RC のステージになってしまうと、発見された もっとも致命的なバグの修正しかできなくなります。

ひとたびリリースが (この例では 4.1-RELEASE) 作られれば、 そのブランチは 4.1-STABLE と改名されます。

7.38. 新しいカーネルを入れようとしたのですが、 chflags に失敗します。どうすれば良いのでしょう?

簡単な回答: 多分、セキュアレベルが 0 より大きくなっているのでしょう。
直接シングルユーザモードで再起動して、 カーネルをインストールしてください。

詳しい回答: FreeBSD では、セキュアレベルが 0 より大きい場合、
システムフラグの変更が禁止されます。
現在のセキュアレベルは、次のコマンドを使って調べることができます。

```
# sysctl kern.securelevel
```

セキュアレベルを下げる操作は、できないようになっています。
そのため、カーネルをインストールするには、 シングルユーザモードで起動するか、/etc/rc.conf のセキュリティ設定を変更して再起動する必要があります。 セキュアレベルの詳細は init(8) を、 rc.conf の詳細は /etc/defaults/rc.conf および、 rc.conf(5) のマニュアルページをご覧ください。

7.39. システムの時刻を 1 秒以上変更することができないのです! どうすれば良いのでしょう?

簡単な回答: 多分、セキュアレベルが 1 より大きくなっているのでしょう。
直接シングルユーザモードで再起動して、 時刻の変更をしてください。

詳しい回答: FreeBSD では、セキュアレベルが 1 より大きい場合、 1 秒以上の時刻変更が禁止されます。
現在のセキュアレベルは、次のコマンドを使って調べることができます。

```
# sysctl kern.securelevel
```

セキュアレベルを下げる操作は、できないようになっています。
そのため、システムの時刻を変更するには、 シングルユーザモードで起動するか、/etc/rc.conf のセキュリティ設定を変更して再起動する必要ばあります。 セキュアレベルの詳細は init(8) を、 rc.conf の詳細は /etc/defaults/rc.conf および、 rc.conf(5) のマニュアルページをご覧ください。

7.40. rpc.statd(8) にメモリリークを見つけました! メモリを 256 メガバイトも使っています。

いいえ。それはメモリリークではありませんし、 256 メガバイトのメモリを使っている、ということでもありません。 おそらく (ほとんどの場合)、
処理に都合が良いように非常にたくさんの量のメモリを
そのプロセスのアドレス空間にマッピングしているのでしょう。

技術的な見地から考えても、これは大きな害があることではなく、単に top(1) や ps(1) といったツールの表示に影響がある程度です。

rpc.statd(8) は、(/var にある) ステータスファイルを自分のアドレス空間にマッピングします。マッピングは、後で大きな空間が必要になった時に再マッピングしないで済むよう、非常に大きなサイズを指定して行なわれます。これは、ソースコードに含まれる mmap(2) 関数のマッピング長を示す引数に `0x10000000` が指定されていることからも分かります。この数字が IA32 アーキテクチャの持つアドレススペース全体の 16 分の 1、すなわち、ちょうど 256 メガバイトに相当するのです。

Chapter 8. X Window System と仮想コンソール

8.1. X を動かしたいのですが、どうすればいいのですか?

もっとも簡単な方法は FreeBSD のインストールの際に X を動かすことを指定するだけです。

それから xf86config ツールのドキュメントを読んでこれに従ってください。このツールはあなたのグラフィックカードやマウスなどに合わせて XFree86™ の設定を行うのを助けてくれます。

Xaccel サーバーについて調べてみるのもいいでしょう。 詳しくは Xi Graphics について か Metro Link をご覧ください。

8.2. X を実行しようとして startx と入力したのですが、KDENABIO failed (Operation not permitted) というエラーが表示されます。何かおかしなことをやってしまったんでしょうか?

あなたのシステムは高いセキュアレベルで運用されていますね? 実は、高いセキュアレベルで X を起動することはできないのです。 どうしてなのかについては、init(8) のマニュアルページに書かれています。

では、代わりにどうすれば良いのかお答えしましょう。 基本的に 2 つの方法があります。一つはセキュアレベルを 0 にする (通常、これは /etc/rc.conf で指定します) こと、 もう一つは起動時 (セキュアレベルを上げる前) に xdm(1) を実行するかです。

起動時に xdm(1) を実行する方法の詳細については、 XDM を起動時に起動させるにはどうしますか? を参照してください。

8.3. 私のマウスはなぜ X で動かないのでしょうか?

syscons (デフォルトのコンソールドライバ) を使っているのであれば、それぞれの仮想スクリーンでマウスポインターをサポートするように FreeBSD を設定できます。X でのマウスの衝突を避けるために、syscons は /dev/sysmouse という仮想デバイスをサポートしています。
本物のマウスデバイスから入力されたすべてのマウスのイベントは、 moused を経由して sysmouse デバイスへ出力されます。 一つ以上の仮想コンソールと X の 両方で マウスを使いたい場合、 X Window System 以外の環境でマウスを使うことは可能ですか? を参照して moused を設定してください。

そして、/etc/XF86Config を編集し、 次のように書かれていることを確認してください。

```
Section         Pointer
Protocol        "SysMouse"
Device          "/dev/sysmouse"
```

```
.....
```

上の例は、XFree86 3.3.2 以降の場合の例です。 それより前のバージョンでは、 *Protocol* という部分を *MouseSystems* と置き換える必要があります。

X で /dev/mouse を使うのを好む人もいます。 この場合は、 /dev/mouse を /dev/sysmouse (sysmouse(4) 参照) にリンクしてください。

```
# cd /dev
# rm -f mouse
# ln -s sysmouse mouse
```

8.4. わたしのマウスにはホイール機能が付いているのですが、X で使うことはできますか?

はい、もちろん使えますが、そのためには X クライアントプログラムを適切に設定する必要があります。これについては、 Colas Nahaboo 氏のウェブページ(http://www.inria.fr/koala/colas/mouse-wheel-scroll/) を参照してください。

imwheel というプログラムを使う場合は、 次のような簡単な手順にしたがってください。

1. ホイールイベントの変換

   imwheel は、 マウスのボタン 4、ボタン 5 をキー押下イベントに変換するプログラムです。 そのためホイールマウスで利用するには、マウスホイールのイベントをボタン 4、 ボタン 5 のイベントに変換するマウスドライバを利用する必要があります。
   この変換を行なうには二つの方法があります。 一つは moused(8) で行なう方法、二つめは X サーバ自身に変換を行なわせる方法です。

   a. ホイールイベントの変換に moused(8) を使う

      moused(8) にイベントを変換させるには、 moused(8) 起動時にオプション `-z 4` を追加します。 たとえば、普段 moused(8) を `moused -p /dev/psm0` として起動しているなら、その代わりに `moused -p /dev/psm0 -z 4` とします。 もし、 /etc/rc.conf を使って自動的に起動するように設定しているなら、 /etc/rc.conf の中の `moused_flags` という変数に `-z 4` を追加するだけです。

      そして、5 ボタンマウスを使うことを X サーバに伝える必要があります。 これを行なうには /etc/XF86Config の "Pointer" セクションに `Buttons 5` という行を追加するだけです。 そうすると /etc/XF86Config の "Pointer" は、 たとえば次のようになるでしょう。

      例 *1. moused* による変換を利用してホイールマウスを 使用するための *XFree86 3.3.x* 系列の *XF86Config* の *"Pointer"* セクションの設定例

      ```
      Section "Pointer"
        Protocol        "SysMouse"
      ```

```
  Device          "/dev/sysmouse"
  Buttons         5
EndSection
```

例 2. 自動的なプロトコル認識機能およびボタン配置変換機能を利用し、ホイールマウスを使用するための *XFree86 4.x* 系列の *XF86Config* の *"InputDevice"* セクションの設定例

```
Section "InputDevice"
  Identifier      "Mouse1"
  Driver          "mouse"
  Option          "Protocol" "auto"
  Option          "Device" "/dev/psm0"
  Option          "Buttons" "5"
  Option          "ZAxisMapping" "4 5"
EndSection
```

例 3. ホイールマウスで *Emacs* 上でのページスクロールを 行うための *".emacs"* の設定例

```
;; wheel mouse
(global-set-key [mouse-4] 'scroll-down)
(global-set-key [mouse-5] 'scroll-up)
```

b. X サーバを使ったホイールイベントの変換

moused(8) を起動していなかったり、ホイールイベントの変換に moused(8) を起動したくない場合には、その代わりに X サーバを使うことができます。これには、/etc/XF86Config ファイルを書き換える必要があります。まず最初に必要なのは、マウスがどのプロトコルを使っているのかを確認することです。ほとんどのホイールマウスは "IntelliMouse" プロトコルを使用していますが、XFree86 サーバはその他のプロトコル、たとえば Logitech MouseMan+ マウスが利用している "MouseManPlusPS/2" プロトコルなどもサポートしています。使用されているプロトコルが確認できたら "Pointer" セクションに `Protocol` の行を追加してください。

つぎに、ホイールのスクロールイベントをマウスボタン 4、マウスボタン 5 に割り当てることを X サーバに伝えます。これを行なうには `ZAxisMapping` オプションを使用します。

たとえば、moused(8) が起動していない状態で、PS/2 マウスポートに IntelliMouse が接続されているとしたら /etc/XF86Config はおそらく次のようになります。

2. X サーバによる変換を利用してホイールマウスを使用するための XF86Config の "Pointer" セクションの設定例

```
Section "Pointer"
  Protocol        "IntelliMouse"
  Device          "/dev/psm0"
  ZAxisMapping    4 5
EndSection
```

3. imwheel のインストール

   さて、つぎに Ports Collection から imwheel をインストールします。 これがあるのは x11 カテゴリです。 このプログラムは、 マウスイベントをキーボードイベントに変換します。 たとえば、マウスホイールを前に回した時、 imwheel は `PageUp` をアプリケーションプログラムに送るような動作をするわけです。 Imwheel はホイールイベントとキーボード押下の対応を設定ファイルを使って設定するため、 アプリケーション毎に異なる対応を持たせることも可能です。 imwheel のデフォルトの設定ファイルは /usr/X11R6/etc/imwheelrc にインストールされます。 これを ~/.imwheelrc にコピーして編集し、 お好きなように imwheel で利用したいアプリケーションの設定をカスタマイズしてください。 設定ファイルの書式は imwheel(1) に説明されています。

4. Emacs で Imwheel を使うように設定する (必須ではありません)

   emacs や Xemacs で利用するには、 ~/.emacs にいくらか書き加える必要があります。 emacs の場合は次の部分を追加してください。

   例 *4. Imwheel* を利用するための *Emacs* の設定例

```
;;; For imwheel
(setq imwheel-scroll-interval 3)
(defun imwheel-scroll-down-some-lines ()
(interactive)
(scroll-down imwheel-scroll-interval))
(defun imwheel-scroll-up-some-lines ()
(interactive)
(scroll-up imwheel-scroll-interval))
(global-set-key [?\M-\C-\)] 'imwheel-scroll-up-some-lines)
(global-set-key [?\M-\C-\(] 'imwheel-scroll-down-some-lines)
;;; end imwheel section
```

   Xemacs の場合は ~/.emacs に次の部分を追加してください。

   例 *5. Imwheel* を利用するための *XEmacs* の設定例

```
;;; For imwheel
(setq imwheel-scroll-interval 3)
(defun imwheel-scroll-down-some-lines ()
```

```
(interactive)
(scroll-down imwheel-scroll-interval))
(defun imwheel-scroll-up-some-lines ()
(interactive)
(scroll-up imwheel-scroll-interval))
(define-key global-map [(control meta \))] 'imwheel-scroll-up-some-lines)
(define-key global-map [(control meta \()] 'imwheel-scroll-down-some-lines)
;;; end imwheel section
```

5. Imwheel の実行

   インストールが完了していれば、単に xterm (訳注: 日本語環境で広く使われている kterm でも構いません) から imwheel を入力するだけで起動できます。起動するとバックグラウンドで動作し、すぐに利用できます。imwheel をいつも使うように設定するには、.xinitrc か .xsession のファイルにそのままコマンドを追加してください。imwheel が PID ファイルに関する警告を表示するかも知れませんが、
   無視しても危険はありません。この警告が意味を持つのは、 Linux 版の imwheel だけです。

8.5. X のメニューやダイアログボックスがうまく動きません。

Num Lock キーをオフにしてください。

Num Lock キーがデフォルトで起動時にオンになる場合は、XF86Config ファイルの Keyboard セクションに以下の行を加えてもいいでしょう。

```
# Let the server do the NumLock processing.  This should only be
# required when using pre-R6 clients
  ServerNumLock
```



この問題は XFree86 3.2 以降では解決しています。

8.6. 仮想コンソールとは何ですか? どうやったら使えますか?

仮想コンソールは、簡単にいうと、ネットワークや X を動かすなどの複雑なことを行なわずに、いくつかのセッションを同時に行なうことを可能にします。

システムのスタート時には、起動メッセージが出た後に login プロンプトが表示されます。そこでログイン名とパスワードを入力すると 1 番目の仮想コンソール上で仕事 (あるいは遊び) を始めることができます。

他のセッションを始めたい場合もあるでしょう。
それは動かしているプログラムのドキュメントを見たり、FTP の転送が終わるまで待つ間、メールを読もうとしたりすることかもしれません。Alt-F2 を押す (Alt キーを押しながら F2 キーを押す) と、2 番目の「仮想コンソール」で ログインプロンプトが待機していることがわかります。最初のセッションに戻りたいときは Alt-F1 を押します。

標準の FreeBSDインストールでは、 3 枚 (3.3-RELEASE では 8 枚) の仮想コンソールが有効になっていて、 Alt-F1、 Alt-F2、 Alt-F3 で仮想コンソール間の切替えを行ないます。

より多くの仮想コンソールを有効にするには、 /etc/ttys (ttys(5) 参照) を編集して "Virtual terminals" のコメント行の後に ttyv4 から ttyvc の手前までのエントリを加えます (以下の例は先頭には空白は入りません)。

```
# /etc/ttys には ttyv3 がありますので
# "off" を "on" に変更します。
ttyv3   "/usr/libexec/getty Pc"         cons25  on secure
ttyv4   "/usr/libexec/getty Pc"         cons25  on secure
ttyv5   "/usr/libexec/getty Pc"         cons25  on secure
ttyv6   "/usr/libexec/getty Pc"         cons25  on secure
ttyv7   "/usr/libexec/getty Pc"         cons25  on secure
ttyv8   "/usr/libexec/getty Pc"         cons25  on secure
ttyv9   "/usr/libexec/getty Pc"         cons25  on secure
ttyva   "/usr/libexec/getty Pc"         cons25  on secure
ttyvb   "/usr/libexec/getty Pc"         cons25  on secure
```

多くするか少なくするかはあなたの自由です。
より多くの仮想ターミナルを使うとより多くのリソースを使うことになります。 8MB 以下のメモリしかない場合はこれは重要な問題です。 もし必要があれば secure を insecure に変更してください。



X を使いたいのであれば、 最低一つの仮想ターミナル (のエントリ) を使わずに残しておくか、 off にしておく必要があります。 つまり、12 個の Alt-ファンクションキーすべてでログインプロンプトを 出したいのならば、 残念ながら X は利用できないということです。 同じマシンで X サーバーも動かしたいのならば 11 個しか使えません。

仮想コンソールを無効にするもっとも簡単な方法は、 コンソールを off にすることです。 たとえば 12 個すべてのターミナルを割り当てている状態で X を動かしたいときは、 仮想ターミナル 12 を変更します。

```
ttyvb   "/usr/libexec/getty Pc"         cons25  on secure
```

これを次のように変更します。

```
ttyvb   "/usr/libexec/getty Pc"         cons25  off secure
```

キーボードにファンクションキーが 10 個しかないのであれば、 次のように設定します。

```
ttyv9   "/usr/libexec/getty Pc"         cons25  off secure
ttyva   "/usr/libexec/getty Pc"         cons25  off secure
ttyvb   "/usr/libexec/getty Pc"         cons25  off secure
```

(これらの行を消すだけでもいいです。)

/etc/ttys を編集したら、次は十分な数の仮想ターミナルデバイスを作らなくてはなりません。
もっとも簡単な方法を示します。

```
# cd /dev
# ./MAKEDEV vty12(12 個のデバイスをつくる場合)
```

さて、仮想コンソールを有効にするもっとも簡単 (そして確実) な方法は、
再起動することです。しかし、再起動したくない場合は、 X ウィンドウシステムを終了させて次の内容を
(root権限で) 実行します。

```
# kill -HUP 1
```

重要な点は、 このコマンドを実行する前に X ウィンドウシステムを完全に終了させておくことです。
もしそうしないと kill コマンドを実行した後、 システムはおそらくハングアップするでしょう。

8.7. X から仮想コンソールに切替えるにはどうすればよいのですか?

仮想コンソールへ戻るには Ctrl + Alt + Fn を使ってください。最初の仮想コンソールへは Ctrl + Alt + F1 で戻れます。

テキストコンソールへ移った後は、その中で移動するのに 今度はいつもどおり Alt + Fn を使ってください。

X のセッションへ戻るには X の走っている仮想コンソールへ 切り替える必要があります。もしあなたが X をコマンドラインから 実行していたのであれば (たとえば startx を使う) X のセッションはそれを実行したテキストコンソールではなく
最初の使われていない仮想コンソールに割り当てられているはずです。 あなたが仮想端末を 8 個用意している場合は X を 9 番目の コンソールにいるはずで、 Alt + F9 を使うことになります。



X に戻るには、 3 枚の仮想コンソールが有効になっている場合は Alt-F4 です。
有効な仮想コンソールの数 +1 のファンクションキーの 位置に X が割り当てられます。

8.8. XDM を起動時に起動させるにはどうしますか?

xdm の起動方法については二つの流派があります。 一方の流派では提供された例を使用して xdm を /etc/ttys (ttys(5) 参照) から起動し、もう一方の流派では xdm を単に rc.local (rc(8) 参照) または /usr/local/etc/rc.d においた X.sh スクリプトから起動します。
どちらも正しく、片方が動作しない場合は、もう片方が動作するでしょう。
どちらも場合でも結果は同じであり、X はグラフィカルな login: プロンプトを表示します。

ttys を利用する方法の利点は、 どの vty で X が起動したかの記録が残せることと、 ログアウト時に X サーバを再起動する責任を init に押しつけることができることでしょう。

rc.local からロードされる場合、 xdm は引数を持たずに (すなわち、デーモンとして) 起動します。 xdm は

getty が起動した後にロードされなければなりません。 そうでないと、xdm は getty と衝突し、コンソールをロックアウトしてしまいます。 この問題に対処する最善の方法は、起動スクリプト (訳注: rc.local のこと) で 10 秒ほどの sleep を実行させ、 その後に xdm をロードすることです。

/etc/ttys から xdm を起動させている場合には、 xdm と getty が衝突する可能性があります。この問題を回避するには、/usr/X11R6/lib/X11/xdm/Xservers に vt 番号を追加してください。

```
:0 local /usr/X11R6/bin/X vt4
```

上の例は、/dev/ttyv3 を X サーバに対応させます。番号は 1 から始まりますので注意してください。 X サーバは vty を 1 から数えますが、 FreeBSD カーネルは vty を 0 から数えます。

8.9. xconsole を動かそうとすると Couldn't open console とエラーが出ます。

X を startx で起動しますと、/dev/console のパーミッションは 変更できないようになっていますので、xterm -C や xconsole は動きません。

これはコンソールのパーミッションが、 標準ではそのように設定されているからです。マルチユーザシステムでは、
ユーザの誰もがシステムコンソールに書き込むことが可能である必要は必ずしもありません。 VTY を使い直接マシンにログインするユーザのために、 このような問題を解決するために fbtab(5) というファイルがあります。

要点を述べると、次のような形式の行を /etc/fbtab (fbtab(5) 参照) に加えます。

```
/dev/ttyv0 0600 /dev/console
```

そうすると、 /dev/ttyv0 からログインしたユーザが コンソールを所有することになるでしょう。

8.10. わたしはいつも XFree86 を一般ユーザから起動していたのですが、 最近になって root ユーザでなければな らないと言われるようになりました。

すべての X サーバは、 ビデオハードウェアに直接アクセスするために root ユーザで実行される必要があります。 古いバージョンの XFree86 (⇐ 3.3.6) に含まれるすべてのサーバは、 自動的に root 権限で実行されるように (root ユーザに setuid されて) インストールされます。 X サーバは大きく複雑なプログラムであり、これは明らかにセキュリティを危険に晒す要因となります。 そのため新しいバージョンの XFree86 では、 サーバを root ユーザに setuid しないでインストールするようになりました。

X サーバを root ユーザで動かすというのは、明らかにセキュリティ的に不適当で受け入れられないことです。 X を一般ユーザで実行するには、二つの方法があります。 一つは xdm や、その他のディスプレイマネージャ (たとえば kdm など) を使うこと、 もう一つは Xwrapper を使うことです。

xdm は、グラフィカルなログイン画面を扱うデーモンです。通常、起動時に実行され、各ユーザの認証とユーザセションを開始させる機能を実現します。基本的に、getty と login のグラフィック版、と考えて良いでしょう。xdm の詳細については、XFree86 関連文書 および FAQ 項目をご覧ください。

Xwrapper とは、X サーバ用のラッパ (wrapper) のことです。これは必要なセキュリティを確保しつつ、一般ユーザが X サーバを実行できるようにした小さなユーティリティで、コマンドライン引数の正当性チェックを行ない、それを通過すれば適切な X サーバを起動します。何らかの理由でディスプレイマネージャを使いたくない場合に これを使うと良いでしょう。Ports Collection 全体をインストールしていれば、/usr/ports/x11/wrapper にあります。

8.11. 私の PS/2 マウスは X ウィンドウシステム上でうまく動きません。

あなたのマウスとマウスドライバがうまく同期していないからかもしれません。

FreeBSD 2.2.5 までのバージョンでは、X から仮想ターミナルへ切替えて、また X へ戻ると再同期するかもしれません。
この問題がよく起きるようであれば、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書いてカーネルを再構成してみてください。

```
options PSM_CHECKSYNC
```

もし、カーネルの再構築を行なったことがないのであれば、
カーネルを構築するの項を参照してください。

このオプションにより、マウスとドライバの同期で問題が起きる可能性は少なくなるでしょう。もしそれでもこの問題が起きるようならば、再同期させるにはマウスを動かさないようにしておいてマウスボタンのどれかを押してください。

このオプションは残念ながらすべてのシステムで働くわけではなく、また、PS/2 マウスポートにつながれているのが タップ (tap) 機能を持つ アルプス社製 GlidePoint デバイスの場合、タップ機能が無効となってしまいます。

FreeBSD 2.2.6 以降のバージョンでは、同期のチェック方法が少し改善されたので標準で有効になっています。GlidePoint でもうまく働きます (同期チェックが標準の機能になったので PSM_CHECKSYNC オプションはこれらのバージョンからは削除されました)。しかしながら、まれにドライバが間違って (訳注: 問題がないのに) 同期に関して問題があると報告し、カーネルから

```
psmintr: out of sync (xxxx != yyyy)
```

というメッセージが出力されて、マウスが正しく動作していないように見える
ことがあるかもしれません。

もしこのようなことが起こる場合には、PS/2 マウスドライバのフラグに 0x100 を指定して同期チェックを無効にしてください。システムの起動時に “-c” 起動オプションを与えて *UserConfig* に入ります。

```
boot: -c
```


```
boot: -c
```


UserConfig のコマンドラインで以下のように入力してください。

```
UserConfig> flags psm0 0x100
UserConfig> quit
```

8.12. MouseSystems の PS/2 マウスがうまく動きません。

MouseSystems の PS/2 マウスのあるモデルは、高解像度モードの場合にのみ正しく動作するということが報告されています。 それ以外のモードでは、マウスカーソルがしょっちゅうスクリーン左上に行ってしまうかもしれません。

残念ながら FreeBSD 2.0.X や 2.1.X のバージョンでは、 この問題を解決する方法はありません。 2.2 から 2.2.5 のバージョンでは、 以下のパッチを /sys/i386/isa/psm.c に適用しカーネルの再構築を行なってください。

もし、カーネルの再構築を行なったことがないのであれば、 カーネルの構築の項を参照してください。

```
@@ -766,6 +766,8 @@
    if (verbose >= 2)
        log(LOG_DEBUG, "psm%d: SET_DEFAULTS return code:%04x\n",
            unit, i);
+    set_mouse_resolution(sc->kbdc, PSMD_RES_HIGH);
+
 #if 0
    set_mouse_scaling(sc->kbdc);    /* 1:1 scaling */
    set_mouse_mode(sc->kbdc);               /* stream mode */
```

FreeBSD 2.2.6 以降のバージョンでは、 PS/2 マウスドライバのフラグに 0x04 を指定してマウスを高解像度モードにします。 システムの起動時に -c 起動オプションを与えて *UserConfig* に入ります。

```
boot: -c
```


UserConfig のコマンドラインで以下のように入力してください。


```
UserConfig> flags psm0 0x04
UserConfig> quit
```

マウスに関する不具合の他の原因の可能性については、 直前のセクションも見てみてください。

8.13. X のアプリケーションを構築する時に、 imake can't find Imake.tmpl となります。どこにあるのでしょうか?

Imake.tmpl は X の標準アプリケーション構築ツールである Imake パッケージの一部です。 Imake.tmpl は、 X アプリケーションの構築に必要な多くのヘッダファイルと同様に、 X のプログラムディストリビューションに含まれています。 `sysinstall` を使うか、 手動で X のディストリビューションファイルからインストールすることができます。

8.14. マウスのボタンを入れ替える方法はありますか?

.xinitrc か .xsession で

```
xmodmap -e "pointer = 3 2 1"
```

というコマンドを実行してください。

8.15. スプラッシュスクリーンのインストールはどうするのですか。 どこで見つけることができますか?

FreeBSD 3.1 のリリース直前に、起動メッセージの表示期間に いわゆる "スプラッシュ" スクリーンを表示させることができる新しい機能が追加されました。
いまのところスプラッシュスクリーンは 256 色のビットマップ (*.BMP) か ZSoft PCX (*.PCX) ファイルです。 それに加えて、標準の VGA アダプタでの動作させるには 320x200 以下の解像度である必要があります。 カーネルに VESA サポートを追加すれば 1024x768 までのより大きいビットマップを使用できます。 VESA サポートを有効化するにはまず、 カーネルが `VM86` カーネルオプションとともにコンパイルされている必要があることに注意してください。 VESA サポートそのものは `VESA` カーネルコンフィグオプション によって直接カーネル中にコンパイルするか、起動時に VESA kld モジュールを読み込ませることができます。

スプラッシュスクリーンを使うには、 FreeBSD の起動プロセスをコントロールするスタートアップファイルを書き換える必要があります。
これらのファイルは FreeBSD 3.2 のリリース以前に変更されましたので、現在は、スプラッシュスクリーンを読み込む方法が二つあります。

- FreeBSD 3.1 の場合

  まず最初のステップは、 スプラッシュスクリーンのビットマップ版を探してくることです。 3.1-RELEASE では Windows のビットマップ形式のスプラッシュスクリーンだけをサポートしています。 お望みのスプラッシュスクリーンを見つけたなら、それを /boot/splash.bmp にコピーします。次に、これらの行が書かれた /boot/loader.rc ファイルが必要です。

  ```
  load kernel
  load -t splash_image_data /boot/splash.bmp
  load splash_bmp
  ```

```
autoboot
```

- FreeBSD 3.2 以降の場合

  PCX 形式のスプラッシュスクリーンのサポートが追加されると同時に、 FreeBSD 3.2 には起動プロセスを設定する、 より洗練された方法が含まれています。 もしお望みなら、上に示した FreeBSD 3.1 用の方法を使うこともできます。 もしそうしたくて、かつ PCX 形式を使いたいなら、 `splash_bmp` を `splash_pcx` と読み換えてください。 そうではなくて、新しい起動設定方法を使うのなら、 次の数行が書かれた /boot/loader.rc ファイルと、

  ```
  include /boot/loader.4th
  start
  ```

  次の数行が含まれた /boot/loader.conf ファイルを作ることが必要です。

  ```
  splash_bmp_load="YES"
  bitmap_load="YES"
  ```

  この例では、スプラッシュスクリーンとして /boot/splash.bmp を使うことを想定しています。PCX 形式のファイルを使う場合には、 そのファイルを /boot/splash.pcx にコピーして、 上で示したように /boot/loader.rc を作ります。 そして、次の内容の /boot/loader.conf というファイルを作ってください。

  ```
  splash_pcx_load="YES"
  bitmap_load="YES"
  bitmap_name="/boot/splash.pcx"
  ```

さて、あとはスプラッシュスクリーンを用意するだけです。 それには http://www.baldwin.cx/splash/ のギャラリーをサーフしてみてください。

8.16. X で Windows™ キーを使うことはできるのでしょうか?

はい、もちろん。 どういう動作をするかについて定義するには xmodmap(1) を使います。

標準的な "Windows™" キーボードの場合、 対応するキーコードは 3 種類あります。

- 115 - 左の Ctrl と Alt の間にある Windows™ キー
- 116 - 右の Alt と Gr の間にある Windows™ キー
- 117 - 右の Ctrl の左隣にあるメニューキー

左にある Windows™ キーを押すとカンマ記号が入力されるようにするには、 こんな風にします。

```
# xmodmap -e "keycode 115 = comma"
```

設定を反映させるには、おそらくウィンドウマネージャを再起動する必要があります。

Windows™ キーのキーマップを X 起動時に毎回、 自動的に有効化するには xmodmap コマンドを ~/.xinitrc に追加するか、 もしくはおすすめできる方法として ~/.xmodmaprc というファイルを作成して、 そのファイルの一行一行に xmodmap のオプションを記述し、次の一行

```
xmodmap $HOME/.xmodmaprc
```

を ~/.xinitrc に追加するという方法があります。

たとえば、先ほどあげた三つのキーを F13、F14、F15 に割り当てるとします。
こうしておけば、後ほど示すように、アプリケーションや ウィンドウマネージャの便利な機能を
その三つのキーに簡単に割り当てることができます。

こうするには、次の内容を ~/.xmodmaprc に追加します。

```
keycode 115 = F13
keycode 116 = F14
keycode 117 = F15
```

たとえば fvwm2 を使っていたら、 F13 をカーソル下のウィンドウのアイコン化、 F14 をウィンドウの前面/背面化、 F15 を、あたかもデスクトップにカーソルが存在しないかのように、メインワークスペース (アプリケーション) のメニューを呼び出せる機能に割り当てられます。
最後の機能は、そのデスクトップがまったく見えないときに便利です。
(また、キートップのロゴにもぴったりです)

~/.fvwmrc の次のエントリは、前述の 設定を実現します。

```
Key F13         FTIWS    A        Iconify
Key F14         FTIWS    A        RaiseLower
Key F15         A        A        Menu Workplace Nop
```

Chapter 9. ネットワーキング

9.1. ディスクレスブート (diskless boot) に関する情報はどこで得られますか?

"ディスクレスブート (diskless boot)" というのは、FreeBSD がネットワーク上で起動し、必要なファイルを自分のハードディスクではなくてサーバから読み込むものです。 詳細については FreeBSD ハンドブックの「ディスクレスブート」を読んでください。

9.2. FreeBSD をネットワークのルータ (router) として使用することはできますか?

インターネット標準やこれまでのよい経験によって指摘されている通り、 FreeBSD は標準ではパケットを転送 (forward) するように設定されていません。 しかし、 rc.conf(5) の中で次の変数の値を YES とする事によってこの機能を有効にすることができます。

```
gateway_enable=YES          # Set to YES if this host will be a gateway
```

このオプションによって sysctl(8) の変数 net.inet.ip.forwarding が 1 になります。

ほとんどの場合、 ルータについての情報を同じネットワークの他の計算機等に知らせるために、経路制御のためのプロセスを走らせる必要があるでしょう。 FreeBSD には BSD の標準経路制御デーモンである routed(8) が付属していますが、より複雑な状況に対処するためには GaTeD(http://www.gated.org/ から入手可能) を使用することもできます。 3_5Alpha7 において FreeBSD がサポートされています。

注意してほしいのは、FreeBSD をこのようにして使用している場合でも、ルータに関するインターネット標準の必要条件を完全には満たしていない ということです。しかし、普通に使用する場合にはほとんど問題ありません。

9.3. Win95 の走っているマシンを、FreeBSD 経由でインターネットに接続できますか?

通常、この質問が出てくる状況は自宅に二台の PC があり、一台では FreeBSD が、もう一台では Win95 が走っているような場合です。 ここでやろうとしていう事は FreeBSD の走っている計算機をインターネット に接続し、Win95 の走っているマシンからは FreeBSD の走っているマシンを経由して接続を行なう事です。 これは二つ前の質問の特別な場合に相当します。

...で、答えは「はい」です。 FreeBSD 3.x のユーザモード ppp には -nat オプションがあります。 ppp を -nat オプション付きで起動し、 /etc/rc.conf にある gateway_enable を *YES* に設定します。 そして Windows マシンを正しく設定すれば、 きちんと動作するでしょう。

設定に関するさらに詳しい情報は、 Steve Sims 氏による Pedantic PPP Primer にあります。

カーネルモード ppp を利用する場合や、 インターネットとのイーサネット接続が利用できる場合は、

natd を利用する必要があります。 この FAQ の natd のセクションを参照してください。

9.4. ISC からリリースされている BIND の最新版はコンパイルできないんでしょうか?

BIND の配布物と FreeBSD とでは cdefs.h というファイルの中でデータ型の矛盾があります。 compat/include/sys/cdefs.h を削除してください。

9.5. FreeBSD で SLIP と PPP は使えますか?

使えます。FreeBSD を用いて他のサイトに接続する場合には、 slattach(8)、sliplogin(8)、ppp(8) そして pppd(8) のマニュアルページをご覧ください。 ppp(8) と pppd(8) は、 PPP のサーバ、クライアント両方の機能を持っています。 その一方で、sliplogin(8) は SLIP のサーバ専用で、 slattach(8) は SLIP のクライアント専用です。

これらを使うためのさらなる情報については、ハンドブックの PPP と SLIP の章をご覧ください。

「シェルアカウント」を通じてのみインターネットへアクセス可能な場合、 slirp package みたいなものが欲しくなるかもしれませんね。 これを使えば、ローカルマシンから直接 ftp や http のようなサービスに (限定的ではありますが) アクセスすることができます。

9.6. FreeBSD は NAT か IP マスカレードをサポートしていますか?

ローカルなサブネット (一台以上のローカルマシン) を持っているが、 インターネットプロバイダから 1 つしか IP アドレスの割り当てを受けていない場合 (または IP アドレスを動的に割り当てられている場合でも)、 natd(8) プログラムを使いたくなるかもしれませんね。 natd を使えば、 1 つしか IP アドレスを持っていない場合でも、 サブネット全体をインターネットに接続させることができます。

ppp(8) も同様の機能を持っており、-nat スイッチで有効にすることができます。 どちらの場合も alias ライブラリ (libalias(3)) が使われます。

9.7. /dev/ed0 デバイスを作成することができません。

Berkeley UNIX におけるネットワークの構成において、 ネットワークのインタフェースはカーネルコードからのみ、 直接あつかうことができます。 より詳しく知りたい場合は、 /etc/rc.network というファイルや、 このファイルの中に書いてある、 さまざまなプログラムについてのマニュアルページを見てください。 それでもまだ分からない場合には、 他の BSD 系の OS のネットワーク管理についての本を読むべきでしょう。 ごく少しの例外をのぞいては、FreeBSD のネットワーク管理は SunOS 4.0 や Ultrix と基本的に同じです。

9.8. Ethernet アドレスのエイリアス (alias) はどのようにして設定できますか?

ifconfig(8) のコマンドラインに netmask 0xffffffff を追加して、次のように書いてください。

```
# ifconfig ed0 alias 204.141.95.2 netmask 0xffffffff
```

9.9. 3C503 で他のネットワークポートを使用するにはどのようにすればよいですか?

他のポートを使用したい場合には、ifconfig(8) のコマンドラインにパラメータを追加しなければなりません。デフォルトでは link0 が用いられるようになっています。BNC のかわりに AUI ポートを使用したい場合には、link2 というパラメータを追加してください。これらのフラグは、/etc/rc.conf (rc.conf(5) 参照) にある ifconfig_* の変数を使って指定されるはずです。

9.10. FreeBSD との間で NFS がうまくできません。

PC 用のネットワークカードによっては、NFS のような、ネットワークを酷使するアプリケーションにおいて問題を起こすものがあります。

この点に関しては FreeBSD ハンドブックの「NFS」を参照してください。

9.11. 何故 Linux のディスクを NFS マウントできないのでしょうか?

Linux の NFS のコードには、許可されたポートからのリクエストしか受けつけないものがあります。以下を試してみてください。

```
# mount -o -P linuxbox:/blah /mnt
```

9.12. 何故 Sun のディスクを NFS マウントできないのでしょうか?

SunOS 4.X が走っている Sun Workstation は、許可されたポートからのマウント要求しか受けつけません。以下を試してみてください。

```
# mount -o -P sunbox:/blah /mnt
```

9.13. mountd から can't change attributes というメッセージがずっと出続けていて、 FreeBSD の NFS サーバでは bad exports list と表示されます。これは何が原因なのでしょう?

最も良くある問題は、exports(5) のマニュアルページの以下の部分を正しく理解していないことです。

このファイルの各行 (# ではじまるコメント行を除く) は、 NFS サーバのローカルファイルシステムに存在する、 他のホストにエクスポートされるマウントポイント (複数可) と、 それに対するエクスポートフラグを指定します。 特定のエクスポート先ホストおよび、すべてのホストに適用されるデフォルトエントリは両方とも、
サーバの各ローカルファイルシステムに対して一回だけしか指定できません。

さて、ありがちな間違いをご覧になればはっきりするでしょう。 もし /usr 以下が単一のファイルシステムである (つまり /usr に何もマウントされない) 場合、 次の exports リストは正しくありません。

```
/usr/src   client
/usr/ports client
```

一つのファイルシステムに対して属性の指定が二行になっています。 /usr は同じホスト client にエクスポートされますから、 正しい書き方は次のようになります。

```
/usr/src /usr/ports  client
```

もう一度マニュアルページの文章を確認すると、
あるホストにエクスポートされる各ファイルシステムの属性は
すべて一行に書かれていなければならない、となっています
(ここでは、「アクセス可能なすべてのホスト」
もう一つの独立したホストとして扱われることに注意してください)。
このことは、ファイルシステムをエクスポートするために
奇妙な書式を使わなければならない原因にもなっているのですが、
ほとんどの人にとって、これは問題にはならないでしょう。

次に示すのは、有効な exports リストの例です。 ここでは、/usr と /exports がローカルファイルシステムです。

```
# Export src and ports to client01 and client02, but only
# client01 has root privileges on it
/usr/src /usr/ports -maproot=0    client01
/usr/src /usr/ports               client02
# The "client" machines have root and can mount anywhere
# up /exports. The world can mount /exports/obj read-only
/exports -alldirs -maproot=0      client01 client02
/exports/obj -ro
```

9.14. PPP で NeXTStep に接続する際に問題があるのですが。

/etc/rc.conf (rc.conf(5) 参照) の中で次の変数を NO にして、 TCP extension を無効にしてみてください。

```
tcp_extensions=NO
```

Xylogic の Annex も同様の問題がありますので、 Annex 経由で PPP を行なう場合にもこの変更を行ってください。

9.15. IP マルチキャスト (multicast) を有効にするには?

FreeBSD 2.0 かそれ以降では、 標準の状態で完全にマルチキャストに対応しています。現在使用している計算機をマルチキャストのルータ (router) として使用するには、 MROUTING というオプションを定義したカーネルを作ったうえで、 mrouted を走らせる必要があります。2.2 かそれ以降の FreeBSD ならば、 /etc/rc.conf でフラグ mrouted_enable を YES にセットしておくことで、起動時に mrouted を起動できます。

MBONE 用のツールは ports 内の専用のカテゴリー mbone にあります。 vic や vat といった会議用のツールを探している場合は、 この場所を見てください。

詳しい情報は Mbone Information Web にあります。

9.16. DEC の PCI チップセットを用いているネットワークカードには、どのような物がありますか?

Glen Foster 氏による一覧に、 最近の製品を追加したものを以下に示します。

```
Vendor          Model
-----------------------------------------------
ASUS            PCI-L101-TB
Accton          ENI1203
Cogent          EM960PCI
Compex          ENET32-PCI
D-Link          DE-530
Dayna           DP1203, DP2100
DEC             DE435, DE450
Danpex          EN-9400P3
JCIS            Condor JC1260
Linksys         EtherPCI
Mylex           LNP101
SMC             EtherPower 10/100 (Model 9332)
SMC             EtherPower (Model 8432)
TopWare         TE-3500P
```

```
Znyx            (2.2.X) ZX312, ZX314, ZX342, ZX345, ZX346, ZX348
              (3.X) ZX345Q, ZX346Q, ZX348Q, ZX412Q, ZX414, ZX442,
                    ZX444, ZX474, ZX478, ZX212, ZX214 (10mbps/hd)
```

9.17. 何故自分のサイトのホストに対して FQDN を使用する必要があるのですか?

実際にはそのホストは別のドメインにあるのではないですか。 たとえば、foo.bar.edu というドメインの中から、 bar.edu ドメインにある mumble というホストを指定したい場合には、 mumble だけではダメで、 mumble.bar.edu という FQDN (fully-qualified domain name) で指定しなければなりません。

伝統的に、BSD の BIND のリゾルバ (resolver) ではこのような事は可能でしたが、 FreeBSD に入っている bind (named(8) 参照) の現在のバージョンでは、 自分以外のドメインに対して FQDN でない別名を自動的につけてくれるような事はありません。 したがって mumble というホスト名は、 mumble.foo.bar.edu という名前か、もしくは root ドメイン内にある場合にしか適用されません。

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行なわれていた 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 ファイル (resolv.conf(5) 参照) の中で

```
domain foo.bar.edu
```

と書いてある行を、 search foo.bar.edu bar.edu のように書きかえることで、上のような事ができます。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。

9.18. すべてのネットワークの操作に対して Permission denied というメッセージが表示されるのですが。

IPFIREWALL オプションを付けてカーネルをコンパイルした場合には、 2.1-STABLE の開発の途中から変更になった 2.1.7R の標準的な方針として、 明示的に許可されていないすべてのパケットは落とされる設定 になっている事を覚えておいてください。

もしファイアウォールの設定を間違えた場合にネットワークの操作が再びできる ようにするには、root でログインして次のコマンドを実行してください。

```
# ipfw add 65534 allow all from any to any
```

/etc/rc.conf に firewall_type='open' を追加してもよいでしょう。

FreeBSD のファイアウォールの設定についての情報は FreeBSD ハンドブックの「ファイアウォール」にあります。

9.19. IPFW のオーバヘッドはどのくらいでしょうか?

この答えは、 使っているルールセットとプロセッサのスピードによってほとんど決まります。 イーサネットに対して少しのルールセットだけを使っている場合には、 ほとんどその影響は無視できる程度です。 実際の測定値を見ないと満足できない方々のために、 実際の測定結果をお見せしましょう。

次の測定は 486-66 (訳注: Intel 社製 CPU i486、66MHz のこと) 上で 2.2.5-STABLE を使用して行なわれました。 IPFW は変更が加えられて、`ip_fw_chk` ルーチン内でかかる時間を 測定して 1000 パケット毎に結果をコンソールに表示するようになっています。

それぞれ 1000 ずつのルールが入っている 2 つのルールセットでテストが行なわれました。 ひとつ目のルールセットは最悪のケースを見るために

```
ipfw add deny tcp from any to any 55555
```

というルールを繰り返したものです。

IPFW のパケットチェックルーチンは、 パケットが (ポート番号のせいで) このルールにマッチしないことがわかるまでに、 何度も実行されます。そのため、これは最悪のケースを示します。 このルールを 999 個繰り返し並べた後に

```
allow ip from any to any
```

が書かれています。

2つ目のルールセットは、なるべく早くチェックが終了するように書かれたものです。

```
ipfw add deny ip from 1.2.3.4 to 1.2.3.4
```

このルールでは、発信元の IP アドレスがマッチしないので、 チェックはすぐに終了します。上のルールセットとおなじように、 1000 個目のルールは

```
allow ip from any to any
```

です。

1 つ目のルールセットの場合、 パケットあたりのオーバヘッドはおよそ 2.703ms/packet、 これはだいたい 1 つのルールあたり 2.7 マイクロ秒かかっていることになります。 したがって、 このルールにおけるパケット処理時間の理論的な限界は、 毎秒約 370 パケットです。 10Mbps のイーサネットで 1500 バイト以下のパケットサイズを仮定すると、 バンド幅の利用効率は 55.5% が限界となることになります。

2 つ目のルールセットでは、それぞれのパケットがおよそ 1.172msで処理されていますので、 だいたい 1 つのルールあたり 1.2 マイクロ秒かかっていることになります。 パケット処理時間の理論的な限界は、 毎秒約 853 パケットとなりますので、 10Mbps Ethernet のバンド幅を使い切ることができます。

このテストでのルール数は多過ぎるため、　実際に使用する際の結果を反映している訳ではありません。
これらは上に示した数値を出すためだけに用いられたものです。
効率の良いルールセットを作るためには、次のような事を考えておけばよいでしょう。

- 「確定している」ルールは先頭の方に持ってきてください。　これは、多数の　TCP のトラフィックがこのルールで処理されるためです。　そしてこのルールの前には　`allow　tcp` という記述を置かないでください。
- 良く使われるルールを、あまり良く使われないルールよりも　前の方に　(もちろんファイアウォールの許可設定を変えない範囲で)　持ってきてください。　`ipfw　-a　l` のようしてパケット数の統計を取ることで、
  どのルールが最もよく使われているかを調べることができます。

9.20. ipfw(8) fwd ルールを使って他のマシンにサービスをリダイレクトしたのですが、うまく動いてくれないようです。どうしてなんでしょう?

おそらく、あなたが期待している動作とは、　単なるパケット転送ではなくネットワークアドレス変換 (NAT)　と呼ばれるものだからでしょう。　"fwd"　ルールは文字どおり、本当に転送しか行ないません。パケットの中身については一切手を加えないのです。そのため、次のようなルールを設定したとすると、

```
01000 fwd 10.0.0.1 from any to foo 21
```

宛先アドレスに *foo* と書かれたパケットが このルールを設定したマシンに到着した場合、そのパケットは *10.0.0.1* に転送されますが、宛先アドレスは *foo* のままになります。つまり、パケットに宛先アドレスが *10.0.0.1*　に書き換えられるということはありません。　自分宛でないパケットを受けとったマシンは、おそらくほとんどの場合、そのパケットを破棄すると思います。　そのため　"fwd"　ルールは、そのルールを書いたユーザが意図したようには動かないことが良くあります。
この動作はバグではなく、仕様なのです。

サービスの転送をきちんと動作させる方法については、サービスのリダイレクトに関する FAQ や natd(8) のマニュアルページ、Ports Collection にいくつか含まれているポート転送ユーティリティなどをご覧になると良いでしょう。

9.21. サービス要求を他のマシンにリダイレクトするには?

FTP　などのサービスのリクエストは、"socket"　パッケージを利用してリダイレクトできます。　"socket" パッケージは　ports　の　sysutils　カテゴリに含まれています。　(/etc/inet.confに書かれている) コマンド行を、次のように "socket" を呼ぶように変更してください。

```
ftp stream tcp nowait nobody /usr/local/bin/socket socket ftp.foo.com ftp
```

ここで *ftp.foo.com* はリダイレクト先のホスト名、行の最後の *ftp* はポート名です。

9.22. バンド幅の管理を行なえるツールはどこで手に入れられますか?

FreeBSD 用のバンド幅管理ツールには、無料で手に入れられる ALTQ と、 Emerging Technologies から入手できる Bandwidth Manager という市販のものの 2 種類があります。

9.23. BIND (named) が、53 番ポートのほかに大きな番号のポートで受け付けています。私のホストは乗っ取られたのでしょうか。

おそらく違います。FreeBSD 3.0 以降では、外向けの問合せにランダムな大きな番号のポートを用いるバージョンの BIND を用いています。ファイアウォールを通すため、またはあなたの 気分で、外向きの問合せを 53 番ポートから行いたいならば、 /etc/namedb/named.conf に次のように 設定してみてください。

```
options {
      query-source address * port 53;
};
```

更に限定したければ、* を単一の IP アドレスに置き換えることもできます。

それはともかく、おめでとうございます。 sockstat の出力を見て、おかしな現象に注目するのはよい習慣です。

9.24. なぜ /dev/bpf0: device not configured が出るのでしょうか?

バークレーパケットフィルタ (bpf(4))
ドライバは、それを利用するプログラムを実行する前に有効にしておく必要があります。
カーネルコンフィグファイルに、次のように追加してカーネルの再構築をしてください。

```
pseudo-device bpfilter      # Berkeley Packet Filter
```

そして再起動してから、次にデバイスノードを作成する必要があります。
これは、次のように入力し、/dev を変更することで行ないます。

```
# sh MAKEDEV bpf0
```

デバイスノードの作成の詳細は、 FreeBSD ハンドブックの「デバイスノード」を参照してください。

9.25. Linux の smbmount のように、 ネットワーク上の Windows マシンのディスクをマウントするにはどうしたら良いのでしょう?

Ports Collection に含まれる sharity light パッケージを使ってください、

9.26. icmp-response bandwidth limit 300/200 pps というメッセージがログファイルに現れるのですが、どういうことでしょう?

これは、カーネル自身から「ICMP や TCP のリセット (RST) 応答を、妥当な数よりも多く送っている」ということを、 あなたに伝えるメッセージです。 ICMP 応答は良く、使われていない UDP ポートに接続しようとした結果として生成されます。 また、TCP リセットはオープンされていない TCP ポートに接続しようとした結果として生成されます。 その他、これらのメッセージが表示される原因となる状況として、 以下のようなものがあります。

- (特定のセキュリティ上の弱点を悪用しようとする攻撃ではなく)
  膨大な数のパケットを使った強引なサービス妨害 (DoS) 攻撃。
- (一部のウェルノウンポートを狙ったものではなく)
  非常に広い範囲のポートに接続を試みるポートスキャン。

メッセージ中の最初の数字は、
上限を設定しなかった場合にカーネルが送っていたであろうパケットの数を示し、
二番目の数字は、パケット数の上限値を示します。 この上限値は `net.inet.icmp.icmplim` という sysctl 変数を使うことで、以下のように変更可能です。 ここでは上限を 1 秒あたりのパケット数で `300` にしています。

```
# sysctl -w net.inet.icmp.icmplim=300
```

カーネルの応答制限を無効にせず、 ログファイル中のメッセージだけを抑制したい場合、`net.inet.icmp.icmplim_output` sysctl 変数を次のようにすることで出力を止めることができます。

```
# sysctl -w net.inet.icmp.icmplim_output=0
```

最後に、もし応答制限を無効にしたい場合は、 `net.inet.icmp.icmplim` sysctl 変数に (上の例のようにして) `0` を設定することで実現できます。
ただし応答制限を無効化するのは、上記の理由からおすすめしません。

Chapter 10. PPP

10.1. ppp が動きません。どこを間違えているのでしょう?

まず ppp(8) のマニュアルと、 FreeBSD ハンドブックの「PPP」を読んでみましょう。 次に、

```
set log Phase Chat Connect Carrier lcp ipcp ccp command
```

という命令を ppp のコマンドプロンプトに対して打ち込むか、 設定ファイル /etc/ppp/ppp.conf に加えて (`default` セクションの先頭に加えるのが一番良いでしょう) ログを有効にしてみてください。 その際、 /etc/syslog.conf (syslog.conf(5) 参照) に

```
!ppp
*.*             /var/log/ppp.log
```

と書かれた行が含まれているか、また、 /var/log/ppp.log が存在しているかどうか確かめておいてください。 さて、これで何が起きているのか突き止めるために、ログファイルからたくさんの情報を得られるようになりました。
ログに訳の分らない部分があっても心配ご無用。 あなたが助けを求めた誰かにとっては、その部分が意味をなす場合があるのです。



ログの取得に syslog を使用するようになったのは 2.2.5 以降からです。

使用中の `ppp` のバージョンで “set log” 命令を解釈しない場合は、最新版をダウンロードすべきです。 FreeBSD の 2.1.5 以降でビルドできます。

10.2. ppp を実行するとハングします

ホスト名の解決がうまくいっていないのでしょう。まず、 リゾルバ (resolver) が /etc/hostsを参照するように、 /etc/host.conf の最初の行に `host` と書き込んでください。 つぎに、/etc/hosts に使用しているマシンのエントリを書き加えます。 ローカルでネットワークを使用していない場合は、 `localhost` の行を以下のように変更してください。

```
127.0.0.1       foo.bar.com foo localhost
```

使用しているホストのエントリを追加してもかまいません。
詳細は関連するマンページを参照してください。

10.3. ppp が -auto モードでダイアルしてくれない

まず最初に、デフォルトルートが確立しているかどうかチェックしてください。 `netstat -rn` (netstat(1) 参照) を実行すると、以下のような情報が表示されるはずです。

```
Destination        Gateway           Flags     Refs     Use     Netif Expire
```

```
default            10.0.0.2           UGSc        0        0      tun0
10.0.0.2           10.0.0.1           UH          0        0      tun0
```

これはあなたがハンドブックやマニュアル、 ppp.conf.sample の中で出てくるアドレスを使用していると仮定した場合の例です。
デフォルトルートが確立していない場合、 ppp.conf の中の HISADDR が理解できない、 古いバージョンの ppp(8) が走っている可能性があります。 FreeBSD 2.2.5 より前のバージョンに付属していた ppp を使用している場合、

```
add 0 0 HISADDR
```

と書かれた行を以下のように修正してください。

```
add 0 0 10.0.0.2
```

netstat -rn でデフォルトルートの情報が表示されない場合、もう一つ、 /etc/rc.conf (rc.conf(5) 参照) (2.2.2 より前のリリースでは /etc/sysconfig と呼ばれていました) の中でデフォルトのルータを誤って設定し、 ppp.conf から

```
delete ALL
```

の行をうっかり消してしまった可能性があります。 この場合は、 FreeBSD ハンドブックの「システムの最終設定」の項を読み直してください。

10.4. No route to host とはどういう意味ですか?

このエラーは通常、 /etc/ppp/ppp.linkup に以下のようなセクションが無い場合に起こります。

```
MYADDR:
delete ALL
add 0 0 HISADDR
```

これは動的 IP アドレスを使用している場合、 またはゲートウェイのアドレスを知らない場合にのみ必要な設定です。
インタラクティブモードを使用している場合、 パケットモードに入った後で (プロンプトが PPP と大文字に変わったらパケットモードに入ったしるしです)、 以下の命令を入力してください。

```
delete ALL
add 0 0 HISADDR
```

詳しい情報については、 FreeBSD ハンドブックの「PPP と動的 IP 設定」の項を参照してください。

10.5. 3 分ほど経つと接続が切れてしまう

ppp のタイムアウトは デフォルトでは 3 分です。 これは

```
set timeout NNN
```

という命令によって調整することができます。 *NNN* には、接続が切れるまでのアイドル時間が秒数で入ります。 *NNN* が 0 の場合、タイムアウトによる切断は起こりません。 このコマンドは ppp.conf に入れることも、インタラクティブモードでプロンプトから入力することも できます。 ソケットを用いる telnet(1) か pppctl(8) を使用し、 ppp サーバに接続することによって、回線がアクティブな間に限定してタイムアウトの時間を調整することも可能です。



pppctl は 2.2.5R からです。

詳しい情報は ppp(8) のマニュアルページを参照してください。

10.6. 負荷が高いと接続が切れてしまう

Link Quality Reporting (LQR) の設定を行っている場合、 マシンと接続先の間で非常にたくさんの LQR パケットが失われている可能性があります。結果として ppp は回線の具合いが悪いと考え、回線を切断するのです。2.2.5 より前のバージョンの FreeBSD では LQR はデフォルトで有効になっています。 現在ではデフォルトの状態で無効です。 LQR は以下の命令で無効にすることができます。

```
disable lqr
```

10.7. 接続がランダムに切れてしまう

ノイズの多い回線、あるいは待ち機能付きの回線では、 時々モデムが (誤って) キャリアを失ったと思い込み、 回線が切断されてしまうことがあります。

大多数のモデムでは、 一時的なキャリアの喪失をどれくらいの時間で検出するかを、設定で決めることができます。 たとえば USR Sportster では、S10 レジスタ の値を 10 倍した秒数がその値になります。 この場合、モデムをもっとのんびり屋さんにするには、 dial 行に次のような文字列を加えると良いでしょう。

```
set dial "...... ATS10=10 OK ......"
```

詳しくはお使いのモデムのマニュアルをご覧ください。

10.8. 接続が不規則にハングアップしてしまう

たくさんの人が、原因不明のハングアップを経験しています。
検証のために必要なのは、まずどちら側のリンクでそれが起こっているか、 ということです。

外部接続型モデムを利用しているなら、 単に ping を使うことで、 データを送信するときに TD ランプが点灯するかどうかを確認することができます。 もし、TD ランプが点灯して、 RD ランプが点灯しなければ、 問題は回線の向こう側にあります。TD が点灯しなければ、 問題は回線のこちら側です。内蔵型モデムの場合、 ppp.conf ファイルに set server コマンドを入れる必要があるでしょう。 回線が切断されたとき、pppctl を使って ppp に接続してください。 そのとき、 ネットワーク接続が急に復旧 (診断ソケットへのアクセスで、 ppp が復活します) するか、 もしくは接続自体が全くできない (ただし、 ppp 起動時に set socket コマンドがちゃんと実行されているとします) としたら、 問題は回線のこちら側です。
もし、接続可能で、かつ状況が変化しなければ、 set log local async を使ってローカル非同期ログ (async logging) を有効にし、 ping を他のウィンドウかターミナルから使ってください。
非同期ログには、こちら側のリンクの送受信データが記録されます。
もし、データが送信されたにもかかわらず返って来ていなければ、
問題は回線の向こう側にあることになります。

問題が回線のどちら側かにあることが分かったら、 つぎの二つの可能性が考えられるでしょう。

10.9. 回線の向こう側での反応がない

これに対処できることはほとんどありません。大部分の ISP は、Microsoft 社製 OS 以外の利用者に対してのサポートを拒否するでしょう。 ppp.conf ファイルの中に enable lqr を記述することで ppp が回線の向こう側で発生する切断を検出することができますが、この検出は比較的遅いため、あまり役に立ちません。また、あなたは user-ppp を利用していることを ISP に知られたくないと思うかも知れませんね。

まず最初に、こちら側の圧縮機能をすべて無効にしてみてください。
それには、設定ファイルをつぎのようにします。

```
disable pred1 deflate deflate24 protocomp acfcomp shortseq vj
deny pred1 deflate deflate24 protocomp acfcomp shortseq vj
```

そして再接続し、変更前と同じように通信できることを確認します。
もしこれによって状況が改善されるか、完全に解決したら、 (上の設定のうち) どの設定で状況が変化したのかを、 色々な組合せで試してみてください。これは、ISP に問い合わせを行なうときの有効な情報となります (ただし、 あなたが Microsoft 社製品以外のものを利用していることも明らかにしてしまいますが)。

ISP に問い合わせを行なう前に、こちら側の非同期ログを有効にして、
接続がハングアップするまで待ってください。この作業は、
非常に多くのディスク空間を消費するかも知れません。
興味の対象となっているのは、通信ポートから最後に読み込まれたデータです。 それは通常 ASCII データで、 問題点の詳細 (“Memory fault, core dump” など) が 記載されている可能性があります。

回線の向こう側で通信ログを監視することは可能なはずですので、 切断が発生した時、ISP の対応が好意的ならば どうして ISP 側で問題が発生したのかこちらに伝えてくれるかも知れません。
brian@Awfulhak.org まで詳細を送って頂くか、ISP に直接私に連絡するように伝えて下さっても構いません。

10.10. ppp がハングアップする

ベストな方法は、 CFLAGS+=-g と STRIP= を ppp の Makefile に追加して、 ppp を再構築し、 そして make clean && make && make install を行なうことです。 ppp がハングアップした時、 ps ajxww | fgrep ppp を使って ppp のプロセス ID を調べ、 gdb ppp PID を実行してください。 gdb のプロンプトから、 bt を使ってスタックをトレースすることができます。

スタックトレースの結果は、brian@Awfulhak.org まで送ってください。

10.11. Login OK! のメッセージが出た後、何も起こらない

2.2.5 より前のリリースの FreeBSD では、 ppp(8) はリンクが確立した後、接続先が Line Control Protocol (LCP) を発信するのを待ちます。しかし、多くの ISP ではネゴシエーションを自分からは起こさず、 クライアントが起こすのを待っています。 ppp に強制的に LCP を発信させるには、 次の命令を使います。

```
set openmode active
```



両方の側がネゴジェーションを起こしても、 大抵の場合は何の問題もありません。 ですから、現在では openmode はデフォルトで有効になっています。 次のセクションでこれが問題になる場合を説明します。

10.12. でもまだ magic is the same というエラーが出る

時折、接続直後のログに “magic is the same” というメッセージがあらわれることがあります。 このメッセージがあらわれても何も起きない場合もありますし、
どちらかの側が接続を切ってしまう場合もあります。 ppp の実装の多くはこの問題に対応できておらず、 その場合にはちゃんと link が上がっている状態であっても、 ppp が最終的にあきらめてしまい、 接続を切るまで設定のリクエストが繰り返し送られ、
設定が行われたという通知がログファイルに残ると思います。

これは通常、 ディスクアクセスの遅いサーバマシンのシリアルポートで getty が生きていて、 ppp がログインスクリプトか、 ログイン直後に起動されたプログラムから実行されている場合に起こります。 slirp を使用している場合に同様の症状が見られたという報告もあります。 原因は getty の終了されるまでと、 ppp が実行され、 クライアント側の ppp が Line Control Protocol (LCP) を送り始めるまでのタイミングにあります。 サーバ側のシリアルポートで ECHO が有効なままになっているので、 クライアント側の ppp にパケットが「反射」してしまうのです。

LCP ネゴシエーションの一部として、 リンクの両サイドで magic number を定めて、 「反射」が起きていないかどうか確かめる作業があります。 規約では、接続相手がこちらと同じ magic number を提示してきたら、 NAK を送って新しい magic number を選択しなければならないと定めています。 この作業の間、サーバのシリアルポートの ECHO がずっと有効になったままなので、 クライアント側の ppp は LCP パケットを送り、 パケットが反射して全く同じ magic number が送られてくるのを見つけ、 それに対して NAK を送るのです。一方 NAK 自体も (これは ppp が magic number を変更しなければいけないことを意味しています) 反射してくるので、 結果として magic number が数えきれないほど変更され、 そのすべてがサーバの tty バッファの中に積み重なることになるのです。

サーバでスタートした ppp は、すぐに magic number であふれかえってしまい、 LCP のネゴシエーションを十分に行ったものと判断して、 さっさと接続を切ってしまいます。 一方、クライアント側は反射が帰ってこなくなったので満足しますが、
それもサーバが接続を切ったことを知るまでです。

この事態は、以下の行を ppp.conf の中に書いて、
相手がネゴシエーションを開始できるようにする事によって回避できます。

```
set openmode passive
```

これで ppp はサーバが LCP ネゴシエーションを起こすのを待つようになります。 しかし、自分からは決してネゴジェーションを起こさないサーバもあるかもしれません。
もしこの状況に遭遇した場合には、次のようにしてください。

```
set openmode active 3
```

これによって ppp は 3 秒間 passive モードを続けた後で、 LCP リクエストを送り始めます。この間に相手がリクエストを送り始めた場合には 3 秒間待たずにこのリクエストに即座に応答します。

10.13. 接続が切れるまで LCP のネゴシエーションが続くのですが。

現在の ppp は、まだ LCP や CCP、 IPCP の返事が、元のリクエストと連携してくれる機能がきちんと実装されていません。 その結果、ある ppp の実装が相手よりも 6 秒以上遅い場合には、 LCP 設定のリクエストをさらに 2 回送ります。これは致命的な物です。

A と Bという 2 つの実装を考えてみましょう。 A が接続の直後に LCP リクエストを送り、 一方 B の方はスタートするのに 7 秒かかったとします。B がスタートする時には A は LCP リクエストを 3 回送ってしまっています。 前の節で述べた magic number の問題が起きないよう、 ECHO は off になっていると考えています。 B は REQ を送ります。 するとこれは A の REQ のうち、最初の物に対する ACK となります。 結果として、A は OPENED の状態に入り、 B に対して (最初の) ACK を送ります。 そのうちに B は、B がスタートする前に A から送られたもう 2 つの REQ に対する ACK を送り返します。 B は A からの最初の ACK を受け取り OPENED の状態に入ります。 A は B からの 2 つ目の ACK を受け取りますので、 REQ-SENTの状態に戻り、 さらに、RFC のとおりに (4 つ目の) REQ を送ります。そして 3 つ目の ACK を受け取って OPENED 状態に入ります。 一方、B は A からの 4 つ目の REQ を受け取りますので、 ACK-SENT の状態に入り、2 つ目の REQ と 4 つ目の ACK を RFC のとおりに送ります。 Aは、 REQ を受けとると REQ-SENT の状態になり、さらに REQ を送ります。そしてすぐに ACK を受け取って OPENED の状態に入ります。

これが、片方の ppp があきらめてしまうまで続きます。

これを回避する最も良い方法は、 片方を passive モードに設定する、すなわち反対側がネゴシエーションを開始するまで待つようにする事です。 これは、

```
set openmode passive
```

というコマンドでできます。 このオプションは気を付けて使わないといけません。さらに

```
set stopped N
```

というコマンドを追加して、 ppp がネゴシエーションが開始するまで待つ最大の時間を設定してください。もしくは、

```
set openmode active N
```

というコマンド (ここで、 *N* はネゴシエーションが始まるまで待つ時間) を使うこともできます。詳しくはマニュアルページを参照してください。

10.14. ppp が接続直後に固まってしまう

2.2.5 より前のバージョンの FreeBSD では、ppp が Predictor1 圧縮のネゴシエーションを誤って解釈して、 接続直後にリンクを無効にしている可能性があります。これは両サイドが異なる Compression Control Protocols (CCP) を使ってネゴジェーションを行った場合にのみ発生します。
この問題は現在は解決していますが、あなたの走らせている ppp のバージョンが古い場合でも、次の命令で解決することができます。

```
disable pred1
```

10.15. ppp の内部でシェルを起動しようとすると固まってしまう

`shell` あるいは `!` コマンドを使用すると、 ppp はシェルを起動し (何か引数を渡した場合は、 ppp は引数も実行します)、 コマンドが終了するまで処理を中断します。 コマンドを実行中に ppp のリンクを使おうとすると、 リンクが固まっているように見えますが、 これは ppp がコマンドの終了を待っているからです。

このような場合は、代わりに `!bg` コマンドを使用してください。与えられたコマンドがバックグラウンドで実行されるので、 `ppp` はリンクに関するサービスを継続することができます。

10.16. ヌルモデムケーブルを使用しているとき、 ppp が終了しない

ヌルモデムケーブルを使用して直接接続している場合、 ppp は自動的には接続の終了を知ることができません。
これはヌルモデムシリアルケーブルの配線に起因しています。 この種の接続形態を用いる場合は、以下の命令を用いて LQR を常に有効にする必要があります。

```
enable lqr
```

こうすると、接続先がネゴシエーションを行う場合、デフォルトで LQR の使用を受け入れるようになります。

10.17. ppp を -auto モードで動かすと、勝手にダイアルすることがある

ppp が思いもしないときにダイアルを始める場合、その原因を突き止め、 防止のためにダイヤルフィルタ (dfilters) をかけてやる 必要があります。

原因を突き止めるためには、以下の命令を使用してください。

```
set log +tcp/ip
```

これで接続を通過するすべてのトラフィックをログに残すことができるようになりました。
次に突然回線がつながったときのログのタイムスタンプをたどれば、
原因を突き止めることができるはずです。

原因がわかったら、次に、このような状況ではダイヤルが起こらないようにしましょう。
通常、この手の問題は、DNS で名前の解決をしようとしたために起こります。 DNS による名前の解決によって、 接続が行われるのを防止するには、 次のような手段を用います (これは ppp の既に確立した接続に関してパケットのフィルタリングをするものではありません)。

```
set dfilter 1 deny udp src eq 53
set dfilter 2 deny udp dst eq 53
set dfilter 3 permit 0/0 0/0
```

これはデマンドダイヤル機能に問題を生じさせるため、 常に適切であるとはかぎりません。
ほとんどのプログラムは他のネットワーク関連の処理を行なう前に DNS への問い合わせが必要になります。

DNS の場合は、 何が実際にホスト名を検索しようとしているのかを突き止めるべきでしょう。
大抵の場合は、 sendmail(8) が犯人です。 設定ファイルで sendmail が DNS に問い合わせないようになっているか確認すべきです。
自分用の設定ファイルを作成するための詳しい方法は、 メールの設定 の項をご覧ください。 または、 .mc ファイルに次のような行を追加してもよいでしょう。

```
define(`confDELIVERY_MODE', `d')dnl
```

この行を追加すると、sendmail はメールキューを処理する (通常 sendmail は 30 分ごとにキューを処理するよう、 “-bd -q30m” というオプションを付けて起動されます) までか、 または (多分 ppp.linkup というファイルの中で) “sendmail -q” というコマンドが実行されるまで、すべてのメールをキューに溜めるようになります。



訳注

“sendmail -q” はその時点のメールキューの内容を処理して終了します。

10.18. CCP エラーとはどういう意味ですか

ログファイル中の以下のエラーは、

```
CCP: CcpSendConfigReq
CCP: Received Terminate Ack (1) state = Req-Sent (6)
```

のネゴシエーションにおいて ppp は Predictor1 圧縮を用いるべく主張したのに対して、接続先は圧縮を使用しないことを主張した場合に起こります。
このメッセージには何の害もありませんが、 出るのが嫌なら、以下の命令を用いてこちら側でも Predictor1 圧縮を無効にすることで対応できます。

```
disable pred1
```

10.19. ファイル転送の途中で、ppp が IO エラーを出して固まってしまう

FreeBSD 2.2.2 以前のバージョンの tun ドライバには、tun インタフェースの MTU のサイズより大きなパケットを受け取ることができないというバグがありました。 MTU のサイズより大きなパケットを受け付けると IO エラーが起こり、 syslogd 経由で記録されるのです。

ppp の仕様では、 LCP のネゴシエーションを行う場合を含むどのような場合でも最低 1500 オクテットの Maximum Receive Unit (MRU) を受け入れる必要があります。 ですから、MTU を 1500 以下に設定した場合でも、ISP はそれに関係なく 1500 の大きさのパケットを送ってくるでしょう。そしてこのイケてない機能にぶちあたって、 リンクが固まるのを目にすることになるのです。

FreeBSD 2.2.2 以前のバージョンでは、MTU を決して 1500 より小さくしないことで、この問題を回避することができます。

10.20. どうして ppp は接続速度をログに残さないんでしょう?

モデムとの「やり取り」すべての行をログに残すには、
以下のようにして接続速度のログの有効化を行ってください。

```
set log +connect
```

これは ppp(8) に最後にくることが要求されている “expect” という文字列がくるまでのすべてのものをログに記録させます。

接続速度はログにとりたいけれど、PAP や CHAP を使っている (その結果、ダイヤルスクリプト中の CONNECT 以降に全く「やりとり」を行わない - “set login” スクリプトには何も書かない) のであれば、 ppp に “expect” を含んだ CONNECT 行すべてがくるまで待たせるようにしないといけません、以下のようになります。

```
set dial "ABORT BUSY ABORT NO\\sCARRIER TIMEOUT 4 \"\" ATZ OK-ATZ-OK ATDT\\T TIMEOUT
60 CONNECT \\c \\n"
```

ここで、CONNECT を受信してから、 何も送らず、復帰改行 (linefeed) を待っています、 ppp に CONNECT の応答すべてを読み込ませているわけです。

10.21. 私の chat スクリプトでは \ という文字を PPP が解釈してくれません。

PPP は設定ファイルを読み込むときに、 set phone "123 456 789" のような文字列を正しく解釈し、番号が実際に*1* つの引数であると理解します。 """ という文字を指定するには、バックスラッシュ (backslash; “\”) でエスケープしなければなりません。

chat の各引数が解釈されるときには、 “\P” や “\T” のような特別なエスケープシーケンス (マニュアルページ参照のこと) を見付けるために、 もう 1 回、字句解析を行います。 このように字句解析は 2 回繰り返されますので、 正しい回数だけエスケープ処理を行わないといけません。

モデムにたとえば “\” のような文字を送りたい場合には、 次のようにする必要があります。

```
set dial "\"\" ATZ OK-ATZ-OK AT\\\\X OK"
```

実際にモデムに送られる文字列は次のようになります。

```
ATZ
OK
AT\X
OK
```

他の例ですと

```
set phone 1234567
          set dial "\"\" ATZ OK ATDT\\T"
```

は次のようになります。

```
ATZ
OK
ATDT1234567
```

10.22. ppp が segmentation fault になるのですが、ppp.core ファイルがありません

ppp (や他のプログラム) は決して core を吐いてはいけません。 ppp は実効 uid が 0

で動いていますので、 オペレーティングシステムは ppp を終了させる前にディスクに core イメージを書き込みません。 しかし ppp は実際にはセグメンテーション違反や、 他の core を吐く原因となるようなシグナルによって終了しており、 さらに最新のバージョン (このセクションの始めを見てください) を使用しているならば、次のようにしてください。

```
% tar xfz ppp-*.src.tar.gz
% cd ppp*/ppp
% echo STRIP= >>Makefile
% echo CFLAGS+=-g >>Makefile
% make clean all
% su
# make install
# chmod 555 /usr/sbin/ppp
```

これでデバッグ可能なバージョンの ppp がインストールされます。 root で ppp を実行し、すべての特権が無効になっているようにする必要があるでしょう。 ppp を実行する時には、カレントディレクトリが make したディレクトリであるようにしてください。

これで、ppp がセグメンテーション例外を受け取ったときには ppp.core という名前の core ファイルを吐くようになります。core が 吐かれたら次のようにしてください。

```
% su
# gdb /usr/sbin/ppp ppp.core
(gdb) bt
..
(gdb) f 0
..
(gdb) i args
..
(gdb) l
..
```

質問する際には、これらすべての情報を提供して、 問題点の分析ができるようにしてください。

gdb の使い方に慣れている場合には、実際に dump の原因となった理由やそのアドレス、関連した変数の値なども調べる事ができるでしょう。

10.23. auto モードでダイアルをするようなプロセスが接続されない。

これは ppp がローカル側の IP アドレスを、動的に通信相手と交渉するように設定されている時に発生する良く知られた障害でした。最新のバージョンでは、 この問題は修正されています。 iface をマニュアルページから検索してみてください。

これは、最初のプログラムが connect(2) を呼び出した時、tun インターフェイスの IP アドレスが、ソケットの終端に割り当てられてしまうという問題です。 カーネルは、外へ出ていく最初のパケットを作り、それを tun デバイスへ書き込みます。 そして ppp は、

そのパケットを読み込んで接続を確立します。 ppp は動的に IP アドレスを割り当てるため、もしインターフェイスのアドレスが変化してしまうと、 最初に割り当てられたソケット終端の IP アドレスは無効になってしまいます。 そのため、それ以降相手に送られるすべてのパケットは通常、相手に届くことはないでしょう。もし仮に届いたとしても、 既にこちらの IP アドレスは変更されているので、 どんな反応も最初のマシンには戻ってきません。

この問題に対処する理論的な方法がいくつかあります。もし可能なら、 相手が再度、同じ IP アドレスを割り当ててくれることが一番です :-)ppp の現在のバージョンはこれを行ないますが、他のほとんどの実装はそういった動作をしません。

我々の側から対処できる最も簡単な方法は、tun インターフェイスの IP アドレスを固定する事です。またそのかわりに、 外に出ていくパケットを変更して、 発信元 IP アドレスをインターフェイスの IP アドレスから、交渉によって得られた IP アドレスに、適宜書きかえる事によっても対処できます。 これは、基本的に ppp の最新バージョンにある iface-alias オプションが行なっていることと同じです (libalias(3) および、ppp の -nat スイッチにも関係します)。それは、以前の IP アドレスをすべて管理し、 それらを最後の交渉によって得られた IP アドレスに対して NAT 機能を有効化します。

もう 1 つの (おそらく最も信頼できる) 方法は、bind された すべてのソケットの IP アドレスを、異なるものに変更できるシステムコールを実装することです。 pppは、 新しい IP アドレスが割り当てられた時、 このシステムコールを用いて実行されているプログラムにある、すべてのソケットを書きかえてやるわけです。 同じシステムコールが、DHCP クライアントが利用するソケットを 強制的に再 bind するのにも使うことができるでしょう。

3 つ目の方法は、IP アドレスを指定しないでインターフェイスを利用できるようにすることです。外に出ていくパケットは、最初の SIOCAIFADDR ioctl の完了まで、 255.255.255.255 という IP アドレスが与えられます。 これによって、ソケットは常に bind することができます。 発信元 IP アドレスを変更するのは ppp の仕事です。ただし、 それは発信元 IP アドレスが 255.255.255.255 になっていて、IP アドレスと IP チェックサムを変更する必要がある場合だけです。これは、カーネルが不適切に設定されたインターフェイスへは
異常なパケットを送出しようとすることを利用して、なにか他の
仕組みが遡及的に修正を行ってくれることを前提にしている、 割り切った方法ではあります。

10.24. 何故ほとんどのゲームが -nat スイッチ付きだと動かないんですか?

libalias を使っている時にゲームなどの類のものが動作しない理由は、外側にあるマシンが接続しようとしているか、内側にあるマシンに (余計な) UDP パケットを送信しようとしているからです。 内側のマシンにこれらのパケットを送るべきかについて、NAT ソフトウェアは関知しません。

うまく動かすためには、 実行中のものが問題の発生しているソフトウェアだけであるかを確認し、ゲートウェイの tun インタフェースに対して tcpdump を実行するか、 ゲートウェイ上で ppp の TCP/IP ログ記録を有効化 (“set log +tcp/ip”) してください。

行儀の悪いソフトウェアを起動する際に、 ゲートウェイマシンを通過するパケットを監視すべきです。外側から何かパケットが戻ってきた時に、 そのパケットは破棄されるでしょう (それが問題なのです)。これらのパケットのポート番号に注意して、 その行儀の悪いソフトウェアを停止してください。

これを数回繰り返してポート番号が常に同じであるかを確認してみてください。 同じであった場合は、/etc/ppp/ppp.conf の適切なセクションに次の行を入れると、そのソフトウェアは動作するようになるでしょう。

```
nat port proto internalmachine:port port
```

ここで *proto* は `tcp` か `udp` であり、 *internalmachine* はパケットを送りたいマシン、そして *port* はパケットの送信先のポート番号です。

上記のコマンドを変更せずに、
他のマシン上でそのソフトウェアを使用できるようにはしたくないかもしれません。
そして同時に二つの内部のマシン上でそのソフトウェアを実行することは、
この質問の範囲を超えています。結局、外側の世界からは、
内部ネットワーク全体がただ一つのマシンとして見えるのです。

ポート番号が常に同じとは限らない場合、さらに三つのオプションがあります。

1. libalias でサポートするようにし、結果を送り付ける。 特定の場合の例は /usr/src/lib/libalias/alias_*.c にあります (alias_ftp.c は良いプロトタイプです)。これには通常、外向きの特定のパケットを読み、
   内部の計算機のある特定のポートへの接続を開始するような命令が、
   外部の計算機対して送られていることを見分け、
   後続のパケットがどこに行けばいいのかが分かるように、 エイリアステーブル中の "*route*" の部分を設定する、という作業が含まれます。

   これは最も難しい方法ですが、最も良い方法でもありますし、ソフトウェアが
   複数の計算機で動くようにできます。

2. プロキシ (proxy) を使う。アプリケーションが、たとえば socks5 をサポートしているか、(cvsup のように) "passive" オプションを持っているとこの方法が使えます。 "passive" とは相手側のほうから接続を求めてくることを避けるためにあるオプションです。

3. “nat addr” を使ってなんでもかんでも内部の計算機に向けて流してしまう。
   これはちょっと無理矢理な解決法です。

10.25. 有用なポート番号のリストはありませんか?

まだ出来ていません。しかし、 これは (関心を持って頂けるならば) そういったリストにしていく予定です。 それぞれの例にある *internal* は、 ゲームで遊ぶマシンの IP アドレスに置き換えてください。

Asheron’s Call

```
nat port udp internal:65000 65000
```

手動でゲームのポート番号を 65000 に変更してください。 マシンが複数ある場合は、それぞれのマシンに重複しないポート番号 (つまり 65001、65002 など) を設定し、その設定ごとに `nat port` の行を追加します。

Half Life

```
nat port udp internal:27005 27015
```

PCAnywhere 8.0

```
nat port udp internal:5632 5632
nat port tcp internal:5631 5631
```

Quake

```
nat port udp internal:6112 6112
```

このように設定する代わりに、 www.battle.net で Quake のプロキシ (proxy) がサポートされているか調べてもいいでしょう。

Quake2

```
alias port udp internal:27901 27910
```

Red Alert

```
nat port udp internal:8675 8675
nat port udp internal:5009 5009
```

10.26. FCS エラーって何?

FCS とは Frame Check Sequence (フレームチェックシーケンス) の略です。 個々の ppp パケットには、送受信するデータが正しいかを調べるためのチェックサムが含まれています。 受信したパケットの FCS が正しくない場合は、そのパケットは廃棄され、 HDLCFCS カウントが増やされます。 HDLC エラーの数は、 `show hdlc` コマンドを使って表示できます。

リンクの品質が悪かったり、 シリアルドライバがパケットを取りこぼしていたりすると、 FCS エラーがたびたび発生します。 FCS エラーは、 圧縮プロトコルの速度低下の原因にはなりますが、特に心配する必要はありません。 外付けモデムを使っている場合は、ケーブルがちゃんとシールドされているかを確認してください。 そうでない場合、 FCS エラーの原因となる場合があります。

接続直後からリンクがフリーズし、大量の FCS エラーが発生する場合は、 リンクが 8 ビットクリーンでない可能性があります。 ソフトウェアフロー制御 (XON/XOFF) が使われていないことを確認してください。
どうしてもソフトウェアフロー制御を使わなければならない場合は、 `set accmap 0x000a0000` コマンドを使用して、 ppp に `^Q` と `^S` をエスケープさせてください。

リモートホストが PPP プロトコルを使用してない場合も、大量の FCS エラーが発生します。
この場合はログをとりながら非同期で接続し、
ログインプロンプトやシェルプロンプトが送られて来ていないか確認してください。

ログファイルにリンクを終了した原因となるような記録がない場合は、 リモートホスト (プロバイダ?) の管理者に、 セッションを終了された理由を尋ねてください。

10.27. ゲートウェイで PPPoE を実行すると MacOS や Windows 98 との接続がフリーズしてしまうのですが、これはなぜなのでしょうか?

Michael Wozniak mwozniak@netcom.ca 氏が、この現象に関して説明してくれました。 また、Dan Flemming danflemming@mac.com 氏は MacOS での解決策を提供してくれました。
情報の提供に感謝します。

これは、いわゆる「ブラックホールルータ (Black Hole router)」に原因があります。 Windows 98 と MacOS (および、おそらく他の Microsoft 社製 OS) の TCP パケット送出は、 PPPoE のフレーム (Ethernet の MTU は標準で 1500) に入らないような大きなセグメントサイズを要求します。 そしてさらに分割禁止 ("don't fragment") フラグビットを (TCP パケットにデフォルトで) セットするのですが、 Telco のルータは、分割が必須 ("must fragment") であることを示す ICMP メッセージを、接続しようとするウェブサイトに対して送出しません (つまり、ルータは正しく ICMP パケットを送出しているのですが、 ウェブサイトのファイアウォールがそれを落としているのです)。 そのためウェブサーバが PPPoE 接続に対して大きすぎるフレームを送出すると Telco のルータはそのフレームを捨ててしまい、 見ようとしたページが表示されないという症状が現われます (MSS より小さいページや画像は表示されます)。 ほとんどの Telco PPPoE 設定は、標準でこのように設定されているようです。
(ああ、彼らがルーティングプログラムの作り方を理解してさえいれば...)。

一つの解決法は、Windows 95/98 マシンで regedit を使い、
次のレジストリエントリを追加することです。

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Class\NetTrans\0000\MaxMTU
```

レジストリエントリは、"1450" の値 (もっと正確に言うと、TCP パケットを PPPoE フレームに完全に適合させるには "1464" であるべきでですが、 "1450" とすると、現われる可能性がある他の IP プロトコルに対してエラーマージンを確保することができます) にする必要があります。 このレジストリキーは、Windows2000 で `Tcpip\Parameters\Interfaces\ID for adapter\MTU` に移されたという報告がありました。

FreeBSD/NAT/PPPoE ルータと共存させるために Windoze の MTU を変更する方法に関する詳細は、Microsoft Knowledge Base にある、 番号 "Q158474 - Windows TCPIP Registry Entries"、 および番号 "Q120642 - TCPIP & NBT Configuration Parameters for Windows NT" を参照してください。

残念なことに、MacOS には TCP/IP 設定を変更する方法がありません。 しかし、Sustainable Softworks 社 が販売している OTAdvancedTuner (OT は OpenTransport という MacOS の TCP/IP スタックの名前のこと) のような商用ソフトウェアが存在します。 このソフトウェアは、ユーザから TCP/IP 設定の変更を行なうことを可能にします。 MacOS NAT ユーザはドロップダウンメニューから

ip_interface_MTU を選択し、 ボックスにある 1500 の代わりに 1450 を入力し、 Save as Auto Configure の隣のボックスをクリックして Make Active をクリックする必要があります。

ppp の最新版 (2.3 かそれ以降) には、自動的に MSS を適切な値に調節する enable tcpmssfixup コマンドがあります。 この機能は標準で有効になっています。 もし旧バージョンの ppp を使わなければならない状況にあるなら、 tcpmssd の port をご覧になると良いでしょう。

10.28. どれにも当てはまらない! どうしたらいいの?

これまでのすべての質問に当てはまらない場合、設定ファイル、 ppp の実行方法、ログファイルの該当部分と netstat -rn コマンドの出力 (接続前と接続後) を含む、あなたの持っているすべての情報を FreeBSD general questions メーリングリスト や comp.unix.bsd.freebsd.misc ニュースグループへ送ってください。誰かがあなたを正しい方向へ導いてくれるでしょう。

Chapter 11. シリアル接続

このセクションでは、FreeBSD でシリアル接続をする時の一般的な質問に答えます。 PPP および SLIP については、 ネットワーキングのセクションを参照してください。

11.1. どうやったら FreeBSD がシリアルポートを認識したことを知る事ができますか?

FreeBSD のカーネルが起動する時、カーネルはその設定にしたがって、システムのシリアルポートを検出します。起動時に表示されるメッセージをよく観察するか、起動後に次のコマンドを実行する事によって確認できます。

```
dmesg | grep sio
```

ここに上に挙げたコマンドの出力例を示します。

```
sio0 at 0x3f8-0x3ff irq 4 on isa
sio0: type 16550A
sio1 at 0x2f8-0x2ff irq 3 on isa
sio1: type 16550A
```

これは、二つのシリアルポートを示しています。1 番目は、 irq が 4 で 0x3f8 のポートアドレスを使用しています。 そして、16550A-type UART チップが存在します。 2 番目は、同じチップを使っていますが、 irq は 3 で、0x2f8 のポートアドレスを使用しています。内蔵のモデムカードは、通常のシリアルポートと同じように扱われますが、常時シリアルポートにモデムが接続されているという点で異なります。

GENERIC カーネルは、上の例と同じ irq とポートアドレスの設定の二つのシリアルポートをサポートしています。これらの設定があなたのシステムに合わない場合、またはモデムカードを追加した場合やカーネルの設定以上にシリアルポートを持っている場合は、カーネルを再構築してください。 詳しくは、 カーネルの構築の項を参照してください。

11.2. どうやったら FreeBSD がモデムカードを認識したことを知ることができますか?

前の質問を参照してください。

11.3. FreeBSD 2.0.5 にアップグレードしたら tty0X が見つからなくなってしまったのですが

心配ありません。 ttydX に統合されました。ただ、古い設定ファイルのすべてを更新する必要があります。

11.4. どうやったら FreeBSD でシリアルポートにアクセスできますか?

3 番目のポート sio2 (sio(4) をご覧ください。DOS では、COM3 と呼ばれます。) には、ダイヤルアウトデバイスとしては /dev/cuaa2、 ダイヤルインデバイスとして /dev/ttyd2 があります。それではこの両者にはどのような違いがあるのでしょうか?

まず、ダイヤルインの時には ttydX を使います。 /dev/ttydX をブロッキングモードでオープンすると、プロセスは対応する cuaaX デバイスがインアクティブになるのを待ちます。 次に CD 信号がアクティブになるのを待ちます。 cuaaX デバイスをオープンすると、シリアルポートが ttydX デバイスによってすでに使われていないかどうかを確認します。
もしこのポートが使用可能であれば、ポートの使用権を ttydX から「奪い取る」のです。また、 cuaaX デバイスは CD 信号を監視しません。 この仕組みと自動応答モデムによって、リモートユーザーをログインさせたり、 同じモデムでダイヤルアウトしたりすることができ、システムのあらゆるトラブルの面倒を見ることができるでしょう。

11.5. マルチポートシリアルカードをサポートさせるにはどうしたらよいのでしょうか?

繰り返しになりますが、 カーネルコンフィグレーションのセクションでは、あなたのカーネルの設定についての情報が得られるでしょう。
マルチポートシリアルカードを使用するためには、カーネルの設定ファイルに、
カードの持つそれぞれのシリアルポートに対応する sio(4) の行を記述する必要があります。しかし、 irq とベクタアドレスは一つのエントリにのみ記述してください。 カード上のすべてのポートは一つの irq を共有しなければなりません。 一貫性を持たせるためにも、 最後のシリアルポートの所で irq を指定してください。 また、COM_MULTIPORT オプションも付けてください。

次に示す例は、AST の 4 ポートシリアルカードを irq 7 で設定したものです。

```
options "COM_MULTIPORT"
device sio4 at isa? port 0x2a0 tty flags 0x781
device sio5 at isa? port 0x2a8 tty flags 0x781
device sio6 at isa? port 0x2b0 tty flags 0x781
device sio7 at isa? port 0x2b8 tty flags 0x781 irq 7 vector siointr
```

このフラグはマスタポートがマイナー番号 7 (0x700) を持っていて、 検出時の診断機能を有効にし (0x080)、 そしてすべてのポートで irq を共有する (0x001) ということを意味しています。

11.6. FreeBSD で複数のマルチポートシリアルカード間で irq を共有することはできますか?

現在のところはできません。それぞれのカード毎に異なった irq を使ってください。

11.7. ポートにデフォルトのパラメータを設定する事は出来ますか?

ttydX デバイス (または cuaaX デバイス) は、アプリケーションのためにオープンする標準的なデバイスです。
プロセスがそのポートをオープンする時、 プロセスはデフォルトの端末 I/O 設定を取得します。
これらの設定は次のコマンドで確認することができます。

```
stty -a -f /dev/ttyd1
```

このデバイスに対する設定を変更した場合、 その設定はデバイスをクローズするまで有効です。
デバイスを再オープンした場合、それらの設定はデフォルトに戻ってしまいます。
デフォルトの設定に変更を加えるために、 「初期設定」デバイスをオープンし、設定を修正することができます。 たとえば、CLOCAL モード、8 ビット、 XON/XOFF フロー制御という設定を ttyd5 のデフォルトにしたい場合、次のように行なってください。

```
stty -f /dev/ttyid5 clocal cs8 ixon ixoff
```

この設定を行なうためのコマンドを記述するのに適切なファイルは、 /etc/rc.serial です。
これでアプリケーションが ttyd5 をオープンした時に、 これらの設定をデフォルトで取得します。
しかし、こういったリンクによる設定は変更可能です。

「設定固定」デバイスを調整してやることによって、
アプリケーションによる設定の変更を禁止することができます。 たとえば、ttyd5 の通信速度を 57600bps に固定するには、次のように行ってください。

```
# stty -f /dev/ttyld5 57600
```

これにより、アプリケーションは ttyd5 をオープンし、ポートの通信速度を変更しようとしますが、通信速度は 57600bps のままになります。

当然のことながら、初期設定デバイスおよび、設定固定デバイスは root のみが書き込みできるようになっていなければなりません。 しかし、MAKEDEV(8) スクリプトはデバイスエントリを作成する時に、 このような設定は行いません。

11.8. どのようにしたらモデム経由でダイヤルアップログインができるのでしょうか?

つまり、インターネットサービスプロバイダーになりたいのですね。 それにはまず、1 台ないし複数の自動応答モデムが必要です。 モデムには、キャリアーを検出した時には CD 信号を出力し、 そうでない場合には出力しないことが必要とされます。 また DTR 信号が on から off になった時には、 電話回線を切断し、モデム自身をリセットしなければなりません。 おそらく、RTS/CTS フロー制御を使うか、 ローカルフロー制御をまったく使わないかのどちらかでしょう。

最後に、コンピュータとモデムの間は固定速度でなければなりません。 ただ、
(ダイヤルアップの発呼者に対して親切であるためには、)
こちらのモデムと相手側のモデムの間の速度を、 モデム間で自動調整できるようにすべきでしょう。

多くあるヘイズコマンド互換モデムに対して、次のコマンドはこれらの設定を行ない、
その設定を不揮発性メモリーに保存します。

```
AT&C1&D3&K3&Q6S0=1&W
```

MS-DOS のターミナルプログラムに頼らずに AT コマンドを送出するには、 「AT コマンドを入力するには」のセクションを参照してください。

次に、モデム用のエントリを /etc/ttys (ttys(5) 参照) に作成しましょう。 このファイルには、
オペレーティングシステムがログインを待っているすべてのポートが記述されています。
以下のような行を追加してください。

```
ttyd1 "/usr/libexec/getty std.57600" dialup on insecure
```

この行は、2 番目のシリアルポート (/dev/ttyd1) には、 57600bps の通信速度でノンパリティ (std.57600: これは /etc/gettytab に記述されています。gettytab(5) 参照)
のモデムが接続されていることを示しています。 このポートの端末タイプは dialup です。
またこのポートは、on すなわちログイン可能であり、insecure です。 これは root
がこのポートから直接ログインするのは、 許可されていないということを意味します。
このようなダイヤルインポートに対しては、 ttydX のエントリを使用してください。

これが一般的な、ターミナルタイプとして dialup を使う方法です。多くのユーザーは、 .profile や .login で、 ログイン時の端末タイプが dialup であった場合には、
実際の端末タイプをユーザーに問い合わせるように設定しています。 この例は、ポートが insecure でした。このポートで root になるには、 一般ユーザーとしてログインし、それから "su" を使って root になってください。 もし、secure を指定したならば、 直接 root がそのポートからログインできます。

/etc/ttys に変更を加えた後は、HUP シグナル (SIGHUP) を init(8) プロセスに送る必要があります。

```
# kill -HUP 1
```

この操作は init プロセスに /etc/ttys を再読み込みさせます。 これにより、init プロセスは getty プロセスをすべての on となっているポートに起動させます。
次のようにして、ポートがログイン可能かを知ることができます。

```
% ps -ax | grep '[t]tyd1'
```

ログイン可能であれば、次のような出力が得られるはずです。

```
747 ??  I      0:00.04 /usr/libexec/getty std.57600 ttyd1
```

11.9. ダムターミナルを FreeBSD マシンに接続するにはどうしたらよいのでしょうか?

もし、他のコンピューターを FreeBSD の端末として接続したいのならば、お互いのシリアルポート間をつなぐヌルモデムケーブル (訳注: リバースケーブルもしくはクロスケーブルとも呼ばれます) を用意してください。もし、既製の端末を使う場合は、付属するマニュアルを参照してください。

そして、/etc/ttys (ttys(5) 参照) を上と同じように変更してください。 たとえば、WYSE-50 という端末を 5 番目のポートに接続するならば、 次のようなエントリを使用してください。

```
ttyd4 "/usr/libexec/getty std.38400" wyse50 on secure
```

この例は、/dev/ttyd4 ポートにノンパリティ、 端末タイプが wyse50、通信速度が 38400bps (`std.38400`: この設定は、 /etc/gettytab に記述されています。gettytab(5) 参照) の端末が存在しており、 `root` のログインが許可されている (`secure`) であることを示しています。

11.10. どうして tip や cu が動かないのですか?

おそらくあなたのシステムでは tip(1) や cu(1) は `uucp` ユーザーか、 `dialer` グループによってのみ実行可能なのでしょう。 `dialer` グループは、モデムやリモートシステムにアクセスするユーザーを管理するために、 使用することができます。それには、/etc/group ファイルの `dialer` グループにあなた自身を追加してください。

そうする代わりに、次のようにタイプすることにより、 あなたのシステムの全ユーザーが `tip` や `cu` を実行できるようになります。

```
# chmod 4511 /usr/bin/cu
# chmod 4511 /usr/bin/tip
```

11.11. 私の Hayes モデムはサポートされていないのですが、どうしたらいいのでしょうか。

実際、 tip(1) のオンラインマニュアルは古くなっています。 すでに、Hayes ダイアラが実装されています。 /etc/remote ファイル (remote(5) 参照) で、 “at=hayes” と指定してください。

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(1) を使う場合には (`ATX0&W`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` のダイヤ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=45&W` を実行してください。

実際、デフォルトの `tip` は Hayes の完全なサポートを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 HAYES1” と変更し、そして “make” と “make install” を実行します。これでうまく動作するでしょう。

11.12. これらの AT コマンドを入力するには?

/etc/remote ファイル (remote(5) 参照) 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 [.filename]#/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` (tip(1) 参照) 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
# ./MAKEDEV cuaa0
```

または `root` になって以下のように cu を使います。

```
# cu -lline -sspeed
```

line にはシリアルポート (たとえば /dev/cuaa0)を指定します。 そして *speed* には接続する速度 (たとえば `57600`) を指定します。 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。

11.13. 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/phones 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。
そこで、バックスラッシュを使ってエスケープを行います。

```
pn=\@
```

11.14. 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 ファイル (remote(5) 参照) に追加します。
たとえば、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 57600 bps:\
: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(1) より cu(1) 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。

11.15. 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(1) は 1200bps 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 1200bps を使わなければならないわけではありません。

11.16. 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のですが。

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。 たとえば、/etc/remote (remote(5) 参照) 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。

11.17. 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 (remote(5) 参照) 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/phones ファイル (phones(5) 参照) 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(1) は一連の電話番号を上から順に試みて、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。

11.18. CTRL+P を 1 回送るために 2 度押す必要があるのはなぜ?

CTRL+P は通常「強制 (force)」文字であり、 tip(1) に次の文字がリテラルデータであることを伝えます。強制文字は「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。いくつかのターミナルサーバで使われているのを見ただけですが、 <*single-char*> に SHIFT+CTRL+6 に割り当てるのもよいでしょう。

$HOME/.tiprc に次のように定義することで、 任意の文字を強制文字として利用できます。

```
force=<single-char>
```

11.19. 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(1) の "raise character" モードに入ったのでしょう。 既に述べた ~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、強制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。

11.20. 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` (送信) や `~t` (受信) 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(1) や echo(1) 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行いませんので、zmodem などの他のプロトコルを使った方がよいでしょう。

11.21. tip から zmodem を実行するには?

まず始めに、FreeBSD Ports Collection から zmodem プログラムのいずれか (lrzszと rzsz の、通信カテゴリーの2 つのプログラムのどちらか) をインストールします。

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、`Enter` キーを押してから “~C rz” (lrzsz をインストールした場合は “~C lrz”) 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、`Enter` キーを押してから “~C sz <files>” (lrzsz をインストールした場合は “~C lsz <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。

11.22. 設定が正しいのにもかかわらず、FreeBSD がシリアルポートを見付けられません。

マザーボードやシリアルカードが Acer の UART チップを使った物の場合、 FreeBSD の sio ドライバでは正しく検出する事が出来ません。 この問題を解決するためには、 www.lemis.com からパッチを入手してください。

Chapter 12. その他の質問

12.1. FreeBSD は Linux より多くのスワップ領域を消費するのはなぜですか?

実際にはそうではありません。 FreeBSD は Linux よりもスワップを多く使っているように見えるだけです。 この点における FreeBSD と Linux の主な違いは、 FreeBSD はより多くのメインメモリを有効利用できるようにするため、完全にアイドルになったものやメインメモリ上の使われなくなったページを、スワップにあらかじめ積極的に移動しているということです。 Linux では、最後の手段としてページをスワップに移動させるだけという傾向があります。 このスワップの使い方は、メインメモリをより効果的に使用することによってバランスが保たれています。

FreeBSD はこのような状況では先手策を取りますが、 システムが本当に空き状態の時に、理由も無くページをスワップしようと決めることはないということに注意してください。 したがって、夜中に使わずにおいたシステムが朝起きたとき、すべてページアウトされているということはないのです。

12.2. ほとんどプログラムは実行されていないのに、どうして top(1) は非常に少ない free memory を報告するのでしょうか?

簡単に言えば、free memory とは無駄になっているメモリのことだからです。 プログラムが確保しているメモリ以外のすべてのメモリは、 FreeBSD カーネル内でディスクキャッシュとして利用されます。 この値は top(1) において Inact、 Cache Buf として表示され、 それぞれは異なるエージングレベル (訳注: データがどれだけ古いかを示す評価値) でキャッシュされた全データを表します。 データがキャッシュされると言うのは、最近アクセスされたデータであれば、再度そのデータをアクセスするためにシステムが遅いディスクにアクセスする必要がない、ということを意味します。 そのため、全体のパフォーマンスが向上します。 一般的に、top(1) で表示される Free メモリが小さい値を示すことは良いことで、 自由に使えるメモリの残量が本当に少ない、 ということを表しているわけではありません。

12.3. FreeBSD の実行フォーマットの a.out、ELF とはどのようなものですか? また、a.out、ELF を使う理由は何でしょう?

FreeBSD が何故 ELF フォーマットを利用しているのかを理解するためには、 まず UNIXにおいて現在「優勢」な 3 種類の実行フォーマットについて いくらか知っておく必要があります。



FreeBSD 3.x より前の FreeBSD では a.out フォーマットが使われていました。

- a.out(5)

最も古く「由緒正しい」unixオブジェクトフォーマットです。
マジックナンバを含む短くてコンパクトなヘッダが先頭にあり、
これがフォーマットの特徴とされています (a.out(5) に詳細な内容があります)。 ロードされる3種類のセグメント、 `.text`、 `.data`、 `.bss` と加えてシンボルテーブルと文字列テーブルを含みます。

- COFF

  SVR3 のオブジェクトフォーマットです。 ヘッダは単一のセクションテーブルから成り、 `.text`、 `.data`、 `.bss` セクション以外の部分を持つことができます。

- ELF

  COFFの後継です。複数のセクションをサポートし、32-bitと64-bitのいずれの値も可能です。大きな欠点の一つは、ELFはそれぞれのシステムアーキテクチャ毎に単一のABIのみが存在するという仮定で設計されていることです。この仮定はまったく正しくありません。商用の SYSV の世界でさえそうです (少なくとも SVR4、 Solaris、SCO の 3種類の ABI があります)。

  FreeBSD はこの問題を解決するための試みとして、 既知の ELF 実行ファイルに ABI に応じた情報を書き加えるユーティリティを提供しています。詳しくは brandelf(1) のマニュアルページを参照してください。

FreeBSD は伝統的な立場をとり、数多くの世代の BSD のリリースで試され、実証されてきた a.out(5) フォーマットを伝統的に使用しています。 いつかは FreeBSD システムでネイティブ ELF バイナリを作り、 実行することができるようになるかもしれませんが、 初期の頃 FreeBSD では ELF をデフォルトのフォーマットに変更するという動きは ありませんでした。なぜでしょうか? ところで Linux においては、 ELF への苦痛をともなった変更は、 その時に a.out 実行フォーマットから逃れたというよりは、
ジャンプテーブルベースの共有ライブラリのメカニズムの柔軟性の低さからの脱却でした。
これはベンダや開発者全体にとって、 共有ライブラリの作成が非常に難しかった原因でした。 ELF のツールには共有ライブラリの問題を解決することができるものが提供されており、
またいずれにせよ一般的に「進歩」していると考えられます。
このため移行のコストは必要なものとして容認され、 移行は行なわれました。

FreeBSD の場合は、共有ライブラリのメカニズムは Sun の SunOS 形式の共有ライブラリのメカニズムに極めて近いものになっていて、非常に使いやすいものになっています。
しかしながら、FreeBSD では 3.0 から ELF バイナリをデフォルトのフォーマットとして公式にサポートしています。 a.out 実行フォーマットはよいものを私達に提供してくれているものの、
私たちの使っているコンパイラの作者である GNU の人々は a.out フォーマットのサポートをやめてしまったのでした。 このことは、私たちに別バージョンのコンパイラとリンカを保守することを余儀なくされることとなり、 最新の GNU 開発の努力による恩恵から遠ざかることになります。 その上、ISO C++ の、とくにコンストラクタやデストラクタがらみの要求もあって、今後の FreeBSD のリリースでネイティブの ELF のサポートされる方向へと話が進んでいます。

12.4. それにしても、なぜそんなに多くのフォーマットがあるのですか?

もうおぼろげになってしまった暗い過去に、単純なハードウェアがありました。
この単純なハードウェアは、単純で小さなシステムをサポートしていました。 a.out はこの単純なシステム (PDP-11) での作業を行なうバイナリとして完全に適したものだったのです。
人々はこの単純なシステムから UNIX を移植する際に、a.out フォーマットをそのまま使いました。というのは Motorola 68k、VAXen、 といったアーキテクチャへの UNIX の初期の移植ではこれで十分だったからです。

やがてある聡明なエンジニアが、 ソフトウェアでちょっとしたトリックを使うことを決めました。
彼はいくつかのゲートを削り取って CPU のコアをより速く走らせることができたのです。
これは新しい種類のハードウェア (今日では RISC として知られています) で動いたのです。 a.out はこのハードウェアには適していなかったので、 このハードウェア上で多くのフォーマットが、限定された単純な a.out フォーマットでのものよりもより良いパフォーマンスを出すことを目指して開発されたのです。 COFF、ECOFF、 そしていくつかの有名でないフォーマットが ELF が標準になる前に開発され、それらの限界が探求されたのです。

さらに、プログラムサイズは巨大になり、 ディスク (および物理メモリ) は依然として相対的に小さかったため、 共用ライブラリのコンセプトが誕生しました。 また、VM システムはより複雑なものになりました。 これらの個々の進歩は a.out フォーマットを使用して遂げられましたが、
その有用性は新しい機能とともにどんどん広がってきました。
これらに加え、実行時に必要なものを動的にロードする、
または初期化コードの実行後にプログラムの一部を破棄し、
コアメモリおよびスワップ空間を節約するという要望が高まりました。
プログラミング言語はさらに複雑になり、`main` 関数の前に自動的にコールされるコードの要望が高まりました。 多くの機能拡張が行なわれ、a.out フォーマットがこれらすべてを実現できるようになり、 それらはしばらくは基本的に動作していました。
やがて、a.out はコードでのオーバヘッドと複雑さを増大させずに、これらの問題すべてを処理することに無理がでてきました。 一方、ELF はこれらの問題の多くを解決しますが、
現状稼働しているシステムからの切替えは厄介なものになるでしょう。 そのため ELF は、a.out のままでいることが ELF への移行よりももっと厄介なものになるまで待つ必要がありました。

しかし時が経つにつれ、FreeBSD のビルドツールの元となったツール群 (特にアセンブラとローダ) と FreeBSD のビルドツール群は異なった進化の経路をたどりました。 FreeBSD のツリーでは、共有ライブラリが追加され、 バグフィックスも行われました。
もともとのツール群を作成した GNU の人たちは、プログラムを書き直し、クロスコンパイラのサポート、 異なるフォーマットを任意に取り込む機能などを追加していきました。
多くの人々が FreeBSD をターゲットとしたクロスコンパイラの構築を試みましたが、 FreeBSD の使っている `as` と `ld` の古いプログラムコードはクロスコンパイルをサポートしておらず、うまくいきませんでした。 新しい GNU のツール群 (binutils) は、クロスコンパイル、共有ライブラリ、C++ 拡張などの機能をサポートしています。
さらに数多くのベンダが ELF バイナリをリリースしています。 FreeBSD にとって ELF バイナリが実行できることは、 非常にメリットがあります。ELF バイナリが FreeBSD

で動くのなら、a.out を動かすのに手間をかける必要はありませんね。
長い間忠実によく働いた老いた馬は、 そろそろ牧草地で休ませてあげましょう。

ELF は a.out に比べてより表現力があり、 ベースのシステムに対してより幅広い拡張性を提供できます。
ELF 用のツールはよりよく保守されています。
また多くの人にとって重要なクロスコンパイルもサポートしています。 ELF の実行速度は、ほんの少し a.out より遅いかもしれませんが、 実際に速度の差をはかるのは困難でしょう。 ELF と a.out の間には、ページマッピング、 初期化コードの処理など多くの違いがありますが、とりたてて重要なものはありません。しかし違いがあるのは確かです。ほどなく、 GENERIC カーネルから a.out のサポートが外されます。 a.out のプログラムを実行する必要性がなくなれば、最終的に a.out のサポートはカーネルから削除されます。

12.5. シンボリックリンクの許可属性を chmod で変えられないのはなぜですか?

シンボリックリンクは許可属性を持ちません。 また chmod(1) のデフォルト動作は、シンボリックリンクをたどってリンク先のファイルの許可属性を変更するようになっていません。
そのため、 foo というファイルがあり、 このファイルへのシンボリックリンク bar があったとすると、以下のコマンドは常に成功します。

```
% chmod g-w bar
```

しかしこの場合、foo の許可属性は変更されません。

この場合、“-H” か “-L” のどちらかのオプションを “-R” と同時に使う必要があります。 chmod(1) と symlink(7) のマニュアルページにはもっと詳しい情報があります。



“-R” オプションは再帰的に chmod(1) を実行します。ディレクトリやディレクトリへのシンボリックリンクを chmod する場合は気をつけてください。
シンボリックリンクで参照されている単一のディレクトリのパーミッションを変更したい場合は、 chmod(1) をオプションをつけずに、シンボリックリンクの名前の後ろにスラッシュ ("/") をつけて使います。たとえば、"foo" がディレクトリ "bar" へのシンボリックリンクである場合、 "foo" (実際には "bar") のパーミッションを変更したい場合には、このようにします。

```
% chmod 555 foo/
```

後ろにスラッシュをつけると、 chmod(1) はシンボリックリンク "foo" を追いかけてディレクトリ "bar" のパーミッションを変更します。

12.6. ログイン名がいまだに 8 文字に制限されているのはなぜですか?

UT_NAMESIZE を変更してシステム全体を作り直せば十分で、

それだけでうまくいくだろうとあなたは考えるかもしれません。
残念ながら多くのアプリケーションやユーティリティ (システムツールも含めて) は、
小さな数値を構造体やバッファなどに使っています (必ずしも "8" や "9" ではなく、 "15" や "20"
などの変った値を使うものもあります)。
(固定長のレコードを期待するところで可変長レコードになるため、)
台無しになったログファイルを得ることになるということだけでなく、 Sun の NIS
のクライアントの場合は問題が起きますし、他の UNIX
システムとの関連においてこれら以外の問題も起きる可能性があります。

しかし、FreeBSD 3.0 以降では 16 文字となり、
多くのユーティリティのハードコードされた名前の長さの問題も解決されます。
実際にはシステムのあまりに多くの部分を修正するために、 3.0
になるまでは変更が行われませんでした。

それ以前のバージョンでは、これらの問題が起こった場合に、
問題を自分自身で発見し、解決できることに絶対的な自信がある場合は /usr/include/utmp.h を編集し、
`UT_NAMESIZE` の変更にしたがって、 長いユーザ名を使うことができます。 また、 `UT_NAMESIZE`
の変更と一致するように /usr/include/sys/param.h の `MAXLOGNAME` 更新しなくてはなりません。
最後に、ソースからビルドする場合は /usr/include
を毎回アップデートする必要があることを忘れないように! /usr/src/..
上のファイルを変更しておいて置き換えましょう。

12.7. FreeBSD 上で DOS のバイナリを動かすことはできますか?

はい、FreeBSD 3.0 からは、 統合と改良が重ねられた BSDI の doscmd DOS
エミュレーションサブシステムを使ってできるようになりました。
今なお続けられているこの努力に興味を持って参加していただけるなら、 FreeBSD-emulation
メーリングリスト へメールを送ってください。

FreeBSD 3.0 以前のシステムでは、 pcemu という巧妙なユーティリティが FreeBSD Ports Collection にあり、 8088 のエミュレーションと DOS のテキストモードアプリケーションを動かすに十分な BIOS サービスを行ないます。これは X ウィンドウシステムが必要です (XFree86 として提供されています)。

12.8. どこで無料の FreeBSD のアカウントを取得できますか?

FreeBSD はいずれのサーバーにもアクセスを開放していませんが、 Unix
システムへの自由なアクセスを提供しているところがあります。
費用はまちまちで、限定されたサービスが利用できます。

M-Net としても知られる Arbornet, Inc は 1983 年から Unix システムへのアクセスを提供しています。System III が動作する Altos に始まり、1991 年には BSD/OS に移行しました。2000 年 6 月には、再び FreeBSD に 移行しています。M-Net には SSH または telnet 経由で アクセスすることができ、FreeBSD ソフトウェア一式が 利用できるようになっています。ただし、ネットワーク接続は 会員と、非営利組織として運営されているシステムに寄付をする 後援者に制限されています。また、M-Net は掲示板システムと 双方向チャットも提供しています。

Grex は、掲示板システムと双方向チャットソフトウェアが同じであることも含め、M-Net とよく似たサイトを提供しています。しかし、マシンは Sun 4M で、SunOS が動作しています。

12.9. sup とは何で、どのようにして使うものなのでしょうか?

SUP とは、ソフトウェアアップデートプロトコル (Software Update Protocol) で カーネギーメロン大学 (CMU) で開発ツリーの同期のために開発されました。
私たちの中心開発ツリーをリモートサイトで同期させるために使っていました。

SUP はバンド幅を浪費しますので、今は使っていません。
ソースコードのアップデートの現在のおすすめの方法は FreeBSD ハンドブックの「CVSup」にあります。

12.10. FreeBSD をクールに使うには?

いいえ。 私たちは 250 マイクログラムの LSD-25 をあらかじめ与えておいたボランティアに対する、目隠し味覚テストを大量に行なっています。 35% のボランティアは FreeBSD はオレンジのような味がすると言っているのに対し、Linux は紫煙のような味わいがあると言っている人もいます。
両方のグループとも温度の不一致については何も触れていません。
この調査で、非常に多くのボランティアがテストを行なった部屋から不思議そうに出てきて、
このようなおかしな結果を示したことに私たちは当惑させられました。
私たちは、ほとんどのボランティアは Apple にいて彼らの最新の「引っかいて匂いをかぐ」GUI を使っているのではないかと考えています。 私たちは奇妙な古い仕事をしているのでしょう!

真面目に言うと、FreeBSD や Linux は共に "HLT" (停止) 命令をシステムのアイドル (idle) 時に使い、エネルギーの消費を押えていますので熱の発生も少なくなります。 また、APM (advanced power management) を設定してあるなら FreeBSD は CPU をローパワーモードにすることができます。

12.11. 誰かが私のメモリカードをひっかいているのですか??

その通り! BSD の文書には良く、デーモン (daemon) という言葉が出てきます。
ほとんどの人は知らないのですが、デーモンとは、あなたのコンピュータを依り代とする、純粋で非物質的な存在のことです。 メモリから聞こえるひっかくような音は、さまざまあるシステム管理タスクの扱いをいかに最善なものにするか、
といったことを決めるときにデーモンたちが交わす、かん高いささやき声なのです。

この雑音が聞こえたとき、DOS から “fdisk /mbr” というプログラムを実行すれば、うまくデーモンを追い出すことができるでしょう。 でも、デーモンはそれに歯向かって `fdisk` の実行をやめさせようとするかも知れません。 もし、それを実行しているときにスピーカならビルゲイツ (Bill Gates) の悪魔のささやきが聞こえてきたら、すぐに立ち上がって逃げてください。決して振り返ってはいけません! BSD のデーモンたちが押え込んでいた双子のデーモン、DOS と Windows が解放され、あなたの魂を永遠の破滅へ導こうとマシンを再び支配してしまうことでしょう。
それを知った今や、選べと言われたら、むしろひっかき音に慣れる方を選ぶのではありませんか?

12.12. "MFC" とはどういう意味ですか

MFC とは、「CURRENT との合流 (Merged From -CURRENT)」の頭文字をとったものです。 CVS ログで -CURRENT から -STABLE ブランチへの合流を示します。

12.13. "BSD" とはどういう意味ですか?

この言葉は、仲間うちだけに分かる隠語で何とかという意味です。
文字どおりに訳すことはできませんが、 BSD の訳は「F1 のレーシングチーム」か「ペンギンはおいしいスナック」、 あるいは「俺たちゃ Linux より洒落は利いてるぜ」とかそのへんだと言っておけばおっけーでしょう。 :-)

冗談はさておき、BSD とは、Berkeley CSRG (コンピュータシステム評議会) が彼らの UNIX の配布形態の名前として当時選んだ "Berkeley Software Distribution" の略です。

12.14. リポジトリ・コピー (repo-copy) とは一体何のことでしょう?

repo-copy ("repository copy" の略) とは、 CVS リポジトリの中で直接ファイルをコピーすることを示す用語です。

repo-copy を行なわない場合を考えます。 リポジトリの中の異なる場所にファイルをコピーしたり、移動したりする必要性が生じると、コミッターは ファイルを新しい場所に置くために `cvs add` を、そして古いファイルが削除される場合は、古いファイルに対して `cvs rm` を実行するでしょう。

この方法の欠点は、ファイルの変更履歴 (たとえば CVS ログのエントリ) が新しい場所にコピーされないことです。 FreeBSD プロジェクトではこの変更履歴をとても有用なものだと考えているため、
前述の方法の代わりにリポジトリコピーが良く用いられます。 この操作は `cvs` プログラムを利用するのではなく、
リポジトリの管理担当者がリポジトリの中でファイルを直接コピーすることによって行なわれます。

12.15. なんでバイク小屋 (bikeshed) の色にまで気を使わなければいけないんですか?

一言で言ってしまえば、そうすべきではありません。 もう少し詳しく説明しましょう。
たとえば、あなたがバイク小屋を建てる技術を持っていたとします。
しかしそれは、塗ろうとしている色が気に入らないからと言って、
他人がバイク小屋を建てようとしているのを止めて良い理由にはなりませんよね。
これは、自分の行動について十分な理解を持っているなら、
あなたは細かな機能すべてにわたって議論する必要はないことを示す比喩です。
ある変更によって産み出されるノイズの総量は、
その変更の複雑さに反比例するのだと言っている人達もいます。

さらに詳しく、完全な回答を紹介しましょう。 Poul-Henning Kamp は、「sleep(1) は分数の秒数を引数として取るべきか」という 非常に長い議論の後で、 "A bike shed (any colour will do) on greener grass..." というタイトルの長文を投稿しました。

関係のある部分だけを以下に掲載します。

1999 年 10 月 2 日 freebsd-hackers にて Poul-Henning Kamp "このバイク小屋、どうだろう?" 誰かがたずねました。

長い...というか、むしろ古い話になりますが、 中身はわりと簡単な話です。パーキンソン (C. Northcote Parkinson) は 1960 年代初頭に "パーキンソンの法則" と呼ばれる本を書きました。この中にはさまざまな経営の力学に関する洞察が含まれています。

[この本に関する解説があったが省略]

バイク小屋に関連する例として、 もう一つの重要な構成要素となっているのは原子力発電所です。この本の年代がわかりますね。

パーキンソンは、あなたが重役会に出席して 数百万から数10億ドル規模の原子力発電所の建設の承認を得る
ことはできるでしょうが、あなたが建てたいのがバイク小屋ならば、
終わりなき議論に巻き込まれるだろうと言っています。

パーキンソンはこのように説明しています。
これは原発が余りに巨大で高価で複雑なので誰もこれを一手に握ることができず、
それを試みるくらいならむしろ、手が出せなくなる前に 他の誰かがすべてを詳細にチェックすることを引き受けることに頼るのです。 リチャード・ファインマン (Richard P. Feynmann) は、ロスアラモスでこの手の重要な経験を何度も見てきたと本に書いています。

一方でバイク小屋の場合は、誰でも週末にこれを作り上げることができ、 しかも TV の試合を見る時間があまるほどです。
なので、どんなに準備が整えてあって、どんなに計画が順当であったとしても、
わたしは仕事をやっているよ、 わたしは注意を払っているよ、そして わたしはここにいるよ、ということを示そうとする人が必ず現れます。

デンマークではこれを「指紋をつける」と呼んでいます。 これは個人的なプライドや名声を求め、ある場所を指し示して「ここ! ここは俺が やったんだぜ〜」というようなものです。
これは政治家に見られる強い特徴ですが、
その他のほとんどの人もこういう風に振舞う可能性はあるのです。
生乾きのセメントにつけられた足跡のことを考えればお分かりでしょう。

12.16. ひとつの電球を取り替えるのに、何人の FreeBSD ハッカーが必要?

1,172人です。

- 電球が消えていると -CURRENT で文句を言うのに 23 人。
- 設定上の問題で -questions で話をすべきことについて騒ぐのに 4 人。
- それを send-pr (訳注: 障害報告) するのに 3 人 (そのうちのひとつは間違って doc カテゴリに送りつけられたうえに、 内容が「暗くなった」というだけのもの)。
- buildworld を失敗させ、5 分後には元に戻されるような電球を テストもせずにコミットするのに 1 人。
- send-pr した人に、パッチが含まれていないと「いちゃもん」を付けるのに 8 人。

- buildworld が失敗すると文句を言うのに 5 人。
- 自分のところではちゃんと動く、 cvsup したタイミングが悪かったんだろうと答えるのに 31 人。
- 新しい電球のためのパッチを -hackers に投げるのに 1 人。
- 自分は 3 年も前にパッチを作ったが、それを -CURRENT に投げたときには無視されただけだった、自分は send-pr のシステムには嫌な経験があると (おまけに、提案された新しい電球には柔軟性が無いとまで) 文句を言うのに 1 人。
- 電球が基本システムに組み込まれていない、committer はコミュニティの意見を聞くこと無しにこんなことをする権利は無いと叫び、 「こんなときに -core は何をやってるんだ!?」とわめきちらすのに 37 人。
- 自転車置き場の色に文句を言うのに 200 人。
- パッチが style(9) 違反だと指摘するのに 3 人。
- 提案された新しい電球は GPL の下にあると文句を言うのに 70 人。
- GPL と BSD ライセンスと MIT ライセンスと NPL と、某 FSF 創立者らの個人的な健康法の優位性についての論争を戦わすのに 586 人。
- スレッドのあちこちの枝を -chat や -advocacy に移動するのに 7 人。
- 提案された電球を、古いのよりずっと薄暗いのにコミットしてしまうのに 1 人。
- FreeBSD に薄暗い電球を付けるくらいなら真っ暗のほうがましだという、コミットメッセージへの凄まじい非難の嵐によって、 それを元に戻すのに 2 人。
- 薄暗い電球が帳消しにされたことに対してどなり声で口論し、 -core の声明を要求するのに 46 人。
- もし FreeBSD をたまごっちに移植することになったときに都合がいいように、もっと小さな電球を要求するのに 11 人。
- -hackers と -chat の S/N比に文句を言い、 抗議のため講読を取りやめるのに 73 人。
- 「unsubscribe」「どうやったら講読をやめられるんですか?」「このメーリングリストからわたしを外してください」といったメッセージを、例のフッタをくっつけて投稿するのに 13 人。
- みんなが激論を戦わせるのに忙がしくて気付かない間に、 作業中の電球をコミットするのに 1 人。
- 新しい電球は TenDRA を使ってコンパイルされた場合に 0.364% も明るくなる (ただし電球を立方体にしなければならない)、 だから FreeBSD は EGCS から TenDRA に変えるべきだと指摘するのに 31 人。
- 新しい電球は美しさに欠けていると文句を言うのに 1 人。
- 「MFC って何ですか?」と聞くのに 9 人 (send-pr した人も含む)。
- 電球が取り替えられてから 2 週間も消えっぱなしだと文句を言うのに 57 人。



Nik Clayton <*nik@FreeBSD.org*> による追記:
これには爆笑しました。

それからわたしは考えました。 「ちょっと待てよ? このリストのどこかに、『これを文書にまとめるのに 1人』というのがあってもいいんじゃないか?」

それからわたしは悟りを開いたのです :-)

この項目の著作権は *Copyright (c) 1999 Dag-Erling Smørgrav* <*des@FreeBSD.org*> にあります。
無断で使用しないでください。

Chapter 13. まじめな FreeBSD ハッカーだけの話題

13.1. SNAP とか RELEASE とかは何?

現在、FreeBSD の CVS リポジトリ には、三つのアクティブ/準アクティブなブランチがあります (アクティブな開発ブランチは三つしか存在しないため、 おそらく RELENG_2 ブランチの変更は年に 2 回だけになるでしょう)。

- RELENG_2_2 通称 *2.2-STABLE*
- RELENG_3 通称 *3.X-STABLE*
- RELENG_4 通称 *4-STABLE*
- HEAD 通称 -CURRENT あるいは *5.0-CURRENT*

HEAD は他の二つと違って、 実際のブランチタグではなく、 「*current*、分岐していない開発本流」のための単なるシンボリックな定数です。 私たちはこれを -CURRENT と呼んでいます。

現在、 "-CURRENT" は 5.0 の開発本流であり、 4.0-STABLE ブランチ、 つまり RELENG_4 は 2000 年 3 月に "-CURRENT" から分岐しています。

2.2-STABLE ブランチ、 RELENG_2_2 は 1996 年 11 月に "-CURRENT" から分岐しました。 これは保守が完全に終了しています。

13.2. 自分用のカスタムリリースを構築するには?

リリースを構築するには三つのことが必要です。まず、 vn(4) ドライバが組み込まれたカーネルを実行させている必要があります。
以下をカーネルコンフィグレーションファイルに追加し、 カーネルを作り直してください。

```
pseudo-device vn        #Vnode driver (turns a file into a device)
```

次に、CVS リポジトリ全体を手元においておく必要があります。 これを入手するには CVSUP が使用できますが、supfile で release の名称を cvs にして 他のタグや date フィールドを削除する必要があります。

```
*default prefix=/home/ncvs
*default base=/a
*default host=cvsup.FreeBSD.org
*default release=cvs
*default delete compress use-rel-suffix

## Main Source Tree
src-all
src-eBones
src-secure
```

```
# Other stuff
ports-all
www
doc-all
```

そして `cvsup -g supfile` を実行して自分のマシンに CVS リポジトリ全体をコピーします...。

最後に、ビルド用にかなりの空き領域を用意する必要があります。 そのディレクトリを /some/big/filesystem として、 上の例で CVS リポジトリを /home/ncvs に置いたものとすると、以下のようにしてリリースを構築します。

```
# setenv CVSROOT /home/ncvs
 # or export CVSROOT=/home/ncvs
# cd /usr/src
# make buildworld
# cd /usr/src/release
# make release BUILDNAME=3.0-MY-SNAP CHROOTDIR=/some/big/filesystem/release
```



ただし、すでに /usr/obj 以下に構築物が存在しているなら、buildworld の必要はありません。

処理が終了すると、 リリース全体が /some/big/filesystem/release に構築され、完全な FTP インストール用の配布物が /some/big/filesystem/release/R/ftp に作成されます。 -current 以外の開発ブランチの SNAP を自分で構築したい場合は、 `RELEASETAG=SOMETAG` を上の `make release` のコマンドラインに追加します。 たとえば、`RELEASETAG=RELENG_2_2` とすると最新の 2.2-STABLE snapshot が構築されます。

13.3. カスタムのインストールディスクを作るにはどうすればいいのですか?

/usr/src/release/Makefile のいろいろなターゲットとしてインストールディスク、ソース、バイナリアーカイブを作る完全な処理を自動的に行なうようになっています。 Makefile に十分な情報があります。 しかし、実行には "make world" が必要で、多くの時間とディスクの容量が必要です。

13.4. make world を行なうと既存のバイナリを上書きしてしまうのですが。

ええ、それが一般的な考え方です。名前が示しているように "make world" はすべてのシステムのバイナリを最初から作り直しますので、結果として、クリーンで一貫性のある環境を得ることができます (これがそれだけ長い時間がかかる理由です)。

環境変数 `DESTDIR` を `make world` や `make install` を実行する時に定義しておくと、新しく作られたバイナリは `${DESTDIR}`を `root` とみなしたディレクトリツリーにインストールされます。

あるでたらめな共有ライブラリの変更やプログラムの再構築によって `make world` は失敗することもあります。

13.5. システム起動時に (bus speed defaulted) とメッセージが出ます。

Adaptec の 1542 SCSI ホストアダプタは、ユーザがソフトウェア的にバスアクセス速度の設定を行なうことができます。 以前のバージョンの 1542 ドライバは、使用可能な最大の速度を求めてアダプタをその設定にしようとしました。これは特定のユーザのシステムでは問題がある事がわかり、 現在ではカーネルコンフィグオプションに "TUNE_1542" が加えられています。これを使用すると、これが働くシステムではディスクが速くなりますが、データの衝突が起きて速くはならないシステムもあるでしょう

13.6. インターネットアクセスに制限があっても current を追いかけられますか?

はい、CTM システムを使って、ソースツリー全体のダウンロードを行なわずに追いかけることができます。

13.7. どのようにして配布ファイルを 240KB に分割しているのですか?

比較的新しい BSD ベースのシステムでは、`split` に任意のバイト境界で分割する “-b” オプションがあります。

以下は /usr/src/Makefile からの例です。

```
bin-tarball:
            (cd ${DISTDIR}; \
            tar cf - . \
            gzip --no-name -9 -c | \
            split -b 240640 - \
            ${RELEASEDIR}/tarballs/bindist/bin_tgz.)
```

13.8. 私はカーネルに拡張を行ないました。誰に送ればいいですか?

FreeBSD ハンドブックの「FreeBSD への貢献」を参照してください。

あなたのアイディアに感謝します!

13.9. PnP ISA
カードの検出と初期化はどのように行なうのですか?

Frank Durda IV 氏 より:

要点は、ホストが認識されていないボードを探す時に、すべての PnP ボードが応答することのできる少数の I/O ポートがあるということです。 それにより、PnP プローブルーチンが開始したとき、PnP ボードが存在するなら、すべての PnP ボードは自分のモデル番号を返します。 そのポートを I/O read するとプローブルーチンは問いに対するワイアード-OR された "yes" を得ます。この場合は 少なくとも 1 ビットが ON になります。 そして、プローブルーチンはモデル ID (Microsoft/Intel によって割り当てられています)が X より小さいボードを "オフライン" にすることができます。 この操作を行ない、問い合わせに応答しているボードがまだ 残っているかどうかを調べます。 もし “0” が返ってくるなら X より大きな ID を持つボードはないことになります。 今度は "X" よりも小さな値を持つボードについて問い合わせます。 もしあるのであれば、プローブルーチンはモデル番号が X より小さいことを知ります。 今度は、X-(limit/4) より大きな値を持つボードをオフラインにして問い合わせを繰り返します。 この ID の範囲による準バイナリサーチを十分繰り返すことにより、
プローブルーチンはマシンに存在するすべての PnP ボードの値を最終的に得ることができます。その繰り返しの回数は 2^64 よりはるかに少ない回数です。

ID は二つの 32-bit (つまり 64bit) フィールド + 8 bit チェックサムからなります。最初の 32 bits はベンダの識別子です。 これは公表されてはいませんが、同一のベンダから供給されている異なるタイプのボードでは異なる 32-bit ベンダ ID を持つことができるように考えられます。 製造元を特定するだけのために 32-bit はいくらか過剰です。

下位の 32-bit はシリアル番号、 イーサネットアドレスなどのボードを特定するものです。 ベンダは上位 32 bits が異なっていないのであれば、 下位 32-bit が同一である 2枚目のボードを製造することはありません。
したがって、同じタイプの複数のボードをマシンにいれることができ、 この場合でも 64-bit 全体ではユニークです。

32-bit のフィールドはすべてを 0 にすることはできません。
これは初期化のバイナリサーチの間ワイアード-OR によって 0 ではない ビットを参照するからです。

システムがすべてのボードの与えられた ID を認識すると、 それぞれのボードに対応した処理を一つずつ (同一の I/O ポートを通して) 行ないます。
そして、利用できる割り込みの選択などのボードが必要とするリソースを検出します。
すべてのボードについてこの情報を集めます。

この情報はハードディスク上の ECU ファイルなどの情報とまとめられ、 マザーボードの BIOS にも結合されます。 マザーボード上のハードウェアへの ECU と BIOS PnP のサポートは通常は統合されていますが、 周辺機器については真の PnPであるとはいえません。
しかし、BIOS の情報に ECU の情報を加えて調査することで、 プローブルーチンは PnP デバイスが再配置できなくなることを避けることができます。

それから、再度 PnP デバイスにアクセスし、I/O、DMA、IRQ、 メモリマップアドレスの設定をします。
デバイスはこのアドレスに見えるようになり、 次に再起動するまでこの位置を占めます。しかし、あなたの望む時に移動させることが不可能である、 といっているわけではありません。

以上の話では大きく単純化をしてありますが、 基本的な考え方は得られたでしょう。

マイクロソフトは、ボードのロジックが対立する I/O サイクルではデコードしていない (訳注: おそらく read 時しかデコードされていず write 時はポートが空いているという意味でしょう)、プライマリプリンタのステータスポートのいくつかを PnP のために占有しました。 私は初期の PnP の提案レビュー時に IBM 純正のプリンタボードでステータスポートの write のデコードがされているということに気がつきましたが、 MS は "tough (頑固、不運、無法な)" と言っています。 そしてプリンタのステータスポートへアドレスの設定のために write を行なっています。また、 そのアドレス + `0x800` と read のための 3番目の I/O ポートが `0x200` から `0x3ff` の間のどこかに置かれるでしょう。

13.10. FreeBSD は、他のアーキテクチャをサポートしないんですか?

いくつかのグループの人々が、FreeBSD の他のアーキテクチャへの移植に関心を示しており、FreeBSD/AXP (ALPHA) はこれらの成果としてはとても成功したものの一つです。 FreeBSD/AXP は現在 ftp://ftp.FreeBSD.org/pub/FreeBSD/alpha から入手できます。 ALPHA への移植版が現在動く機種は増えつつあり、 その中には AlphaStation、AXPpci、PC164、Miata そして Multia といったモデルが含まれています。 現状についての情報を得るには freebsd-alpha@FreeBSD.orgメーリングリストに参加してください。

その他に FreeBSD の SPARC アーキテクチャへの移植があります。プロジェクトへの参加に興味がある方は freebsd-sparc@FreeBSD.orgメーリングリストに参加してください。 進行中のプラットホームのリストにもっとも最近追加されたのが IA-64 と PowerPCです。詳細は freebsd-ia64@FreeBSD.org および/あるいは freebsd-ppc@FreeBSD.orgメーリングリストに参加してください。
新しいアーキテクチャに関する一般的な議論については
新しいアーキテクチャに関する一般的な議論については freebsd-platforms@FreeBSD.orgメーリングリスト へ参加してください。

13.11. デバイスドライバを開発したので、メジャー番号が欲しいのですが。

これは、開発したドライバを公開するかどうかに依存します。
公開するのであれば、ドライバのソースコード、 files.i386 の変更、 コンフィグファイルのサンプル、デバイスが使うスペシャルファイルを作成する MAKEDEV(8) のコードを私たちに送ってください。
公開するつもりがない場合、ライセンスの問題により公開できない場合は、 キャラクタメジャー番号 32 および、 ブロックメジャー番号 8 がこのような目的のために予約されています。これらの番号を使用してください。 どちらの場合であれ、ドライバに関する情報を FreeBSD technical discussions メーリングリスト に流して頂けると助かります。

13.12. 代替のディレクトリ配置ポリシー

現在使われているディレクトリの配置ポリシーは、 私が 1983 年に書いたものから全く変更されていません。 私は当初の配置ポリシーを、オリジナルの fast filesystem

のために書き、 まったく改定していません。
このポリシーはシリンダグループを使い尽くすのを防ぐにはうまくいきましたが、
お気づきの方もいる通り find の動作には不適切です。 ほとんどのファイルシステムの内容は、
深さ優先検索 (ftw とも呼ばれます) によって作られたアーカイブから、 抽出 (restore)
して作成されます。この際、 ディレクトリは、シリンダグループにまたがって配置され、
以降の深さ優先検索を行うには、 考え得る限り最悪の状態になります。
もし作成するディレクトリの総数がわかっていれば、 解決方法はあります。(総数/シリンダグループ数)
個のディレクトリを、 シリンダグループごとにまとめて作成すれば良いのです。
もちろん最適なディレクトリ配置になるように、 総数を予測する方法を考えなければなりません。
しかし仮にシリンダグループあたりのディレクトリ数を 10
くらいの小さな数に固定してしまったとしても、 大幅な改善が望めるでしょう。
このポリシーを用いるべきリストア作業を、通常の作業
(おそらく既存のポリシーを使用したほうが良いでしょう) を区別するには、 10
秒間の間に作成されたディレクトリを最大 10
個までまとめて単一のシリンダグループに書き込むという手順が使えるでしょう。
とにかく私の結論は、そろそろ実験を始めて見る時期だろうということです。

13.13. カーネルパニックを最大限に利用する



この節は、freebsd-current メーリングリストに Bill Paul <wpaul@FreeBSD.org> 氏が投稿したメールを、 Dag-Erling Smørgrav <des@FreeBSD.org> 氏が校正し、[] 内のコメントを追加して引用したものです。

```
From: Bill Paul <wpaul@skynet.ctr.columbia.edu>
Subject: Re: the fs fun never stops
To: ben@rosengart.com
Date: Sun, 20 Sep 1998 15:22:50 -0400 (EDT)
Cc: current@FreeBSD.ORG
```

[<ben@rosengart.com> が以下のパニックメッセージを投稿しました。]

```
> Fatal trap 12: page fault while in kernel mode
> fault virtual address   = 0x40
> fault code              = supervisor read, page not present
> instruction pointer     = 0x8:0xf014a7e5
                              ^^^^^^^^^^
> stack pointer           = 0x10:0xf4ed6f24
> frame pointer           = 0x10:0xf4ed6f28
> code segment            = base 0x0, limit 0xfffff, type 0x1b
>                         = DPL 0, pres 1, def32 1, gran 1
> processor eflags        = interrupt enabled, resume, IOPL = 0
> current process         = 80 (mount)
> interrupt mask          =
> trap number             = 12
> panic: page fault
```

このようなメッセージが表示された場合、問題が起きる状況を確認して、

情報を送るだけでは十分ではありません。　　下線をつけた命令ポインタ値は重要な値ですが、
残念ながらこの値は構成に依存します。つまり、
この値は使っているカーネルのイメージに依存するのです。　　もしスナップショットなどの　GENERIC
カーネルを使っているのであれば、　　他の人間が問題のある関数について追試をすることができますが、
カスタマイズされたカーネルの場合は、
使っている本人にしか問題の起こった場所は特定できないのです。

何をすれば良いのでしょう?

1. 命令ポインタ値をメモします。 `0x8:` という部分は今回必要ありません。 必要なのは `0xf0xxxxxx` という部分です。
2. システムが再起動したら、以下の操作を行います。

   ```
   % nm -n /kernel.that.caused.the.panic | grep f0xxxxxx
   ```

   ここで、`f0xxxxxx`　　は命令ポインタ値です。
   カーネルシンボルのテーブルは関数のエントリポイントを含み、
   命令ポインタ値は、関数内部のある点であり最初の点ではないため、
   この操作を行っても完全に一致するものが表示されない場合もあります。　　この場合は、
   最後の桁を省いてもういちどやってみてください。 このようになります。

   ```
   % nm -n /kernel.that.caused.the.panic | grep f0xxxxx
   ```

   これでも一致しない場合は、　　桁を減らしながら何らかの出力があるまで繰り返してください。
   何か出力されたら、　　それがカーネルパニックを引き起こした可能性のある関数のリストです。
   これは、問題点を見付ける正確な方法ではありませんが、何もないよりましです。

このようなパニックメッセージを投稿している人はよく見掛けますが、 このように、命令ポインタ値を、
カーネルシンボルテーブルの中の関数とつき合わせて調べている人はまれです。

パニックの原因を突き止める最良の方法は、クラッシュダンプをとり、　　gdb(1)
でスタックトレースを行うことです。

どっちにしろ、私は普通以下のようにします。

1. カーネルコンフィグファイルを作ります。 カーネルデバッガが必要そうであれば `options 'DDB'` を加えても良いです　　(私は永久ループが起こっていそうな場合に、
   ブレークポイントを設定するのに使っています)。
2. `config -g KERNELCONFIG` としてビルドディレクトリを設定します。
3. `cd /sys/compile/KERNELCONFIG; make` を実行します。
4. カーネルのコンパイルが終了するのを待ちます。
5. `make install` を実行します。
6. 再起動します。

make(1) プロセスは２つのカーネル、 kernel と kernel.debug をビルドします。 kernel は /kernel としてインストールされ、 kernel.debug は gdb(1) のデバッグ用シンボル情報を取り出すために利用されます。

確実にクラッシュダンプをとるには、/etc/rc.conf を編集して dumpdev を使用しているスワップパーティションに指定する必要があります。 こうすると rc(8) スクリプトから dumpon(8) コマンドが実行され、 クラッシュダンプ機能が有効になります。 手動で dumpon(8) コマンドを実行してもかまいません。 パニックの後、クラッシュダンプは savecore(8) コマンドを使用して取り出すこと ができます。 dumpdev が /etc/rc.conf で設定されていれば、 rc(8) スクリプトから savecore(8) が自動的に実行され、クラッシュダンプを /var/crash に保存します。



FreeBSD のクラッシュダンプのサイズは、 ふつう物理メモリサイズと同じです。 つまり 64MB のメモリを積んでいれば、 64MB のクラッシュダンプが生成されることになります。 /var/crash に十分な空き容量があることを確認してください。手動で savecore(8) を実行すれば、もっと空き容量のあるディレクトリにクラッシュダンプを保存できます。 options MAXMEM=(foo) という行をカーネルコンフィグファイルに追加することで、カーネルのメモリ使用量を制限できます。 たとえば 128MB のメモリがある場合も、カーネルのメモリ使用量を 16MB に制限し、クラッシュダンプのサイズも 128MB ではなく 16MB にすることができます。

クラッシュダンプを取り出せたら、 以下のように gdb(1) を使ってスタックトレースをとります。

```
% gdb -k /sys/compile/KERNELCONFIG/kernel.debug /var/crash/vmcore.0
(gdb) where
```

必要な情報が 1 画面に収まらないことも多いので、できれば script(1) を使って出力を記録します。 strip していないカーネルイメージを使うことで、 すべてのデバッグシンボルが参照でき、パニックの発生したカーネルのソースコードの行が表示されているはずです。
通常、正確なクラッシュへの過程を追跡するには、
出力を最後の行から逆方向に読まなければなりません。 また gdb(1) を使って、変数や構造体の内容を表示させ、 クラッシュした時のシステムの状態を調べられます。

もしあなたがデバッグ狂で、同時に別のコンピュータを利用できる環境にあれば、 gdb(1) をリモートデバッグに使うこともできます。 リモートデバッグを使うと、あるコンピュータ上の gdb(1) を使って、 別のコンピュータのカーネルをデバッグできます。
ブレークポイントの設定、カーネルコードのステップ実行など、
ふつうのプログラムのデバッグと変わりません。 コンピュータを 2 台並べてデバッグするチャンスにはなかなか恵まれないので、
私はまだリモートデバッグを試したことはありません。



Bill による追記:

DDB を有効にしていてカーネルがデバッガに 落ちたら、ddb のプロンプトで “panic” と入力すれば、強制的にパニックを起こしクラッシュダンプさせることができます。
パニックの途中で、再びデバッガに落ちるかもしれませんが、 “continue” と入力すれば、 クラッシュダンプを最後まで実行させられます。

13.14. dlsym() が ELF 実行形式では動作しなくなります!

ELF のツール類は、デフォルトでは実行形式の中に定義されているシンボルを、ダイナミックリンカから見えるようにはしません。このため、dlopen(NULL, flags) を呼び出して得られたハンドルに対して、dlsym() で探索を行っても、こういったシンボルを見つけられません。

もし、あなたがプロセスの中心にあたる実行形式の中にあるシンボルを探索したければ、ELF リンカ (ld(1)) に -export-dynamic オプションを付けて実行形式をリンクする必要があります。

13.15. カーネルアドレス空間を大きくしたり、小さくするにはどうしたら良いのですか?

カーネルアドレス空間は、FreeBSD 3.X 上で 256MB、FreeBSD 4.X 上で 1GB がデフォルトになっています。負荷の高いネットワークサーバ (たとえば大きな FTP、HTTP サーバ) を運用する場合は、256MB では足りないことに気付くかも知れません。

では、アドレス空間を大きくするにはどうしたら良いのでしょうか?
それには、二つの段階を踏みます。まず、
より大きいアドレス空間を割り当てることをカーネルに知らせる必要があります。
次に、カーネルはアドレス空間の先頭にロードされるため、アドレスの先頭が天井 (訳注:カーネルアドレス空間の最下端アドレスのこと) とぶつかることのないように、ロードアドレスを今までより低位に設定する必要があります。

最初の段階は、src/sys/i386/include/pmap.h にある NKPDE の値を増加させることで行ないます。ここに 1GB のアドレス空間にするために、どのようにすれば良いかを示します。

```
#ifndef NKPDE
#ifdef SMP
#define NKPDE                   254     /* addressable number of page tables/pde's */
#else
#define NKPDE                   255     /* addressable number of page tables/pde's */
#endif  /* SMP */
#endif
```

正確な NKPDE の値を計算するには、望みのアドレス空間の大きさ (メガバイト単位) を 4 で割って、それから単一プロセッサ (UP) なら 1、SMP なら 2 を引き算してください。

次の段階を行なうには、ロードアドレスを正確に計算することが必要です。
単純に、アドレス空間の大きさ (バイト単位) を 0x100100000 から引き算してください。1GB アドレス空間の場合、その結果は 0xc0100000 になります。そして、src/sys/i386/conf/Makefile.i386 にある LOAD_ADDRESS に、今計算した値を入れます。また、次のように src/sys/i386/conf/kernel.script のセクションの始めの方にあるロケーションカウンタにも同じ値を入れてください。

```
OUTPUT_FORMAT("elf32-i386", "elf32-i386", "elf32-i386")
OUTPUT_ARCH(i386)
ENTRY(btext)
```

```
SEARCH_DIR(/usr/lib); SEARCH_DIR(/usr/obj/elf/home/src/tmp/usr/i386-unknown-
freebsdelf/lib);
SECTIONS
{
/* Read-only sections, merged into text segment: */
. = 0xc0100000 + SIZEOF_HEADERS;
.interp     : { *(.interp)    }
```

それが完了したら、`config` し直してカーネルを再構築してください。 おそらく、ps(1)、top(1) などに不具合が出るでしょう。 それらを正常にするために、`make world` (もしくは、変更した pmap.h を /usr/include/vm/ にコピーした後に、 libkvm、 `ps` および `top` を手動で再構築すること) を行なうべきです。



カーネルアドレス空間の大きさは、4MB の倍数である必要があります。



David Greenman <*dg@FreeBSD.org*> 氏による補足*:*

カーネルアドレス空間は 2 の乗数である必要があると思いますが、
それが確かなことかどうかははっきりしていません。
昔の起動コードには、良く高位アドレスビットのトリックが使われていたため、
少なくとも 256MB の粒度であることが想定されていたと思います。

Chapter 14. 謝辞

FreeBSD Core Team この FAQ について問題を見つけたり、何か登録したい場合は、 FAQ 管理者 <faq@FreeBSD.org> までメールを送ってください。
フィードバックしてくれるみなさんには感謝感謝なのです。 みなさんに手伝ってもらわないとこの FAQ はよくなりませんから!

Jordan K. Hubbard <jkh@FreeBSD.org>

たまに起こす FAQ の並べ替えや更新の発作

Doug White <dwhite@FreeBSD.org>

freebsd-questions メーリングリストでの義務を超えたサービス

Jörg Wunsch <joerg@FreeBSD.org>

Usenet (NetNews) での義務を超えたサービス

Garrett Wollman <wollman@FreeBSD.org>

ネットワーク節の執筆と文書整形

Jim Lowe

マルチキャストについて

Peter da Silva <pds@FreeBSD.org>

FreeBSD FAQ タイピング機械奴隷

FreeBSD チーム

不平を言ったり、うめいたり、情報提供してくれたり

あと、抜けてしまった他の方々に対して、謝罪と心からの感謝を捧げます!

Chapter 15. FreeBSD FAQ 日本語化について

FreeBSD 日本語ドキュメンテーションプロジェクトは、 FreeBSD 関係の日本語文書が少ないことを嘆いた数人の FreeBSD ユーザの提唱によって 1996 年 2 月 26 日にスタートし、 FreeBSD 日本語ハンドブックの作成をはじめとした活動を行なってきました。 FreeBSD FAQ の日本語化についてはオリジナルの翻訳作業だけでなく、 日本国内に固有の話題についても広く情報を集め、 日本の FreeBSD ユーザにとって真に有益なドキュメントを提供しようと考えています。 オリジナルの FAQ は日毎に更新されており、
私たちもまたこれに追い付くために作業を続けていきます。もちろん、新しいメンバも大歓迎です。
日本語翻訳版について、何かお気づきの点がありましたら、 日本語ドキュメンテーションプロジェクト <doc-jp@jp.FreeBSD.org> までご連絡ください。 また、もし私たちの作業を手伝ってくれるなら、 FreeBSD 日本語ドキュメンテーションプロジェクトのページをご覧の上、是非参加してください。

15.1. 翻訳者 (五十音順)

- 有村 光晴 <arimura@jp.FreeBSD.org>
- 一宮 亮 <ryo@azusa.shinshu-u.ac.jp>
- 岩崎 満 <iwasaki@jp.FreeBSD.org>
- 内川 喜章 <yoshiaki@kt.rim.or.jp>
- 栗山 淳 <kuriyama@FreeBSD.org>
- こがよういちろう <y-koga@ccs.mt.nec.co.jp>
- 今野 元之 <motoyuki@FreeBSD.org>
- 杉村 貴士 <sugimura@jp.FreeBSD.org>
- 中井 幸博 <nakai@FreeBSD.org>
- にしか <nishika@cheerful.com>
- 花井 浩之 <hanai@FreeBSD.org>
- はらだ きろう <kiroh@jp.FreeBSD.org>
- 広瀬 昌一 <shou@kt.rim.or.jp>
- 福間 康弘 <yasuf@big.or.jp>
- むらたしゅういちろう <mrt@mickey.ai.kyutech.ac.jp>
- 山下 淳 <junkun@esys.tsukuba.ac.jp>

15.2. 査読者 (五十音順)

- 浅見 賢 <asami@FreeBSD.org>
- 岩崎 満 <iwasaki@jp.FreeBSD.org>
- 内川 喜章 <yoshiaki@kt.rim.or.jp>
- 大橋 健 <ohashi@mickey.ai.kyutech.ac.jp>

- 栗山 淳 <kuriyama@FreeBSD.org>
- 今野 元之 <motoyuki@FreeBSD.org>
- 佐伯 隆司 <saeki@jp.FreeBSD.org>
- 杉村 貴士 <sugimura@jp.FreeBSD.org>
- 花井 浩之 <hanai@FreeBSD.org>
- 浜田 直樹 <nao@tom-yam.or.jp>
- はらだ きろう <kiroh@jp.FreeBSD.org>
- 日野 浩志 <hino@ccm.cl.nec.co.jp>
- 檜山 卓 <shiyama@intercity.or.jp>
- 広瀬 昌一 <shou@kt.rim.or.jp>
- むらたしゅういちろう <mrt@mickey.ai.kyutech.ac.jp>
- 若井 久史 <earth@hokuto7.or.jp>

15.3. 作業環境整備 (五十音順)

- 一宮 亮 <ryo@azusa.shinshu-u.ac.jp>
- 岩崎 満 <iwasaki@jp.FreeBSD.org>
- 下川 英敏 <simokawa@jp.FreeBSD.org>
- 鈴木 秀幸 <hideyuki@jp.FreeBSD.org>

有用な書籍

[biblio-44sysman] 4.4BSD System Manager’s Manual. Computer Systems Research Group, University of California, Berkeley. O’Reilly and Associates. 1st Edition. June 1994. 804 pages. ISBN 1-56592-080-5.

[biblio-44userman] 4.4BSD User’s Reference Manual. Computer Systems Research Group, University of California, Berkeley. O’Reilly and Associates. 1st Edition. June 1994. 905 pages. ISBN 1-56592-075-9.

[biblio-44suppman] 4.4BSD User’s Supplementary Documents. Computer Systems Research Group, University of California, Berkeley. O’Reilly and Associates. 1st Edition. June 1994. 712 pages. ISBN 1-56592-076-7.

[biblio-44progman] 4.4BSD Programmer’s Reference Manual. Computer Systems Research Group, University of California, Berkeley. O’Reilly and Associates. 1st Edition. June 1994. 866 pages. ISBN 1-56592-078-3.

[biblio-44progsupp] 4.4BSD Programmer’s Supplementary Documents. Computer Systems Research Group, University of California, Berkeley. O’Reilly and Associates. 1st Edition. June 1994. 596 pages. ISBN 1-56592-079-1.

[biblio-44kernel] The Design and Implementation of the 4.4BSD Operating System. McKusick M. K. [FAMILY Given], Marshall Kirk [FAMILY Given], Bostic Keith [FAMILY Given], Karels Michael J [FAMILY Given], 、 Quarterman John [FAMILY Given]. Addison-Wesley. Reading MA . 1996. ISBN 0-201-54979-4.

[biblio-nemeth3rd] Unix System Administration Handbook. Nemeth Evi [FAMILY Given], Snyder Garth [FAMILY Given], Seebass Scott [FAMILY Given], Hein Trent R. [FAMILY Given], 、 Quarterman John [FAMILY Given]. Prentice-Hall. 3rd edition. 2000. ISBN 0-13-020601-6.

[lehey3rd] The Complete FreeBSD. Lehey Greg [FAMILY Given]. Walnut Creek. 3rd edition. June 1999. 773 pages. ISBN 1-57176-246-9.

[McKusick et al, 1994] Berkeley Software Architecture Manual, 4.4BSD Edition. McKusick M. K. [FAMILY Given], Karels M. J. [FAMILY Given], Leffler S. J. [FAMILY Given], Joy W. N. [FAMILY Given], 、 Faber R. S. [FAMILY Given]. 5:1-42.