Debian リファレンス

Osamu Aoki (青木 修) `<osamu\#at\#debian.org>`
翻訳: 角田 慎一 `<tsuno\#at\#ngy.1st.ne.jp>`
'著者' on page 255

CVS, 2007年 1月 18日 木曜日 11時54分01秒 UTC時間

概要

本Debian リファレンス (`http://qref.sourceforge.net/`)は**システム導入後のユーザーガイド**として、Debian システムの広範な説明を提供します。本書はシェルコマンド例を通してシステム管理の多くの局面をカバーします。Debian システム、システム導入ヒント、Debian パッケージ管理、Debian のもとでの Linux kernel、システムのチューニング、gateway の組み上げ方、テキストエディター、CVS、プログラミングや GnuPG に関して基本的な入門情報やティップ他が、**non-developers**を対象に提供されています。

著作権表示

目次

章 1

序文

本 Debian リファレンス (`http://qref.sourceforge.net/`) はインストール後のユーザガイドとして Debian システムの広範囲な概観を供給することを意図しています。 対象とする読者はシェルスクリプトを読むのを厭わない人です。読者が本文書を 読む前に Unix ライクなシステムでの基本的なスキルを既に取得していることを 想定しています。

詳細がマニュアルページ, info ページ, HOWTO 文書に見つかる場合は、全てを 詳しく説明しないことにしました。 完全に説明する代わりに、正確なコマンドシーケンスを文書内に入れたり、 参照として `http://www.debian.org/doc/manuals/debian-reference/examples/` にサンプルスクリプトを 置いたりすることにより、より直接な実践的情報を与えようと試みています。 コマンドを実行する前にスクリプトの中身を理解する必要があります。 システムによってコマンドシーケンスを少し変える必要があるかもしれません。

本文書に含まれるほとんどの情報は、'参考文献' on page 247 にリスト化 されている権威あるリファレンスへのリマインダやポインタから 構成されています。

本文書は、元々は "quick reference" として作成されましたが、今や大規模な 文書に成長しました。 それにもかかわらず、**Keep it short and simple** (KISS) がこのガイドの原則です。

緊急のシステム維持の手助けが欲しい場合は、すぐに 'Debian で生き残るためのコマンド' on page 85 に進んでください。

1.1 公式文書

Debian の最新版の公式文書は `debian-reference-ja` にパッケージ化されており、`http://www.debian.org/doc/manuals/debian-reference/` からも取得可能 です。

最新の開発版は `http://qref.sourceforge.net/Debian/` にあります。 本プロジェクトは `http://qref.sourceforge.net/Debian/` でホスト されています。本文書はプレインテキスト, HTML, PDF, SGML, そして PostScript フォーマットで取得可能です。

1.2 本文書の慣習

この “Debian リファレンス” は短い `bash` シェルコマンドを通じて 情報を供給します。 次に規約を示します。

```
# command は root アカウントでのコマンド実行
$ command はユーザアカウントでのコマンド実行
を示します。
```

これらのシェルスクリプトの例では `PS2=" "` を使っています。 `bash` に関するより詳しい情報は ‘Bash – **GNU** 標準の対話シェル’ on page 221 を見てください。

- Unix スタイルな マニュアルページ への参照は `bash(1)` のような形式で与えられます。
- GNU の **TEXINFO** ページ への参照は `info libc` のような形式で与えられます。
- **書籍** への参照は *The C Programming Language* のような形式で与えられます
- **URL** への参照は `http://www.debian.org/doc/manuals/debian-reference/` のような形式で与えられます。
- システム上の ファイル への参照は `/usr/share/doc/Debian/reference/` のような形式で与えられます。

次の略語が使用されます。

- **LDP**: Linux Documentation Project (`http://www.tldp.org/`)
- **DDP**: Debian Documentation Project (`http://www.debian.org/doc/`)

他の略語は最初に用いられる前にテキスト内で定義されます。

本文書では LDP 文書に対しては URL しか示していませんが、LDP 文書は Debian でパッケージ化されています。 インストールすると LDP 文書は `/usr/share/doc/HOWTO/` にインストールされます。 ‘参考文献’ on page 247 をご覧下さい。

1.3 サンプルスクリプト

`debian-reference-ja` パッケージに収録されている 本文書に付属する Example scripts (`http://www.debian.org/doc/manuals/debian-reference/examples/`) は `/usr/share/doc/Debian/reference/examples/` で得られます。 隠しファイルは、ファイル名の前の “.” を “_” に変換してあります。 代替となるサンプルスクリプトがいくつかある場合は、さらに拡張子が追加してあります。

1.4 基本設定

システムに最小限のパッケージしかインストールしておらず、 本文書を最大限活用したい場合、有用な文書を含む 他のパッケージをインストールするために次のコマンドを 実行することをお薦めします。

```
# aptitude install mc less ssh vim kernel-package \
                   manpages-dev doc-debian doc-linux-text \
                   debian-policy developers-reference maint-guide \
                   apt-howto harden-doc debian-reference \
                   libpam-doc glibc-doc samba-doc exim4-doc-html
```

1.5 Debian ディストリビューションの基礎

Debian は 3 つの異なるディストリビューションを同時に維持しています。3 つのディストリビューションには、次のような特徴があります。

- `stable` — セキュリティ修正があった時にのみ更新されるため、サーバ製品で使用するのに最適。'`stable` ディストリビューション' on page 6 参照。
- `testing` — かなりのテストを受けた新しめのバージョンのデスクトップソフトウェアを収録するため、ワークステーションで使用するのに最適なディストリビューション。'`testing` ディストリビューション' on page 7 参照。
- `unstable` — 最先端を走っています。Debian 開発者向けです。'`unstable` ディストリビューション' on page 7 参照。

`unstable` にあるパッケージに release-critical (RC) なバグが無くなってから 1,2 週間経つと、自動的に `testing` に昇格します。

Debian ディストリビューションは、'Debian ディストリビューションのコードネーム' on page 8 に記述されているようなコードネームも持っています。Sarge が 2005 年 6 月にリリースされる前は、Debian の 3 つのディストリビューションはそれぞれ Woody (stable)、Sarge (testing)、そして Sid (unstable) でした。Sarge のリリース後は、それぞれ Sarge、Etch、そして Sid になりました。Etch がリリースされると、`stable` と `unstable` ディストリビューションはそれぞれ Etch と Sid になり、一方、新しい `testing` ディストリビューションが (通常 `stable` のコピーとして) 作成され、新しいコードネームを与えられます。

Debian に関する重要なアナウンスを受けるには、トラフィックが少ない `debian-devel-announce@lists.debian.org` メーリングリストを購読してください。

'Debian アーカイブ' on page 5 参照。

現在使用しているディストリビューションがリリースしたバージョンよりも新しいバージョンのパッケージを利用したい場合、'`stable`, `testing`, 又は `unstable` ディストリビューションへのアップグレード' on page 75 に記述しているようにより新しいディストリビューションに更新することもできますし、選択したパッケージのみを更新することもできます。そのパッケージが簡単にはアップグレードできない場合には、'`stable` システムへのパッケージ移植' on page 92 に記述しているようにバックポートしたくなるかもしれません。

`testing` ディストリビューションを追いかけることには、セキュリティ修正を含むパッケージのインストールが遅くなるという副作用があります。そのようなパッケージはまず `unstable` にアップロードされ、それから遅れて `testing` に移ってきます。

異なる Debian ディストリビューションを混ぜる、すなわち `stable` で `testing` や `unstable` の 一部を使った場合、結果として `libc6` のような コアパッケージを `testing` や `unstable` から 引っ張ってくることになり、これらには バグが無いという保証は 全くありません。警告しておきます。

`testing` や `unstable` ディストリビューションを使用 していると、深刻なバグに遭遇するリスクが高まります。このリスクは Debian のより安定なディストリビューションとのマルチブートを 実施することや、'`chroot`' on page 126 に記述されているように `chroot`を用いることにより管理できます。 後者の手段は異なった Debian ディストリビューションを 同時に異なった端末で起動できるようになります。

'Debian の基礎知識' on page 5 において Debian ディストリビューションの いくつかの基礎を説明した後、最新のソフトウェアを用いて楽しく過ごし、 Debian の `testing` や `unstable` ディストリビューション を利用するのを手助けするいくつかの基本的な情報を提供します。我慢できない人はすぐに 'Debian で生き残るためのコマンド' on page 85 に進んで下さい。 Happy upgrading!

章 2

Debian の基礎知識

本章は非開発者のために Debian システムの基礎的な情報を供給します。 信頼できる情報が欲しい場合は、次を参照してください。

- Debian ポリシーマニュアル
- Debian デベロッパーリファレンス
- Debian ニューメンテナーズガイド

'参考文献' on page 247 にリストされています。

あまり詳しくない "how-to" 的な説明を探している場合、 'Debian パッケージ管理' on page 79 や他の該当する章にすぐ飛んでください。

本章は "Debian FAQ" から取られた文書に基づき、通常の Debian システム管理者が始められるように大規模な再構成を行いました。

2.1 Debian アーカイブ

2.1.1 ディレクトリ構造

Debian 用にパッケージングされているソフトウェアは Debian mirror site (`http://www.debian.org/mirror/`) のそれぞれのいくつかの ディレクトリツリーの一つから FTP 並びに HTTP 経由で取得できます。

次のディレクトリは 各 Debian ミラーサイトの `debian` ディレクトリの下で見つかります。

`dists/`: このディレクトリには "distributions" が含まれており、これは 現在 Debian release および pre-release で得られるパッケージへの 標準的なアクセス手段として使用されます。いくつかの古いパッケージや `Contents-*.gz` 、そして `Packages.gz` も まだここにあります。

`pool/`: ここは Debian release および pre-release の全パッケージ が物理的に存在する場所です。

`tools/`: ブートディスク作成、ハードディスクのパーティショニング、 ファイル圧縮/解凍、そして Linux のブート用の DOS ユーティリティが ここにあります。

`doc/`: FAQ, バグ報告システム指示書などの基本的な Debian に関する 文書がここにあります。

`indices/`: Maintainers ファイルや override ファイルがここにあります。

`project/`: 主に開発者のみ必要なものです。例えば、

`project/experimental/`: 本ディレクトリにはまだ開発中であり、まだ α テスト段階の パッケージやツールが含まれます。ユーザはここにあるパッケージを 使用するべきではありません。なぜなら、最も経験を積んだ人でさえも 危険で害を及ぼす可能性があるからです。

`project/orphaned/`: 古いメンテナにより「みなしご化」され、ディストリビューションから なくなったパッケージです。

2.1.2 Debian ディストリビューション

通常 `dists` ディレクトリには 3種類の Debian ディストリビューションが存在します。これらは `stable`. `testing` そして `unstable` と呼ばれます。 時々 `frozen` も存在します。(現在は`frozen`は`testing` の開発ステージのひとつです。)各ディストリビューションは `dists` ディレクトリにある、コードネームが付いた 実際のディレクトリのシンボリックリンクとして定義されています。

2.1.3 `stable` ディストリビューション

`stable` ディストリビューション, Debian Sarge (3.1r0) 用のパッケージ エントリは `stable` (`sarge/` への シンボリックリンク) ディレクトリに記録されています。

- `stable/main/`: このディレクトリには Debian システムの最新の公式リリースに属する バージョンのパッケージが含まれます。

  これらのパッケージは全てフリーです。すなわち、これらは全て Debian Free Software Guidelines (`http://www.debian.org/social_contract#guidelines`) (DFSG) に適合しています。(`debian-doc` により インストールされる `/usr/share/doc/debian/social-contract.txt` としても 得られます。)

- `stable/non-free/`: このディレクトリには DFSG に従ってフリーと認定されない パッケージが含まれます。

  例えば、いくつかのパッケージは商用配布を禁止するライセンスを持っています。 また、あるパッケージは再配布可能ですが、実はシェアウェアです。

- `stable/contrib/`: このディレクトリには それら自体は DFSG-free ですが、DFSG-free ではない パッケージになぜか依存しているパッケージです。

現在は、上に挙げた場所に加えて、`pool` ディレクトリに新しい物理的なパッケージがあります。 (‘`pool` ディレクトリ’ on page 9 参照)

`stable` ディストリビューションの最新バグステータスは Stable Problems (`http://ftp-master.debian.org/testing/stable_probs.html`) web ページに報告されています。

2.1.4 **`testing`** ディストリビューション

`testing` ディストリビューション、Debian Etch のパッケージエントリは `unstable` でしばらくテストを受けた後に `testing` (`etch/` へのシンボリックリンク) ディレクトリに記録されます。 上に挙げた場所に加え、`pool` ディレクトリに新しい物理的なパッケージがあります。 (‘`pool` ディレクトリ’ on page 9 参照) `testing/` に `stable/` と同じ機能を提供する `main`、`contrib` そして `non-free` サブディレクトリもあります。

これらのパッケージはビルドされる全てのアーキテクチャで同期を取られ、インストール可能でなくてはなりません。また、`unstable` にあるバージョンよりも release-critical bug が少なくてはなりません。 このように、`testing` は常に release candidate 間近であることを希望しています。testing のメカニズムの詳細については `http://www.debian.org/devel/testing` にあります。

`testing` ディストリビューションの最新バグステータスはこれらのサイトで報告されています。

- update excuses (`http://ftp-master.debian.org/testing/update_excuses.html`)
- testing problems (`http://ftp-master.debian.org/testing/testing_probs.html`)
- release-critical bugs (`http://bugs.debian.org/release-critical/`)
- base system bugs (`http://bugs.qa.debian.org/cgi-bin/base.cgi`)
- bugs in standard and task packages (`http://bugs.qa.debian.org/cgi-bin/standard.cgi`)
- other bugs and bug-squashing party notes (`http://bugs.qa.debian.org/`)

2.1.5 **`unstable`** ディストリビューション

`unstable` ディストリビューションは、常に “Sid” というコードネームであり、パッケージエントリは Debian archive にアップロードされた後に `unstable` (`sid/` へのシンボリックリンク) ディレクトリに記録され、`testing/` に移動されるまでここにあります。 上に挙げた場所に加え、`pool` ディレクトリに新しい物理的なパッケージがあります。 (‘`pool` ディレクトリ’ on page 9 参照) `testing/` に `stable/` と同じ機能を提供する `main`, `contrib`, そして `non-free` サブディレクトリもあります。

`unstable` ディストリビューションには最新の開発版システムのスナップショットが含まれます。ユーザがパッケージを使ってテストするのは歓迎されますが、これらのパッケージが準備段階にあることは警告されます。 `unstable` ディストリビューションを使う利点としては、常に最新の Debian ソフトウェアプロジェクトを使えます。 — たとえシステムを壊すようなことがあったとしても、壊れたシステムをそのままキープできます。

`unstable` ディストリビューションの最新バグステータスは Unstable Problems (`http://ftp-master.debian.org/testing/unstable_probs.html`)web ページで 報告されています。

2.1.6 `frozen` ディストリビューション

`testing` ディストリビューションが充分成熟すると、frozen されます。 そして、新しいコードはもはや受け付けず、必要ならばバグフィックスのみ 受け付けられます。さらに、新しい testing ツリーが `dists` ディレクトリに作成され、新しいコードネームを割り当てられます。 frozen ディストリビューションは数ヵ月テストされ、“テストサイクル” と呼ばれる deep freeze に入ります。

パッケージのリリースを遅らせたり、リリース全体を止めてしまう frozen ディストリビューションのバグを記録し続けています。 バグ総数が最大許容数を下回ったら、frozen ディストリビューション は `stable` になり、リリースされます。そして以前の stable ディストリビューションは obsolete になります。(そして archive に移動します)

2.1.7 Debian ディストリビューションのコードネーム

`sarge/` や `etch/` などの、`dists` ディレクトリにある物理的に存在するディレクトリ名は、 単なる “コードネーム” です。Debian ディストリビューションが開発段階に あるとき、バージョン番号を持ちませんが、その代わりコードネームを持ちます。 これらのコードネームの目的は、Debian ディストリビューションのミラーリングを より簡単に行うためです。(もし真のディレクトリ名`unstable`が 突然`stable` に変わると、不必要に多くの ダウンロードを再び行わなくてはならなくなります)

現在、`stable` は `sarge/` の、 `testing` は `etch/` のシンボリック リンクです。これは `Sarge` が現在の stable ディストリビューションであり、`Etch` が現在の testing ディストリビューションであることを意味しています。

Sid は常に unstable ディストリビューションですので、`unstable/` は `sid/` の永続的なシンボリックリンクです。

2.1.8 過去に使用されたコードネーム

過去に使われたコードネームは次の通りです。 release 1.1 では “Buzz”, release 1.2 では “Rex”, release 1.3.x では “Bo”, release 2.0 では “Hamm”, release 2.1 では “Slink”, release 2.2 では “Potato”, release 3.0 では “Woody”, release 3.1 では “Sarge”。

2.1.9 コードネームの由来

いままではコードネームは Pixar 作の映画 トイストーリーの キャラクター名から取られていました。

- **Buzz** (Buzz Lightyear) は宇宙飛行士です。

- **Rex** はチラノザウルスです。
- **Bo** (Bo Peep) は羊飼いの娘です。
- **Hamm** Hamm は豚の形の貯金箱です。
- **Slink** (Slinky Dog) おもちゃの犬です。
- **Potato** はもちろん、ミスターポテトです。
- **Woody** はカウボーイです。
- **Sarge** はグリーンプラスティックアーミーのリーダーです。
- **Etch** (Etch-a-Sketch) はおもちゃの黒板です。
- **Sid** は隣に住んでいる、おもちゃを壊す男の子です。

2.1.10 `pool` ディレクトリ

歴史的に、パッケージはパッケージを含むディストリビューションに対応した `dists` サブディレクトリに保持されました。これは大きな変更が 発生した場合、ミラーするのに大きなバンド幅を消費するなどのさまざまな 問題を引き起こしました。

パッケージは現在大きな “pool” に保存され、source パッケージの名前に 従って構造化されます。この “pool” を管理可能にするため、pool は セクション (`main`, `contrib`, そして `non-free`) および source パッケージの先頭の文字により分割されます。これらのディレクトリ には、各アーキテクチャ用のバイナリパッケージ、そして バイナリパッケージの生成元である source パッケージなどが含まれます。

`apt-cache showrc` *mypackagename* のようなコマンドを 実行し、“Directory:” 行を見ることにより、パッケージの場所を見つけられます。 例えば、`apache` パッケージは `pool/main/a/apache/` にあります。非常に多くの `lib*` パッケージが存在するため、これらのパッケージは特別扱いされています。 例えば、`libpaper` パッケージは `pool/main/libp/libpaper/` に保存されています。

`dists` ディレクトリは依然 `apt` のような プログラムにより使用される索引ファイルのために使われています。

新しい `apt` や多分古い `dpkg-ftp` はシームレスな処理を行うので、通常は これらのことを心配する必要はありません。詳細を知りたい場合は、 RFC: implementation of package pools (`http://lists.debian.org/debian-devel-announce/2000/debian-devel-announce-200010/msg00007.html`) をご覧ください。

2.1.11 Sid に関するヒストリカルノート

昔 Sid は存在していませんでした。Debian archive サイトは大きな欠陥を 1つ持っていました。アーキテクチャが最新の `unstable` に作られると、ディストリビューションが新しい `stable` になった時に それがリリースされるという前提がありました。このケースにあてはまらない 多くのアーキテクチャにとって、リリース時にこれらのディレクトリが移動 しなければならないという結果となりました。この移動が膨大なバンド幅を 消費するため、これは実際的ではありません。

アーカイブ管理者はこの問題に数年間取り掛かり、`sid` と 呼ばれる特別なディレクトリに未リリースのアーキテクチャのためのバイナリを置く ことにより問題を解決してきました。未

リリースのアーキテクチャが最初にリリースされる 場合、最新の `stable/` に対し `sid/` がリンク されました。アーキテクチャが最初にリリースされると 現在の `stable/` から `sid/`にリンクを作り、それ以降は 通常どおり`unstable/`のツリーの中につくられていきました。 この配置はユーザーの混乱を招きやすいです。

Woody ディストリビューションの開発中にパッケージプールを (‘`pool` ディレクトリ’ on page 9 参照) 発明したことにより、バイナリパッケージは ディストリビューションに依らずにプール内の標準的な場所に保存され始めました。 それゆえ、ディストリビューションのリリースを行っても、もはやミラー時に 大きなバンド幅を消費しません。(しかし、開発過程を通し、緩やかなバンド幅 消費が発生します。)

2.1.12 `incoming/` にパッケージをアップロードする

パッケージのアップロードはそれらが本当に Debian 開発者からのものかを 検査した後に `http://incoming.debian.org/` にまず置かれます。 (そしてそれがノンメンテナアップロード (NMU) の場合は、`DELAYED` サブディレクトリに置かれます。) 1日に 1度、それらは `incoming/` から `unstable/` に移されます。

緊急時には、`incoming/` が `unstable/` に移る前に パッケージをインストールしたい場合があるかもしれません。

2.1.13 古いパッケージを取得する

最新の Debian ディストリビューションは Debian mirror site (`http://www.debian.org/mirror/`) の `debian` ディレクトリ下に保持されますが、Slink などの古い Debian ディストリビューションのアーカイブは `http://archive.debian.org/` や Debian の各ミラーサイトの `debian-archive` ディレクトリに保存されています。

`testing` や `unstable` の昔のパッケージは `http://snapshot.debian.net/` にあります。

2.1.14 アーキテクチャセクション

主要なディレクトリツリーそれぞれ (`dists/stable/main`, `dists/stable/contrib`, `dists/stable/non-free`, `dists/unstable/main` など) の中に、バイナリパッケージエントリが パッケージがコンパイルされたアーキテクチャを示す名前を持つサブディレクトリ 内に存在します。

- `binary-all/`: アーキテクチャに依存しないパッケージ用。 これらには例えば、Perl スクリプトや、純粋なドキュメントが含まれます。
- `binary-`*`platform`*`/`: 特定のバイナリ プラットフォームで実行するパッケージ用。

`testing` と `unstable` 用の実際のバイナリパッケージは もはやこれらのディレクトリに無く、`pool` ディレクトリに あることに注意してください。しかし、索引ファイル (`Packages` と `Packages.gz`) は下位互換性のために保持されています。

実際にサポートされているバイナリパッケージについては、各ディストリビューションのリリースノートをご覧ください。これらは stable (`http://www.debian.org/releases/stable/releasenotes`) および testing (`http://www.debian.org/releases/testing/releasenotes`)のためのリリースノート サイトにあります。

2.1.15 ソースコード

ソースコードは Debian システムにおける全てに対して含まれます。さらに、システムのほとんどのプログラムのライセンス事項は、ソースコードがプログラムと共に配布されるか、プログラムに添付するソースコードを提供することを要求します。

通常ソースコードは `source` ディレクトリで配布され、アーキテクチャ特有のバイナリディレクトリ全てと並列になっているか、より最近では `pool` ディレクトリにあります。 (‘`pool` ディレクトリ’ on page 9 参照) Debian アーカイブの構造を熟知せずにソースコードを取得するには、`apt-get source` *`mypackagename`* のようなコマンドを試してみてください。

いくつかのパッケージ、とりわけ `pine` はそのライセンスの制限により、ソースパッケージでしか得られません。 (最近、`pine-tracker` パッケージが Pine のインストールを容易にするために導入されました。‘`stable` システムへのパッケージ移植’ on page 92 や ‘パッケージング’ on page 240 に記述された手順により、パッケージを手動で構築する方法を習得できます。

公式には Debian システムの一部では無い `contrib` や `non-free` ディレクトリにあるパッケージのソースコードは入手できるかもしれませんし、できないかもしれません。

2.2 Debian パッケージ管理システム

2.2.1 Debian パッケージの概観

一般にパッケージには関連するコマンドや機能を実装するのに必要なファイルすべてが含まれています。Debian パッケージには 2つのタイプがあります。

- バイナリパッケージ。これには実行ファイル、設定ファイル、man ページと info ページ、著作権情報やその他の文書が含まれます。これらのパッケージは Debian 固有のアーカイブ形式で配布されています。(‘Debian パッケージのフォーマット’ on page 12参照) また .deb というファイル拡張子を持っています。バイナリパッケージは Debian ユーティリティ `dpkg` を用いて展開できます。詳細は `dpkg` のマニュアルページに記載されてあります。
- ソースパッケージ。ソースパッケージの解説が書かれた `.dsc` ファイル (このファイルには以下のファイルの名前も書かれています) や、修正されていないオリジナルのプログラムソースが gzip で圧縮された tar フォーマット形式で含まれている `.orig.tar.gz` ファイル、通常 Debian 固有の変更を記した `.diff.gz` ファイルから構成されています。`dpkg-source` ユーティリティは Debian ソースアーカイブをパックしたりアンパックしたりします。詳細は`dpkg-source` のマニュアルページに記載されています。

このパッケージシステムでは、ソフトウェアをインストールするとき、パッケージ保守担当者により宣言された “依存情報” を使います。この依存情報はそれぞれのパッケージに関連する 制御(`control`) ファイルに記載されています。例えば、GNU C コンパイラ (gcc) を含むパッケージは、リンカやアセンブラを含む `binutils` パッケージに “依存”しています。もしユーザがあらかじめ `binutils` をインストールしていないのに `gcc` をインストール>しようとしたなら、Debian のパッケージシステムは `binutils` も必要であるというエラーメッセージを出力し、ユーザがまず `binutils` をインストールするのに同意するまで `gcc` をインストールしません (とは言うものの、頑固なユーザは この機能を上書きできます。`dpkg(8)` 参照) さらに詳しい情報は、‘パッケージの依存性’ on page 16 下をご覧ください。

Debian のパッケージングツールは以下の用途に使えます。

- パッケージやパッケージの一部を操作したり管理したりする。
- フロッピーディスクなどの限られたサイズの媒体を通じて 輸送しなければならないパッケージをユーザが分解するのを助ける。
- パッケージアーカイブを開発者が構築するのを助ける。
- 遠隔の Debian アーカイブサイトに置かれたパッケージをユーザが インストールするのを助ける。

2.2.2 Debian パッケージのフォーマット

Debian の “パッケージ” つまり Debian アーカイブファイルには、実行プログラム一式や関連するプログラムのセットに関係する実行ファイルや、ライブラリ、ドキュメントが含まれています。通常、Debian アーカイブファイルは ファイル名の最後に `.deb` が付いています。[1]

Debian バイナリパッケージフォーマットの内部仕様は `deb(5)` マニュアルページに解説されています。このフォーマットは (Debian のメジャーリリースの間で) 変更されることがあるので、`.deb` ファイルを操作する時は必ず `dpkg-deb(1)` を使って下さい。

少なくても Sarge ディストリビューションを通じ、全ての Debian アーカイブ ファイルは、`dpkg` コマンドが使えない場合であっても、標準的な Unix コマンドである `ar` や `tar` により操作できます。

---

[1] `debian-installer` プロジェクトは `.udeb` で終わる ファイル名のパッケージを導入しました。要するに、マイクロ `.deb` フォーマットは ポリシーに正しくは従う必要はなく、ドキュメントなどの 内容を欠いており、`debian-installer` だけに用いられることを 意味しています。この新しい Debian インストーラは Sarge のリリースのために 初めて開発されました。`.deb` と `.udeb` のファイルフォーマット は同じ物です。`udpkg` プログラムは 標準の `dpkg` より制限された能力を持つ `.udeb` パッケージを操作するために使われ、より少ないパッケージの関係をサポートします。名前に違いがあるのは、Debian アーカイブのメンテナはポリシーに従わないアーカイブに `.deb` という名前を持つのを好まなかったからです。それゆえ、2つのフォーマットの違い を目立たせ、本当のシステムに `.udeb` ファイルを無意識にインストール しないように名前を変更しました。`.udeb` は本当に制限された Debian システムを作成するため、base システムをインストールする間に最初のラムディスク 上で使用されます。

2.2.3 Debian パッケージ名の命名規則

Debian パッケージのファイル名は次のような規則に従います。

```
foo_バージョン番号-リビジョン番号_アーキテクチャ.deb
```

ここで、通常 *foo* はパッケージ名であり、*var* は 上流のバージョン番号、*rev* は Debian でのリビジョン番号、そして *arch* はターゲットのアーキテクチャです。 ファイルはもちろん簡単に改名できます。 次のコマンドを実行することにより、*filename* という名前の ファイルに含まれているパッケージを調べることができます。

```
dpkg --info filename
```

Debian のリビジョン番号は Debian 開発者又はパッケージを構築した人 により指定されます。通常リビジョン番号の変更はパッケージン グの観点から 変更が加えられたことを意味しています。

2.2.4 ローカル設定の保存

ローカルの管理者により変更可能とされているファイルは `/etc/` に保持されています。Debian ポリシーはローカルの設定可能なファイルへの全ての変更が パッケージのアップグレードを通じて保持されるべきであることを指示しています。

ローカルの設定可能なファイルの標準のバージョンがパッケージ自身に 含まれている場合、そのファイルは “conffile” としてリストされます。 パッケージは管理者の許可無しで最後にインストールされたので、 パッケージ管理シ ステムは管理者により変更された conffile を更新しません。 一方、conffile が管理者により変更されていないと、パッケージの他のファイル と一緒に conffile もアップグレードされます。 これはほとんど常に好ましく、conffile への変更を最小限にすることは好都合です。

パッケージに所属する conffile をリストするには、次のコマンドを実行します。

```
dpkg --status package
```

このリストの “Conffiles:” 行に conffile が表示されます。

conffile に関するより詳しい情報については、Debian ポリシーマニュアルの “Configuration files” という章を読んでください。(‘参考文献’ on page 247 参照)

2.2.5 Debian メンテナンススクリプト

Debian メンテナンススクリプトはパッケージがインストールされる前か後で 自動的に実行される実行可能なスクリプトです。これらのファイルは `control` という名前のファイルと一緒に全て Debian アーカイブファイルの “制御 (control)” セクションの一部となっています。

個々のファイルは以下の通りです。

preinst このスクリプトは、パッケージが Debian アーカイブ (`.deb/`) ファイルからアンパックされる前に実行されます。パッケージがインストールかアップグレードし終わる (“postinst” スクリプトが正常に実行された後) まで、多くの“preinst” スクリプトの中で、更新されるパッケージのために サービスが停止されるようになっています。

postinst このスクリプトの典型的な仕事は、Debian アーカイブファイル (`.deb/`) からアンパックされたら、それに必要な設定をすべて完了させる ことです。 “postinst” スクリプトのよくある動作として、ユーザに入力を求め、既定値を受け入れるなら後戻りしてこのパッケージを環境に沿うように 再設定することを忘れないように警告を表示します。 新しいパッケージがインストールされるかアップグレードされると、 多くの “postinst” スクリプトはサービスを開始または再開するのに 必要なコマンドをすべて実行します。

prerm このスクリプトは、典型的にはパッケージに関連したあらゆる デーモンを停止します。これはパッケージに関連したファイルを 削除する前に実行されます。

postrm このスクリプトは、典型的にはパッケージに関連したリンクや 他のファイルを修整したりパッケージが作成したファイルを削除したりします。 (‘仮想パッケージ’ on page 15 も参照。)

現在、制御ファイルは全て `/var/lib/dpkg/info/` に置かれています。パッケージ `foo` に関係するファイルは “foo” で始まる 名前を持ち、“preinst” や “postinst” などの適当なファイル拡張子を持ちます。 このディレクトリにある `foo.list` というファイルは、 パッケージ `foo` によってインストールされたファイルがすべて リストされています (これらのファイルの存在場所は `dpkg` が 内部に持っていることに注意して下さい。存在場所を頼りにしないほうがいいでしょう)。

2.2.6 パッケージ優先度

それぞれの Debian パッケージには、パッケージ管理システムの助けとして、 ディストリビューション保守担当者が **優先度** を割りあてています。 優先度には以下のものがあります。

- **Required (要求)** パッケージはシステムを正しく 動作させるために必要なパッケージです。

  システムの欠陥を修復するのに必要なツールをすべて含みます。 これらのパッケージを消去してはいけません。システムがすっかり破壊され、 復旧 するために `dpkg` を使うことすら恐らくできなくなります。 Required パッケージだけのシステムは恐らく使いものになりませんが、 システム管理者が起動したり他のソフトウェアをインストールするだけの 機能はあります。

- **Important (重要)** パッケージはどのような Unix ライクなシステムにもあるべきパッケージです。

  Required 以外のパッケージで、無いとシステムがうまく動かなかったり 不便だったりするものにこの優先度がつけられています。これには Emacs や X11、TeX 他の巨大な

アプリケーションは含まれていません。ここのパッケージは、素のインフラストラクチャを構成するだけです。

- **Standard (標準)** パッケージはどんな Linux システムにも標準的なパッケージで、手頃な小ささですが機能が限定されすぎていないキャラクタモードシステムを含んでいます。

  これはユーザが何も指示しなかったらデフォルトでインストールされます。多くの巨大なアプリケーションは含まれませんが、Emacs はあります (これはアプリケーションというよりも多くのプログラムのための インフラストラクチャの一部です) し、TeX と LaTeX の手頃なサブセットが (X なしで稼動可能な部分なら) 含まれています。

- **Optional (任意)** パッケージにはそのパッケージが何なのかを知らなかったり、そのパッケージを使わなければならない特別な要求がなかったりしてもインストールしてかまわないパッケージが含まれています。

  これには X11 や TeX 配布物全体、多くのアプリケーションが含まれています。

- **Extra (付録)** パッケージはより高い優先度を持つ パッケージと衝突するか、そのパッケージがどういうものか知っているか、あるいは “任意” というには相応しくない特殊な要求を持っているパッケージです。

パッケージの説明にある “Priority: required”, “Section: base” と “Essential: yes” の違いについて注意してください。“Section: base” は、新しいシステムでは他の全てをインストールする前に、このパッケージがインストールされることを意味しています。“Section: base” なパッケージの ほとんどは “Priority: required” または、少なくても “Priority: important” であり、それらの多くは “Essential: yes” にタグづけられています。“Essential: yes” は、このパッケージをシステムから削除するには、`dpkg` のようなパッケージ管理システムに特別な強制オプションを 指示する必要があることを意味しています。例えば、`libc6`、`mawk` や `makedev` は “Priority: required” かつ “Section: base” ですが、“Essential: yes” ではありません。

2.2.7 仮想パッケージ

仮想パッケージとは、すべて同じ基本機能を提供するパッケージの集まりの どれか一つに供される一般的な名前のことです。例えば、`tin` と `trn` プログラムはどちらも ニュースリーダであり、それゆえ、動作するか利用するためにニュースリーダを 要求するプログラムの依存性を満たします。したがって、両プログラムは `news-reader` と呼ばれる “仮想パッケージ” を供給します。

同様に、`exim` 、`exim4`、`sendmail`、`postfix` のような多くのパッケージは メール配送エージェント (mail transport agent) の機能を備えているために 仮想パッケージ `mail-transport-agent` を 提供すると言われます。これらのうちどれかがインストールされていれば、mail transport agent がインストールされていることに依存する プログラムはどれでもこの仮想パッケージが存在しているために 条件を満足しています。

Debian はこのようなしくみを提供するので、同じ 仮想パッケージを持つ パッケージが 1つ以上システムにインストールされると、システム管理者は 優先パッケージを設定できます。関

連するコマンドは `update-alternatives` で、'Alternative コマンド' on page 96 で さらに詳しく述べられています。

2.2.8 パッケージの依存性

Debian パッケージン グシ ステムは、あるパッケージが機能したり、うまく 動くようにインストールされるべき他のパッケージを要求するという事実を 表現するために使われる依存性に関する宣言を操作します。

- A を使うために B が絶対インストールされなければならない場合、 パッケージ A はパッケージ B に **Depends** すると表現します。 いくつかの場合、A は B だけでなく、B の特別なバージョンに依存 します。この場合、バージョンの依存性は、A がある指定したバージョン 以降のバージョンの B に依存するという意味で、通常下限を表します。
- ほとんどのユーザは B により供給される機能も持たない A を 欲しいとは思わないとパッケージのメンテナが判断した場合、 パッケージ A はパッケージ B を **Recommends** すると 表現します。
- A に関係し、A の機能を強化するファイルが B に含まれている場合、 パッケージ A はパッケージ B を **Suggusts** すると表現します。 同じ関係は、パッケージ B がパッケージ A を **Enhances** すると表現します。
- B がシ ステムに インストールされていると A が適切に動作しない場合、 パッケージ A はパッケージ B と **Conflicts** すると 表現します。"Conflicts" ステータスはたびたび "Replaces" と置き換えられます。
- B によりインストールされるファイルが A にあるファイルにより 削除あるいは上書きされる場合、パッケージ A はパッケージ B を **Replaces** すると表現します。
- B の全てのファイルと機能が A にも組み込まれている場合、 パッケージ A はパッケージ B を **Provides** すると 表現します。

これらの用語の使用法についてのより詳しい情報は *Debian* パッケージン グ マニュアル と *Debian* ポリシーマニュアル にあります。

Depends と指定されたパッケージ全てを単に取得しますが、 **Recommends** や **Suggests** と指定された パッケージを全て無視する `apt-get` に比べ、`aptitude`や `dselect` は、**Recommends** や **Suggests** により指定されるより洗練されたパッケージ制御機能を有します。これらのプログラムは、共に現代的な形で APT をバックエンドとして使用します。

2.2.9 "pre-depends" の意味

`dpkg` は、常に他のパッケージが依存しているパッケージを 先に設定します。しかしながら、`dpkg` は通常ファイルを任意の順番で、依存性と関係なく解凍します。 (解凍とは、アーカイブファイルからのファイルの展開とそれらのファイルを 正しい場所に置くことから構

成されます。) しかしながら、もしパッケージが他のパッケージに **Pre-Depends** している場合、Pre-Depends しているパッケージが解凍されていても、 その前に Pre-Depends されているパッケージの解凍と設定が行われます。[2] この依存性の使用は最小限に保たれています。

2.2.10 パッケージステータス

パッケージステータスには “unknown”, “install”, “remove”, “purge”, “hold” があります。 これらの “want” フラグは利用者がそのパッケージをどう扱いたいかを示しています。 利用者は `dselect` の “Select” セクションでのアクション や `dpkg` の直接起動によってこれを示すことができます。

それぞれの意味は以下の通りです。

- **unknown** - インストールするかどうかユーザが 表明していないパッケージ
- **install** - ユーザがインストールまたはアップグレード したいパッケージ
- **remove** - 削除はしたいが、既存の設定ファイルは 一つも削除したくないパッケージ
- **purge** - 設定ファイルを含め、完全に削除してしまう パッケージ
- **hold** - 処理はしない、つまりどのような状態 であれ現在の状態で現在のバージョンを維持するパッケージ

2.2.11 更新からパッケージを hold する

パッケージを hold するには二つの方法があります。`dpkg` を使う方法と、Woody から始まった APT を使う方法です。

`dpkg` では、パッケージ選択の一覧を

```
dpkg --get-selections \* > selections.txt
```

で書き出すだけです。それから書き出されたファイル *`selections.txt`* を編集して hold したいパッケージの 行を変更します。例えば、

```
libc6                          install
```

を

```
libc6                          hold
```

にします。ファイルを保存して、

```
dpkg --set-selections < selections.txt
```

---

[2] このメカニズムはパッケージを展開する **順番**が非常に重要 であった `a.out` フォーマットから ELF フォーマットへのシステムの 安全な更新をサポートするために供給されました。

でdpkg データベースに再ロードしてください。又、hold したいパッケージ名 を知っている場合は、単に

```
echo libc6 hold | dpkg --set-selections
```

を実行するだけです。この手順は各パッケージファイルの インストール処理で パッケージを hold します。

同様の効果が `dselect` を通しても得られます。[S]elect 画面に 入って hold したいパッケージの現在の状態を確認し、「=」キー (もしくは「H」キー) を押下するだけです。変更は [S]elect 画面を終了するとすぐに反映します

Woody ディストリビューションにおける APT システムは `Pin-Priority` を用いてアーカイブ取得処理中にパッケージを hold する新しいもう一つの機構 を持ちます。 `http://www.debian.org/doc/manuals/apt-howto/` 又は `apt-howto` パッケージのに、マニュアルページ `apt_preferences(5)` をご覧ください。

2.2.12 ソースパッケージ

ソースパッケージは `source` と呼ばれるディレクトリで 配布されています。手動でダウンロードできますし、

```
apt-get source foo
```

を使ってダウンロードもできます。 (このための APT の設定法については apt-get(8) マニュアルページを 参照してください。) apt-get source foo を使って取得することもできます (このための APT の設定法については `apt-get(8)` を参照願います。)

2.2.13 ソースパッケージからバイナリパッケージを作る

パッケージ *foo* をソースからコンパイルするには、 *foo_**`.dsc`, *foo_**`.tar.gz` および *foo_*.diff*`.gz` の全てが必要となります (注意:Debian 固有のパッケージに は `.diff.gz` はありません)。

これらを入手し、`pkg-dev` パッケージをインストールしている場合、

```
$ dpkg-source -x foo_version-revision.dsc
```

というコマンドを実行すれば *foo-version* というディレクトリ にパッケージが取り出されます。

バイナリパッケージをコンパイルしたいならば、

```
$ cd foo-version
$ su -c "apt-get update ; apt-get install fakeroot"
$ dpkg-buildpackage -rfakeroot -us -uc
```

を実行し、

```
# su -c "dpkg -i ../foo_version-revision_arch.deb"
```

で新たに構築したパッケージをインストールします。'`stable` システムへのパッケージ移植' on page 92 をご覧ください。

2.2.14 新しい Debian パッケージを作る

新しいパッケージを作ることに関する詳細な情報は、`maint-guide` パッケージで得られる *Debian* メンテナ入門 又は `http://www.debian.org/doc/manuals/maint-guide/` をご覧ください。

2.3 Debian システムのアップグレード

Debian の目標の一つは首尾一貫したアップグレード方針と安全な アップグレード手順を提供することです。パッケージング システムは 管理者に重要な変更について警告し、時々は管理者に決定を促します。 リリースノートも読むべきです。これは全ての Debian CD と一緒に 出荷されており、WWW でも `http://www.debian.org/releases/stable/releasenotes` や `http://www.debian.org/releases/testing/releasenotes` で利用可能です。

アップグレードを行う実際的なガイドは 'Debian パッケージ管理' on page 79 で供給されます。 本章では単に概要を供給します。まずはパッケージングツールから始めます。

2.3.1 dpkg

これはパッケージファイルの操作のための主要なプログラムです。 完全な説明は `dpkg(8)` を読んでください。

`dpkg` にはいくつかの原始的な補助プログラムが付随します。

- `dpkg-deb`: `.deb` ファイルを操作します。 `dpkg-deb(1)`
- `dpkg-ftp`: 旧型のパッケージファイル取得コマンドです。 `dpkg-ftp(1)`
- `dpkg-mountable`: 旧型のパッケージファイル取得用コマンドです。 `dpkg-mountable(1)`
- `dpkg-split`: 大規模なパッケージを小さなファイルに分割します。 `dpkg-split(1)`

`dpkg-ftp` と `dpkg-mountable` は APT システムに 取って代わられました。

2.3.2 APT

APT (the Advanced Packaging Tool) は Debian パッケージング システムの 先進的なインターフェースであり、“apt-” で始まる名前を持ついくつかのプログラム から構成されています。`apt-get`, `apt-cache`, と `apt-cdrom` はパッケージ操作用のコマンドラインツールです。これらは、`dselect` や `aptitude` のような他のツールへのユーザ “バックエンド” としても働きます。 現在、`aptitude`は、システム維持のための推奨ツールです。

より詳しい情報は、`apt` パッケージと `aptitude` パッケージをインストールして、 `aptitude(8)`, `apt-get(8)`, `apt-cache(8)`, `apt-cdrom(8)`, `apt.conf(5)`, `sources.list(5)`,`apt_preferences(5)` を読んでください。

もう一つの情報源としては、 APT HOWTO (`http://www.debian.org/doc/manuals/apt-howto/`) があります。 `/usr/share/doc/Debian/apt-howto/` の `apt-howto` によりインストールできます。

`apt-get upgrade` と `apt-get dist-upgrade` は “Depends:” にリストされたパッケージのみを引っ張ってきますが、 “Recommends:” や “Suggests:” にリストされたパッケージは無視します。 これを避けるには、`dselect` を御使用ください。

2.3.3 `dselect`

このプログラムは Debian パッケージ管理システムへの メニュドリブンな ユーザインターフェースです。最初のインストール時や大規模なアップグレード時 に特に役立ちます。‘`dselect`’ on page 82 をご覧ください。

より詳しい情報は、`install-doc` パッケージをインストールし、 `/usr/share/doc/install-doc/dselect-beginner.en.html` や dselect Documentation for Beginners (`http://www.debian.org/releases/woody/i386/dselect-beginner`) を読んでください。

2.3.4 起動中のシステムをアップグレードする

Debian システムにおける kernel (ファイルシステム) はファイルを使用中 でさえも、そのファイルの置き換えをサポートしてます。 現在のランレベルで起動されるようにパッケージが設定されている場合、 パッケージの更新時にパッケージにより供給されるサービスが再起動 されます。 Debian システムは起動中のシステムをアップグレードするのにシングルユーザモードを必要としません。

2.3.5 `.deb` アーカイブファイルのダウンロードとキャッシュ

手動でパッケージファイルをディスクにダウンロードした場合、(これは必ずしも必要ではありません。上に記述した `dpkg-ftp` または APT の説明をご覧ください) パッケージをインストールした後、 システムから `.deb` ファイルを削除できます。

APT を使っている場合、これらのファイルは `/var/cache/apt/archives` ディレクトリにキャッシュされます。これらをインストール後に削除 (`apt-get clean`) できますし、その後のインストール中のダウンロード時間を節約するため、他のマシーンの `/var/cache/apt/archives` ディレクトリにコピーしてもかまいません。

2.3.6 アップグレードの記録を取る

`dpkg` は展開、設定、削除またはパージされたパッケージの記録を取りますが、(現在) パッケージにそのような操作が行われている間に起こったターミナル上の コマンドログを取っていません。

コマンドログを取る最もシンプルな方法は、`dpkg`, `dselect`, `apt-get` などのセッションを `script(1)` プログラム内で起動することです。

2.4 Debian ブートプロセス

2.4.1 `init` プログラム

他の Unix ライク OS と同様に、Debian は `init` プログラムを 実行することによりブートを始めます。`init` 用の設定ファイル (`/etc/inittab`) は、実行されるべき最初のスクリプトが `/etc/init.d/rcS` であることを指定しています。

次に起きることは `sysv-rc` 又は `file-rc` のどちらかがインストールされているかどうかに依存しています。次に述べることは `sysv-rc` がインストールされていることを 仮定しています。(`file-rc` には固有の `/etc/init.d/rcS` スクリプトが含まれており、ランレベル毎に 起動されるサービスの種類を制御するために rc ディレクトリにある シンボリックリンクの代わりにこのスクリプトが用いられます。)

`sysv-rc` パッケージにある `/etc/init.d/rcS` ファイルはファイルシステムのチェックやマウント、モジュールの読み込み、ネットワークサービスの開始、時計の設定などの初期化作業を行うために `/etc/rcS.d/` にあるスクリプト全てを起動します。そして、互換性のために、`/etc/rc.boot/` 下にあるファイル (ファイル名に '.' が付くファイルを除く) も実行します。後者のディレクトリ内のスクリプトは、通常システム管理者が使用するために 予約されており、パッケージがこのディレクトリにスクリプトを置くのは 時代遅れです。詳細は Debian ポリシーマニュアルの 'システムの初期化' on page 131 や System run levels and init.d scripts (`http://www.debian.org/doc/debian-policy/ch-opersys#s-sysvinit`) をご覧ください。

2.4.2 ランレベル

ブートプロセス完了後、`init` は (`/etc/inittab` の `id` 用のエントリにより与えられる) 標準のランレベルで 起動されるように設定された全てのサービスを起動します。標準のランレベルは `/etc/inittab` 中の `id` エントリ により与えられます。Debian は `id=2` となっています。

Debian は次のランレベルを使用しています。

- 1 (シングルユーザモード),
- 2 から 5 (さまざまなマルチユーザモード),
- 0 (システムを停止), そして
- 6 (システムをリブート)。

ランレベル 7, 8, 9 も使用可能ですが、パッケージがインストールされる時に これらの rc ディレクトリにはあまり起動スクリプトがリンクされません。

ランレベルを切替えるには `telinit` コマンドを用います。

あるランレベルに入ると、`/etc/rc`*`runlevel`*`.d/` にある全てのスクリプトが実行されます。 スクリプトの最初の 1文字はスクリプトの起動方法 を決定します。 `K` で始まる名前のスクリプトは、引数 `stop` を取って 起動されます。 `S` で始まる名前のスクリプトは、引数 `start` を取って 起動されます。 これらのスクリプトは名前のアルファベット順で起動されます。それゆえ、“stop” スクリプトは “start” スクリプトより前に起動され、`K` や `S` に続く2桁の数字はスクリプトの起動順序 を決定します。

実は、`/etc/rc`*`runlevel`*`.d/` ディレクトリにある スクリプトは、`/etc/init.d/` にあるスクリプトの 単なるシンボリックリンクです。 これらのスクリプトは引数として “restart” や “force-reload” という 引数も受け取ります。システムのブート後にサービスを再起動 するためや、設定ファイルを再読み込みさせるためにこれらの引数を使えます。

例えば、

```
# /etc/init.d/exim4 reload
```

のように使います。

2.4.3 ランレベルのカスタマイズ

ランレベルをカスタマイズするのは一歩進んだシステム管理者の仕事です。 ほとんどのサービスの場合、次に示すアドバイスが適用できます。

ランレベル *R* でサービス *service* を有効にするには、 ターゲットを `../init.d/`*`service`* としてシンボリックリンク `/etc/rc`*`R`*`.d/S`*`xyservice`* を作成します。 シーケンス番号 *xy* はパッケージがインストールされた時の サービスに割り当てられたシーケンス番号と一致させてください。

サービスを無効にするには、名前が `S` で始まるのではなく `K` で始まるようにシンボリックリンクをリネームし、 シーケンス番号を 100-*xy* にします。

これらの目的には `sysv-rc-conf` や `ksysv` のようなランレベルエディタを使うのが便利です。

特定のランレベルディレクトリにあるサービスのシンボリックリンクを リネームするのではなく削除することも可能です。 これはサービスを無効にしますが、`sysv-rc` の 初期化システムに関する限りに “floating” な状態に保ちます。 ランレベルを変更すると、サービスは起動されませんし、 停止もされませんが、起動しているかそうでないかにかかわらずそのままに 保たれます。 しかしながら、そのような浮いた状態にされたサービスは、 サービスを提

供するパッケージがアップグレードされた場合、 アップグレード前に起動されているかにかかわらず起動されてしまうことに 注意してください。 これは現在の Debian システムの欠点として良く知られています。 又、ランレベル 0 と 6 では `K` で始まるサービスのシンボリックリンク を保持すべきであることに注意してください。 このサービスに対するシンボリックリンクを全て削除してしまうと、 アップグレードの際、出荷時の標準の状態にサービスを提供するパッケージがシンボリックリンクを回復させてしまいます。

`/etc/rcS.d/` にあるシンボリックリンクにあらゆる変更を加える ことは推奨できません。

2.5 多様性のサポート

Debian はシステムを壊さずにシステム管理者の希望を満たすためのいくつかの手段 を提供します。

- `dpkg-divert`, ‘`dpkg-divert` コマンド’ on page 95 参照。
- `equivs`, ‘`equivs` パッケージ’ on page 96 参照。
- `update-alternative`, ‘Alternative コマンド’ on page 96 参照。
- `make-kpkg` は多くのブートローダに対応します。 `make-kpkg(1)` および ‘Kernel (再)構築’ on page 99 参照。

`/usr/local/` 以下のファイルはシステム管理者のものであり、 Debian は一切触りません。 `/etc` 以下のほとんどの ファイルは `conffiles` であり、 Debian はシステム管理者が明示的に要求しない限りアップグレード時に 上書きしません。

2.6 国際化

Debian システムは国際化されており、コンソール上ならびに X 上で 多種の言語の文字を表示し、入力するためのサポートを供給します。 たくさんのドキュメント、マニュアルページ、そしてシステムメッセージが 翻訳されており、翻訳されている言語は増加し続けています。 インストール中、 Debian はユーザにインストール言語 (そして時々はローカル言語変数)の選択を 促します。

あなたが必要な言語の機能全てをインストールしたシステムがサポート していない場合、又はあなたの言語をサポートするために言語の変更や、 異なるキーボードのインストールが必要な場合、‘ローカライゼーション (l10n)’ on page 165 をご覧ください。

2.7 Debian と kernel

‘Debian での Linux kernel’ on page 99 をご覧ください。

2.7.1 非 Debian なソースから kernel をコンパイルする

header に関する Debian ポリシーを理解する必要があります。

Debian C ライブラリは **kernel** header の最新の **stable** リリースを用いて構築されています。

例えば、Debian-1.2 リリースは version 5.4.13 のヘッダを用いていました。この慣習は全ての Linux FTP アーカイブサイトにおいて配布される Linux kernel ソースパッケージに対照しており、Linux kernel ソースパッケージはより最新の header を使ってさえいます。kernel source により配布される kernel header は `/usr/include/linux/include/` にあります。

`libc6-dev` により供給されるものより新しい kernel header を用いてプログラムをコンパイルする必要がある場合には、コンパイル時、コマンドラインに `-I/usr/src/linux/include/` を付け加える必要があります。これは、例えば automounter daemon (`amd`) のパッケージングをする場合に使われます。新しい kernel が NFS を扱うための内部処理を変更した場合、`amd` はそれを知る必要があるため、最新の kernel header を含める必要性が発生します。

2.7.2 カスタム kernel 構築ツール

カスタム kernel を構築したい (する必要がある) 人は `kernel-package` パッケージのダウンロードが推奨されます。本パッケージには kernel パッケージの構築用スクリプトが含まれ、次のようなコマンドを kernel ソースディレクトリの最上段で実行するだけで Debian kernel-image パッケージを構築する機能を供給します。

```
# make-kpkg kernel_image
```

ヘルプは次のコマンドを実行すると得られます。

```
# make-kpkg --help
```

また、マニュアルページ `make-kpkg(1)` 全体と 'Debian での Linux kernel' on page 99 も参照願います。

kernel-source-*version* (*version* は kernel バージョンを表す) パッケージが得られない場合、好みの Linux アーカイブサイトから最新の kernel (又は選んだ kernel) のソースコードを別途ダウンロードする必要があります。Debian の `initrd` ブートスクリプトは `initrd` と呼ばれる特別な kernel patch を要求します。`http://bugs.debian.org/149236` をご覧ください。

`kernel-package` パッケージの詳細な使用方法は `/usr/share/doc/kernel-package/README.gz` にあります。

2.7.3 モジュールを扱うための特別な準備

Debian の `modconf` パッケージはモジュールの設定を カスタマイズするために使用できるシェルスクリプト (`/usr/sbin/modconf`) を供給します。このスクリプトはメニューベースのインターフェースを表示し、システムのローダブルデバイスドライバに関する詳細に 対してユーザを促します。 その応答は `/etc/modules.conf` や `/etc/modules` をカスタマイズするのに使用されます (これらにはブート時にロードされる モジュールがリストされています)。

カスタム kernel の構築をサポートするために得られる (新しい) `Configure.help` ファイルと同様に、`modconf` パッケージにはモジュールそれぞれに対して適切な引数についての詳細な情報を 供給する (`/usr/share/modconf/` にある) ヘルプファイルが付属します。

2.7.4 古い kernel パッケージを削除する

`kernel-image-`*`NNN`*`.prerm` スクリプトは現在起動している kernel が削除しようとしている kernel と同じかどうかチェックします。それゆえ、 使わない kernel image パッケージを安全に次のコマンドを用いて削除できます。

```
# dpkg --purge --force-remove-essential kernel-image-NNN
```

(もちろん、*NNN* を削除したい kernel のバージョンとリビジョンに 置き換えます。)

章 3

Debian システムインストールのヒント

Debian のインストール用の公式文書は `http://www.debian.org/releases/stable/` と `http://www.debian.org/releases/stable/installmanual` にあります。

開発版は `http://www.debian.org/releases/testing/` と `http://www.debian.org/releases/testing/installmanual` にあります。(作業中であり、時々なくなります。)

本章は当初 Potato のインストーラの時代に書かれましたが、内容のほとんどは Woody のインストーラ用に更新されており、さらに 2 つのインストーラの内容は非常に 似通っています。Sarge は完全に新たなインストーラを用いるため、Sarge の インストーラのためには、本文書は基準点として使用してください。 また、いくつかのパッケージは名前と優先度が変更されています。 例えば、Sarge の標準の MTA は `exim` ではなく `exim4` であり、`coreutils` が 複数のパッケージを置き換えるために導入されています。 行動を調整する必要があるかもしれません。

3.1 一般的な Linux システムのインストールヒント

Debian インストーラのコンパクトな CD イメージを探している場合は `http://www.debian.org/CD/netinst/` を忘れずにチェックしてください。

Debian の `testing` 又は `unstable` ディストリビューション を使うことは、深刻なバグに直面するリスクを増大させます。このリスクは Debian のより安定なディストリビューションとのマルチブートを行ったり、'`chroot`' on page 126 に記述されているより安定なディストリビューション内で `chroot` することにより供給される素晴らしい手段を用いることにより 管理できます。後者の方法は Debian の異なるディストリビューションを複数の 端末で同時に起動することを可能とします。

3.1.1 ハードウェア互換性の基礎

Linux はほとんどの PC ハードウェアとの互換性を有しており、ほとんどのシステム にインストールできます。私にとって Linux のインストールは Windows 95/98/Me をインストー

ルするのと同じ位簡単です。ハードウェア互換性リストは常に増大し続けているようです。

ラップトップ PC を持っている場合は、そのブランドとモデル名によるインストールヒントを Linux on Laptops (`http://www.linux-laptop.net/`) で調査できます。

デスクトップ PC ハードウェアのお薦めは “とにかく保守的” ということです。

- 仕事では IDE よりも SCSI、私用では IDE/ATAPI HD。
- IDE/ATAPI CD-ROM (又は CD-RW)。
- ISA よりも PCI。ネットワークカード (NIC) は特に。
- 安い NIC を使用する。PCI では Tulip、ISA では NE2000 が良いでしょう。
- 初めての Linux インストールは PCMCIA (ノート PC) を避ける。
- 人柱になりたくなければ USB キーボードやマウスを避ける。

低速なマシンを持っている場合、ハードディスクを取り外して他の高速なマシンに取り付けてインストールするのは良いアイデアです。

3.1.2 PC のハードウェアとチップセットを決定する

インストール中、PC のハードウェアやチップセットを聞かれることがあります。時々それらの情報は見付かりづらいかもしれません。次にハードウェアやチップセットの情報を見付ける方法を挙げます。

1. PC のケースを開けて中を見る。
2. ビデオカードやネットワークカードの大きなチップ、シリアルポートの近くにあるチップ、IDE ポートの近くにあるチップに付いているプロダクト ID を記録する。
3. PCI 又は ISA カードの後ろにプリントされているカードの名前を記録する。

3.1.3 Debian から PC のハードウェアを決定する

Linux system 上で次のコマンドを実行すると実際のハードウェアとその設定のヒントが得られます。

```
$ pager /proc/pci
$ pager /proc/interrupts
$ pager /proc/ioports
$ pager /proc/bus/usb/devices
```

これらのコマンドはインストールの進行中に端末から Alt-F2 を押すことにより実行可能になります。

最初のイントール後、`pciutils`, `usbutils`, や `lshw` などの追加パッケージをインストールすると、より広範囲なシステム情報が得られます。

```
$ lspci -v |pager
$ lsusb -v |pager
# lshw  |pager
```

典型的な PC は次のように IRQ を使用します。

- IRQ0: タイマ出力 (8254)
- IRQ1: キーボードコントローラ
- IRQ2: PC-AT では IRQ8–IRQ15 とカスケード接続
- IRQ3: セカンダリシリアルポート (io-port=0x2F8) (`/dev/ttyS1`)
- IRQ4: プライマリシリアルポート (io-port=0x3F8) (`/dev/ttyS0`)
- IRQ5: free [sound card (SB16: io-port=0x220, DMA-low=1, DMA-high=5)]
- IRQ6: フロッピディスクコントローラ (io-port=0x3F0) (`/dev/fd0`, `/dev/fd1`)
- IRQ7: パラレルポート (io-port=0x378) (`/dev/lp0`)
- IRQ8: リアルタイムクロック
- IRQ9: ソフトウェア割り込み (int 0x0A)、IRQ2 にリダイレクト
- IRQ10: free [ネットワークカード (NE2000: io-port=0x300)]
- IRQ11: free [(SB16-SCSI: io-port=0x340, SB16-IDE: io-port=0x1E8,0x3EE)]
- IRQ12: PS/2 マウス
- IRQ13: free (かつては 80287 コプロ用)
- IRQ14: プライマリ IDE コントローラ (`/dev/hda`, `/dev/hdb`)
- IRQ15: セカンダリ IDE コントローラ (`/dev/hdc`, `/dev/hdd`)

古い非 PnP ISA カードでは、BIOS から非 PnPな IRQ として IRQ5, IRQ10 および IRQ11 を設定したいかもしれません。

USB デバイスに対しては、デバイスクラスが次のようにリストされます。`/proc/bus/usb/devices` に `Cls=`*`nn`* として 存在する

- Cls=00 : 未使用
- Cls=01 : Audio (スピーカなど)
- Cls=02 : 通信デバイス (モデム、NIC など)
- Cls=03 : HID (Human Interface Device: KB, mouse, joystick)
- Cls=07 : プリンタ
- Cls=08 : Mass storage (FDD, CD/DVD drive, HDD, Flash, . . .)
- Cls=09 : ハブ (USB ハブ)
- Cls=255 : ベンダ独自

デバイスクラスが 255 ではない場合は、Linux はそのデバイスをサポートしています。

3.1.4 他の OS から PC のハードウェアを決定する

ハードウェア情報は他の OS からも取得可能です。

他の商用 Linux ディストリビューションをインストールしてください。これらの ディストリビューションのハードウェア検知機能は現在の Debian の検知機能に比べ 優れています。(この状況は Sarge に `debian-installer` が導入 されても変わらないでしょう。)

Windows をイントールします。ハードウェアの設定は “My Computer” を 右クリックしてデバイスマネージャ/プロパティを開くことにより取得できます。 IRQ、 I/O ポートアドレス、そして DMA のような全てのリソースが 記録されています。ある古い ISA カードは DOS 上で設定する必要があります。

3.1.5 Lilo の俗説

“Lilo is limited to 1024 cylinders.” は間違っている!

Debian potato 以降で使用されている `lilo` は lba32 をサポート しています。マザーボードの BIOS が lba32 をサポートしているくらい新しいなら、`lilo` は古い 1024 シリンダの制限を越えてロードできます。

古い `lilo.conf` を保持している場合、`lilo.conf` の 先頭の近くのどこかに “lba32” を読み込む 1 行を確実に追加してください。 `/usr/share/doc/lilo/Manual.txt.gz` を見てください。

3.1.6 GRUB

GNU Hard プロジェクト由来の新しいブートローダである `grub` は Debian の Woody system に次のようにインストールできます。

```
# apt-get update
# apt-get install grub-doc
# mc /usr/share/doc/grub-doc/html/
... 内容を読む。
# apt-get install grub
# pager /usr/share/doc/grub/README.Debian.gz
... これを読む。
```

GRUB メニュを編集するには、`/boot/grub/menu.lst` を編集してください。 `lilo` とはブートパラメータの設定方法がかなり違うので、 ブート中にブートパラメータを設定する方法について ‘GRUB ブートパラメータの設定’ on page 106 を 見てください。

3.1.7 ブートフロッピーの選択

Potato では、通常のデスクトップへのインストールには IDEPCI disk set がいいでしょう。 Woody では、 bf2.4 ブートディスクセットが良いです。これらはブートフロッピーを 作成するため、共に `boot-floppies` 版を使用しています。

PCMCIA ネットワークカードを持っている場合、標準のブートディスクセットを使用して (大量のフロッピーを必要としますが全てのドライバモジュールが含まれています) PCMCIA setup で NIC の設定を行う必要があります。標準のネットワークセットアップ ダイアログで NIC カードの設定を行おうとしないでください。

特殊なシステムでは、カスタムレスキューディスクを作成する必要があります。 これは違うマシンでコンパイルした圧縮カーネル イメージにより Debian レスキューディスク上の “linux” と言う名前のカーネル イメージを上書きして 置き換えることにより行えます。詳細はレスキューディスク内の `readme.txt` にあります。 レスキューフロッピーは MS-DOS ファイルシステムを使用しているので、 読み込みと編集にあらゆるシステムを使用できま

す。この特徴により、特別な ネットワークカードなどを使用したい人にとって使い易くなっています。

Sarge では、ブートフロッピーの作成に `debian-installer` 又は `pgi` が使われると期待されています。

3.1.8 インストール

`http://www.debian.org/releases/stable/installmanual` 又は `http://www.debian.org/releases/testing/installmanual` (作業中で時々なくなります) にある公式な指示に従ってください。

`testing` ディストリビューションで `boot-floppies` を用いた システムをインストールしている場合、APT ソースを調整するためにインストール中に Alt-F2 を押して端末を開き、`/etc/apt/sources.list` エントリを 編集し、“stable” を “testing” に変更する必要があるかもしれません。

私は `lilo` を `/dev/hda3` のような場所にインストールし、 `mbr` の場合は `/dev/hda` にインストールします。 このようにするとブート情報を上書きする危険性を最小限に抑えることができます。

ここにインストール中私が選ぶ選択肢を挙げます。
- MD5 passwords “yes”
- shadow passwords “yes”
- Install “advanced” (dselect **) and select
  - emacs (選択されていれば), nvi, tex, telnet, talk(d) を除外する。
  - mc, vim および nano-tiny 又は elvis-tiny のうちどちらかを入れる。 ‘`dselect`’ on page 82 を見てください。あなたが Emacs 愛好者の場合でも、 インストール中は emacs をインストールするのを避けて nano に満足 してください。現段階では TeX (Potato が昔インストールしようとしていました) などの巨大なパッケージのインストールを避けてください。nano-tiny 又は elvis-tiny をここでインストールする理由については ‘緊急用エディタ’ on page 208 をごらんください。
- パッケージのインストール ダイアログの間、全ての設定に関する質問に対して “y” (現在の設定と置き換える) を選ぶ。
- `exim`: メールを ISP の SMTP サーバ経由で送るため “2” を 選択する。

dselect に関するより詳しい情報は、‘`dselect`’ on page 82 をごらんください。

3.1.9 LAN を使用するためのホストと IP

LAN 設定例 (C サブネット: 192.168.1.0/24)

```
Internet
   |
   +--- 外部 ISP が POP service を供給 (fetchmail 経由の取得)
   |
```

```
      ISP アクセスポイントが DHCP サービス と SMTP relay サービス を供給
       |                       :
      ケーブルモデム (ダイアルアップ)
       |                       :
   LAN ゲートウェイマシン 外部ポート: eth0 (ISP の DHCP により付与)
    古いノートPCを使用 (IBM Thinkpad, 486 DX2 50MHz, 20MB RAM)
    ext3 ファイルシステムを用いた Linux 2.4 カーネルを使用
    "ipmasq" パッケージを使用 (stronger patch, NAT, さらに firewallを使
用)
    "dhcp-client" パッケージを使用し、eth0 に対して設定 (DNS 設定を上書きす
る)
    "dhcp" パッケージを使用し、eth1 に対して設定
    "exim" を smarthost として使用 (mode 2)
    長時間の間隔で "fetchmail" を使用 (fallback)
    LAN からインターネットへの接続のためのキャッシュサーバおよび LAN から
    LAN ドメインへのアクセスのための信頼できるネームサーバとして
    "bind" を使用。
    port 22 および 8080 にて "ssh" を使用 (どこからでも接続するため)
    (APT からの) Debian archive のため、キャッシュサーバとして "squid"
    を使用。
    LAN ゲートウェイマシン内部ポート: eth1 (IP = 192.168.1.1 固
定)                      |
          +--- LAN スイッチ (100base T) ---+
          |                                |
   LAN 上のいくつかの固定IP クライアント LAN 上のいくつかの DHCP クライアント
   (IP = 192.168.1.2-127 固定)     (IP = 192.168.1.128-200 動的取得)
```

ネットワークの設定の詳細は'ネットワークの設定' on page 177 をごらんください。 LAN ゲートウェイサーバの詳細な設定については'ゲートウェイルータを構築する' on page 203 を ごらんください。

3.1.10 ユーザアカウント

複数のマシンを通じて同じ感覚を保つために、最初のいくつかのアカウントはいつも システムを通じて同一にしています。

いつも最初のユーザアカウントを "admin" (uid=1000) のような名前で作成 しています。全ての root へのメールをそこに転送します。このアカウントは `adm` group ("'なぜ GNU `su` は `wheel` グループを サポートしていないのか"' on page 134 参照) の メンバに与えられ、 root 権限を PAM を通じた `su` 又は `sudo` コマンド を通じて与えることが可能です。詳細は'ユーザアカウントを追加' on page 49 をごらんください。

3.1.11 ファイルシステムを作成

ハードディスクのパーティション

システムクラッシュの際の被害を最小限に抑えるため、異なったディレクトリツリー に対して異なったパーティションを使用する方が好みです。例えば次のように パーティションを分割します。

```
/          == (/ + /boot + /bin + /sbin)
           == 50MB+
/tmp       == 100MB+
/var       == 100MB+
/home      == 100MB+
/usr       == 700MB+ with X
/usr/local == 100MB
```

`/usr/` ディレクトリのサイズは X Window アプリケーションや ドキュメントの量に極めて依存しています。コンソールのみを使用する場合、 `/usr/` は 300MB にすることができます。一方、多数の Gnome アプリケーションをインストールした場合、2GB–3GB のサイズを割り当てるのは 異常ではありません。 `/usr/` が非常に大きくなってしまった時、`/usr/share/` を他のパーティションに移すのは最も効率的な対策です。新しい大規模な パッケージ化された Linux 2.4 カーネルでは、`/` は 200MB 以上必要となるかもしれません。

例えば、私の Internet ゲートウェイサーバの現在の状況は次のようになっています。 (`df -h`コマンドの出力)

```
Filesystem             Size  Used Avail Use% Mounted on
/dev/hda3              300M  106M  179M  38% /
/dev/hda7              100M   12M   82M  13% /home
/dev/hda8              596M   53M  513M  10% /var
/dev/hda6              100M  834k   94M   1% /var/lib/cvs
/dev/hda9              596M  222M  343M  40% /usr
/dev/hda10             596M  130M  436M  23% /var/cache/apt/archives
/dev/hda11             1.5G  204M  1.2G  14% /var/spool/squid
```

(`/var/spool/squid/` に割り当てられている大容量の領域は パッケージのダウンロードの際の proxy キャッシュのためです。)

次は パーティション構造の考えを示すための `fdisk -l` の出力です。

```
# fdisk -l /dev/hda # comment

/dev/hda1             1         41    309928+   6  FAT16 # DOS
/dev/hda2            42         84    325080   83  Linux # (not used)
/dev/hda3   *        85        126    317520   83  Linux # Main
```

```
/dev/hda4           127       629   3802680    5  Extended
/dev/hda5           127       143    128488+  82  Linux swap
/dev/hda6           144       157    105808+  83  Linux
/dev/hda7           158       171    105808+  83  Linux
/dev/hda8           172       253    619888+  83  Linux
/dev/hda9           254       335    619888+  83  Linux
/dev/hda10          336       417    619888+  83  Linux
/dev/hda11          418       629   1602688+  83  Linux
```

いくつかの未使用のパーティションが存在しています。これらは2個目のLinuxディストリビューションのインストールや大きくなったディレクトリツリーのための拡張スペースのためのものです。

ファイルシステムのマウント

上記のファイルシステムを適切にマウントするために、次に示すように `/etc/fstab` を設定しています。

```
# /etc/fstab: static filesystem information.
#
# filesystem    mount point     type    options                  dump pass
/dev/hda3       /               ext2    defaults,errors=remount-ro 0 1
/dev/hda5       none            swap    sw                         0 0
proc            /proc           proc    defaults                   0 0
/dev/fd0        /floppy         auto    defaults,user,noauto       0 0
/dev/cdrom      /cdrom          iso9660 defaults,ro,user,noauto    0 0
#
# パーティションを分割しつづけます
/dev/hda7       /home           ext2    defaults                   0 2
/dev/hda8       /var            ext2    defaults                   0 2
/dev/hda6       /var/lib/cvs    ext2    defaults                   0 2
# noatime オプション はファイルの読み込みのアクセスを高速化する
/dev/hda9       /usr            ext2    defaults,noatime           0 2
/dev/hda10      /var/cache/apt/archives ext2    defaults           0 2

# proxy キャッシュのための非常に大きなパーティション
/dev/hda11      /var/spool/squid ext2   rw                         0 2

# ブート可能な DOS のバックアップ
/dev/hda1       /mnt/dos        vfat    rw,noauto                  0 0
# ブート可能な Linux システムのバックアップ (未実施)
/dev/hda2       /mnt/linux      ext2    rw,noauto                  0 0
#
```

```
# nfs マウント
mickey:/        /mnt/mickey     nfs     ro,noauto,intr          0 0
goofy:/         /mnt/goofy      nfs     ro,noauto,intr          0 0
# minnie:/ /mnt/minnie smbfs ro,soft,intr,credentials={filename} 0 2
```

NFS の場合、`noauto,intr` オプションをデフォルトの`hard` オプションと組み合わせて使用します。このようにすると、Ctrl-C を用いた dead-connection によるプロセスのハングから回復することが可能です。

Samba (smbfs) により接続した Windows マシンに対しては、`rw,auto,soft,intr`オプションを使用するのは良い考えです。‘Samba の設定’ on page 41 をごらんください。

フロッピードライブに対しては、`noauto,rw,sync,user,exec` を通常のオプションの代わりに用いて unmount する前に誤ってディスクをイジェクトしてしまった後のファイル破壊を防ぐことができます。

Autofs マウント

auto mount する際の注意点は次の通り。

- `vfat` モジュールをロードすると、`/etc/auto.misc` に `-fstype=auto` を含ませることを可能とします。
  ```
  # modprobe vfat # フロッピーへのアクセスを行う前に実行
   ... 又は本設定を自動化させるには
  # echo "vfat" >> /etc/modules
   ... そしてシステムをリブートする。
  ```
- `/etc/auto.misc`を次のように設定します。
  ```
  floppy -fstype=auto,sync,nodev,nosuid,gid=100,umask=000 :/dev/fd0
   ... ここで gid=100 は "users" です。
  ```
- `cdrom` と `floppy` へのリンクを `/home/`*user* に、それぞれ `/var/autofs/misc/cdrom` と `/var/autofs/misc/floppy` を指すように設定する。
- *user* を “users” グループに追加する。

NFS マウント

ファイアーウォール (ゲートウェイ) の後ろに (まぬけにも) 外部の Linux NFS サーバが存在します。私しかユーザが居ないので、非常に緩やかなセキュリティポリシーを LAN 内では適用しています。NFS アクセスを可能とするには、NFS サーバ側で `/etc/exports` に対して次のように追加する必要があります。

```
# /etc/exports: the access control list for filesystems which may be
#               exported to NFS clients.  See exports(5).
/       (rw,no_root_squash)
```

これは NFS サーバパッケージおよびクライアントパッケージをインストールして有効にすることに加え、NFS サーバを有効にするためにも必要です。

単純化のため、通常実験又は2個目の不精な Linux インストールのための 2GB の 単独のパーティションを作成します。これらのインストールのために swap や `/tmp` の共有を任意で行います。複数パーティションとすると、 これらの手段が使用できなくなるほど複雑になってしまいます。単独の端末システム のみを必要とする場合、500MB もあれば十分です。

3.1.12 DRAM メモリ量のガイドライン

次に DRAM 搭載量の大まかなガイドラインを示します。

```
   4MB:   Linux kernel が動く最小量
  16MB:   適切なコンソールのみのシステムに最低限必要な量
  32MB:   シンプルな X Window system に最低限必要な量
  64MB:   GNOME/KDE を用いた X Window system を動かすのに最低限必要な量
 128MB:   GNOME/KDE を用いた X Window system が快適な量
 256MB(以上): なぜ買えないの? DRAM は安いよ。
```

`mem=4m` ブートオプションを用いる (又は lilo で `append="mem=4m"` とする) とシステムにメモリが 4MB しか無い場合の振舞がどのようなものかを 示します。BIOSが古く、64MB 以上のメモリを搭載しているシステムには 特別な lilo ブートパラメータが必要です。

3.1.13 スワップ

スワップに対しては次のようなガイドラインを使用します。

- 各スワップパーティション < 128MB (古い 2.0 kernel の場合), < 2GB (最近のカーネルの場合)
- 合計 = (搭載した RAM の量の1から2倍) 又は (128MB から 2GB)
- 各スワップパーティションを異なったディスクに分散し、それらすべてを `/etc/fstab` に `sw,pri=1` オプションを追加してマウント する。これはカーネルがスワップパーティションのストライピング RAID を行う ことを保証し 、最大のスワップパフォーマンスをもたらします。
- 可能ならばハードディスクの中央部分を使用できれば、より良い。

スワップを必要としない場合でも、メモリリークを起こすプログラムによりクラッシュ する前にシステムが低速化するので、いくらかのスワップ (128 MB) を用意するのが 望ましいです。

3.2 Bash 設定

システム全体に渡って自分の好みに合うようにシェルの起動スクリプトを変更します。

```
/etc/bash.bashrc        個人独自のコピーで置き換える
/etc/profile            distribution のコピーを保つ ( \w -> \W)
```

/etc/skel/.bashrc	個人独自のコピーで置き換える
/etc/skel/.profile	個人独自のコピーで置き換える
/etc/skel/.bash_profile	個人独自のコピーで置き換える
~/.bashrc	全てのアカウントについて個人独自のコピーで置き換える
~/.profile	全てのアカウントについて個人独自のコピーで置き換える
~/.bash_profile	全てのアカウントについて個人独自のコピーで置き換える

詳細は私のサンプルスクリプトをご覧下さい (`http://www.debian.org/doc/manuals/debian-reference/examples/`)。 私は透過的なシステムが好きなので、`umask` を 002 又は 022 にセットします。

`PATH` は次に挙げる順で設定ファイルによりセットされます。

```
/etc/login.defs  - シェルが PATH をセットする前
/etc/profile     (may /etc/bash.bashrc を呼んでいるかもしれません)
~/.bash_profile  (~/.bashrc を呼んでいるかもしれません)
```

3.3 マウスの設定

3.3.1 PS/2 マウス

ATX マザーボードの PS/2 コネクタにつなげるマウスの場合、信号の流れは 次のようになります。

```
mouse -> /dev/psaux -> gpm -> /dev/gpmdata = /dev/mouse -> X
```

ここで、`/dev/psaux`というシンボリックリンクが作成され、 `/dev/gpmdata` を指すことにより、いくつかの設定ユーティリティ がマウスを使い易くし、再設定を容易にすることができます。 (例えば、結局 gpm デーモンを使わないことにした場合、 `gpm` デーモンを削除した後に `/dev/mouse` という シンボリックリンクは単に `/dev/psaux` を指すことになります。)

この信号の流れにより、再接続時に gpm を再起動することにより キーボードやマウスが外され、再び接続することができるようになります。 X は動いたままです!

gpm の出力と X への入力の間の信号プロトコルは “ms3” (Microsoft 3ボタンシ リアルマウス プロトコル) 又は “raw” (同じ プロトコルを接続された マウスとして使用する) のどちらでも実装できます。そしてこのプロトコルの選択 が X の設定で使用されるプロトコルの選択を規定します。

Logistech 3ボタン (伝統的な Unix スタイルのマウス) PS/2 マウスを 例として使用して次にマウスの設定の例を挙げます。

新しい X Version 4 でサポートされていないビデオカード (いくつかの ATI 64ビットカード) を使っており、古い X Version 3 を使う必要 があるなら、X Version 3 パッケージをインストールして次に示すように `/etc/X11/XF86Config-4` の代わりに `/etc/X11/XF86Config` を設定してください。

The ms3 プロトコルを用いる方法

```
/etc/gpm.conf               | /etc/X11/XF86Config-4
==========================+=======================================
device=/dev/psaux           | Section "InputDevice"
responsiveness=             |  Identifier "Configured Mouse"
repeat_type=ms3             |  Driver     "mouse"
type=autops2                |  Option     "CorePointer"
append=""                   |  Option     "Device"   "/dev/mouse"
sample_rate=                |  Option     "Protocol" "IntelliMouse"
                            | EndSection
```

この手法を用いると、マウス形式の調整は `gpm.conf`の 編集のみ行えばよく、X の設定はそのままでかまいません。 my example scripts (`http://www.debian.org/doc/manuals/debian-reference/examples/`) をごらんください。

raw プロトコルを用いる方法

```
/etc/gpm.conf               | /etc/X11/XF86Config-4
==========================+=======================================
device=/dev/psaux           | Section "InputDevice"
responsiveness=             |  Identifier "Configured Mouse"
repeat_type=raw             |  Driver     "mouse"
type=autops2                |  Option     "CorePointer"
append=""                   |  Option     "Device"   "/dev/mouse"
sample_rate=                |  Option     "Protocol" "MouseManPlusPS/2"
                            | EndSection
```

本手法を用いると、マウス形式の調整は `gpm.conf` を編集する のに加えて、Xの設定も調整する必要があります。

異なるマウスを調整する方法

`gpm` における `autops2` デバイスタイプは 市場に出回っているほとんどの PS/2 マウスを自動検知するはずです。不幸にも これはいつも動くとは限らず、Woody 以前には使えませんでした。そのような 場合には `autops2` の代わりとして `/etc/gpm.conf` に `ps2` 又は `imps2` を使用してください。 `gpm` が認識できる特別なタイプのマウスを調べるには、 `gpm -t help` を実行してください。`gpm(8)` をごらんください。

2 ボタン PS/2 マウスを使用している場合、X プロトコルが `Emulate3Buttons` を有効化するように設定してください。中央のボタンを 1 度押すと 2ボタンマウスと 3 ボタンマウスの間のプロトコルの違いが自動検知され、gpm 用に 自動調整されます。

X プロトコルで 'raw プロトコルを用いる方法' on page 38 を用いる、又は gpm を 用いない場合、次のマウス形式を使用してください。

- `IntelliMouse`: シリアルポートマウス (gpm は "ms3" を繰り返します)
- `PS/2`: PS/2 ポートマウス (常に最初に試されます)
- `IMPS/2`: PS/2 ポートマウス (2 ボタン、3 ボタン、そしてスクロールマウスではこれがよりふさわしい)
- `MouseManPlusPS/2`: Logitech PS/2 ポートマウス
- `...`

詳細は Mouse Support in XFree86 (`http://www.xfree86.org/current/mouse.html`) を ごらんください。

典型的な Microsoft スクロールマウスの場合、次の設定によりもっとうまく動くと 報告されています。

```
/etc/gpm.conf            | /etc/X11/XF86Config-4
=========================+======================================
device=/dev/psaux        | Section "InputDevice"
responsiveness=          |  Identifier "Configured Mouse"
repeat_type=raw          |  Driver     "mouse"
type=autops2             |  Option     "CorePointer"
append=""                |  Option     "Device"    "/dev/mouse"
sample_rate=             |  Option     "Protocol" "IMPS/2"
                         |  Option     "Buttons" "5"
                         |  Option  "ZAxisMapping" "4 5"
                         | EndSection
```

最近の薄型の東芝ノートPC では、gpmを PCMCIA よりも先に 起動するとシステムのデッドロックを防ぐことができるようです。不気味ですが 本当です。

3.3.2 USB マウス

必要な全ての kernel 機能をカーネルコンパイル時の設定やモジュールを 通して有効にしたかどうかを確認してください。

- "Input core support" の下 の
  - "Input core support" (CONFIG_INPUT, `input.o`),
  - "Mouse support" (CONFIG_INPUT_MOUSEDEV, `mousedev.o`),
- "USB support" の下の
  - "Support for USB" (CONFIG_USB, `usbcore.o`),
  - "Preliminary USB device filesystem" (CONFIG_USB_DEVICEFS),
  - "UHCI" 又は "OHCI" (CONFIG_USB_UHCI || CONFIG_USB_UHCI_ALT || CONFIG_USB_OHCI, `usb-uhci.o` || `uhci.o` || `usb-ohci.o`),

  - “USB Human Interface Device (full HID) support” (CONFIG_USB_HID, `hid.o`) および
  - “HID input layer support” (CONFIG_USB_HIDINPUT)

ここで、小文字の名前はモジュール名です。

devfs を使用していない場合、`/dev/input/mice` デバイスノードを major デバイス番号 13 、 minor デバイス番号 63 を用いて次のように 作成してください。

```
# cd /dev
# mkdir input
# mknod input/mice c 13 63
```

典型的なスクロール **USB** マウスの場合、 次のような 2 種類の設定を組み合わせる必要があります。

```
/etc/gpm.conf            | /etc/X11/XF86Config-4
=========================+======================================
device=/dev/input/mice   | Section "InputDevice"
responsiveness=          |  Identifier "Generic Mouse"
repeat_type=raw          |  Driver     "mouse"
type=autops2             |  Option     "SendCoreEvents" "true"
append=""                |  Option     "Device"    "/dev/input/mice"
sample_rate=             |  Option     "Protocol" "IMPS/2"
                         |  Option     "Buttons" "5"
                         |  Option  "ZAxisMapping" "4 5"
                         | EndSection
```

詳細は the Linux USB Project (`http://www.linux-usb.org/`) をごらんください。

3.3.3 タッチパッド

ノート PC にあるタッチパッドは 2 ボタン PS/2 マウスをデフォルトの動作 としてエミュレートしますが、`tpconfig` パッケージ によりタッチパッドのフル機能を有効化できます。例えば、 `/etc/default/tpconfig` にて `OPTIONS="--tapmode=0"` と設定すると、くどい “click by tap” 機能を無効にできます。 コンソール上でタッチパッドと USB マウスを同時に使用するには `/etc/gpm.conf` を次のように設定してください。

```
device=/dev/psaux
responsiveness=
repeat_type=ms3
type=autops2
append="-M -m /dev/input/mice -t autops2"
sample_rate=
```

3.4 NFS 設定

`/etc/exports` を調整することにより NFS の設定を行えます。

```
# apt-get install nfs-kernel-server
# echo "/ *.LAN上のホストノドメイン名(rw,no_root_squash,nohide)" \
        >> /etc/exports
```

詳細は私のサンプルスクリプトをごらんください (`http://www.debian.org/doc/manuals/debian-reference/examples/`)。

3.5 Samba の設定

リファレンスをいくつか示します。

- `http://www.samba.org/`
- `samba-doc` パッケージ

WfW-type の共有ドライブを作成するので、Samba を “share” モードに 設定するのはより容易です。しかし、私は “user” モードで設定する方が好きです。

Samba は `debconf` 又は `vi` を通じて設定可能です。

```
# dpkg-reconfigure --priority=low samba # Woody にて
# vi /etc/samba/smb.conf
```

詳細は私のサンプルスクリプトをごらんください (`http://www.debian.org/doc/manuals/debian-reference/examples/`)。

`smbpasswd` への新規ユーザの追加は、`smbpasswd`を 次のように用いることにより可能です。

```
$ su -c "smbpasswd -a username"
```

最適な互換性を保つため、暗号化パスワードを確実に使用してください。

次に示すシステム等価係数 (大きくなるほど、サーバとしての優先度が高い) に従い、`os level` を設定してください。

```
0:      緩い考えの Samba (マスタブラウザには決してならない)
1:      WfW 3.1, Win95, Win98, Win/Me?
16:     Win NT WS 3.51
17:     Win NT WS 4.0
32:     Win NT SVR 3.51
33:     Win NT SVR 4.0
255:    パワフルな Samba
```

ユーザは共有アクセスを与えるディレクトリを所有するグループに属しており、 ディレクトリパスにはアクセスするための実行権があることを確認してください。

3.6 プリンタの設定

プリンタを設定するための伝統的な方法は `lpr` 又は `lpd` です。今は新しい CUPS™ システム (Common UNIX Printing System) が存在します。PDQ はもう一つのアプローチです。詳細は Linux Printing Howto (`http://www.tldp.org/HOWTO/Printing-HOWTO.html`) をごらんください。

3.6.1 lpr又はlpd

ポストスクリプトプリンタ又はテキストのみのプリンタを接続しているならば、`lpr` 又は `lpd` タイプのスプールのために `/etc/printcap` を次のように設定してください。

```
lp|alias:\
        :sd=/var/spool/lpd/lp:\
        :mx#0:\
        :sh:\
        :lp=/dev/lp0:
```

上記の行はそれぞれ次のような意味があります:

- 先頭の行: *lp* – スプール名, *alias* = エイリアス
- mx#0 – ファイルサイズの上限は制限されていない
- sh – バーストページのヘッダの印刷を止める。
- lp=/dev/lp0 – ローカルのプリンタデバイス又はリモートプリンタの ポート@ホスト名を示す

本設定はポストスクリプトプリンタを接続している場合、良い設定です。又、 Samba を通じて Windows マシンに接続したプリンタから印刷する場合の Windows をサポートした (双方向通信がサポートされない) プリンタ にとっても良い設定です。Windows マシン上で対応するプリンタの設定を選択 する必要があります。

ポストスクリプトプリンタを所有していない場合、`gs` により フィルタシステムの設定を行う必要があります。 `/etc/printcap` の設定のために供給される数多くの自動設定 ツールが存在します。 これらを自由に組み合わせることが可能です。

- `gnulpr`, (`lpr-ppd`) と `printtool`—私はこの組合せを使っています。
- `lpr` と `apsfilter`
- `lpr` と `magicfilter`
- `lprng` と `lprngtool`
- `lprng` と `apsfilter`
- `lprng` と `magicfilter`

`printtool` のような GUI 設定ツールを起動するために root 権限 を取得するには 'X で root 権限を取得する' on page 153 を見てください。`printtool` により作成されたプリンタスプールは `gs` を使用し、ポストスクリプト プリンタのように振る舞います。それゆえ、これらのプリンタにアクセスした時、 ポストスクリプトプリンタのドライバを使用します。Windows 側では、この プリンタを用いるには "Apple LaserWriter" が標準的なドライバです。

3.6.2 CUPS™

The Common UNIX Printing System (又は CUPS™) は `aptitude` を使い “Tasks” -> “Servers” -> “Print Server” の下の全てのパッケージをインストールすることで導入できます。 (Sarge) 最善の結果のためには `aptitude` を “F10” -> “Options” -> “Dependency handling” -> “[X] Install Recommended packages automatically”. と設定するべきです。

KDE や Gnome デスクトップ環境では簡単にプリンターの設定ができます。 この他に `swat` が導入されている場合は web ブラウザを用いてシステムの設定を行えます。

```
$ 好みのブラウザコマンド http://localhost:631
```

例えば、アクセス可能なプリンタのリストにプリンタを追加するには、次を実行します。

- メインページの “Printers” をクリックし、“Add Printer” を押し、
- ユーザ名として “root” を入力し、パスワードとして root のパスワードを
- 入力する。
- プロンプトに従い、プリンタの追加を進める。
- “Printers” ページに戻り、“Configure Printer” を押し、
- 用紙サイズ、解像度などのパラメータの設定を進める。

詳細は `http://localhost:631/documentation.html` 及び `http://www.cups.org/cups-help.html` をご覧ください。

3.7 デスクトップPC用のCRON

Vixie `cron` はプログラムのスケジュール実行処理の デフォルトとしてインストールされます。 システムが毎日２４時間連続で走っていないとこれは うまく機能しません。デスクトップ用途に用いるPCの場合、この問題に 対処するように `cron` に加え `anacron` もインストールする必要があります。 `fcron` パッケージはこの代用となります。

CRON ジョブのコンフィギュレーションに関しては‘タスクのスケジューリング (`cron`、`at`)’ on page 123 を参照してください。

3.8 他のホストへの インストールのヒント

3.8.1 最初の インストール後にさらにいくつかパッケージを インストールする

いったんここまで来たら、小さいけれどもちゃんと機能する Debian システムを 持っていることになります。もっと大きいパッケージをインストールしても良い 時間です。

まずあなたが最初に行っても良いのは、好みのエディタや他のプログラムを `aptidude` を使って選ぶことです。多数の Emacs の変種をインストールして共存できます。'有名なエディタ' on page 207 をご覧ください。

次に、完璧にエンドユーザ指向なシステムを得るために、再び `aptitude` を起動して "タスク" –> "End-user" の順に進めば、(ほとんど) 全てのパッケージを選択できます。

より簡単にシャットダウンさせるため、私は通常 `/etc/inittab` を次のように編集します。

```
...
# CTRL-ALT-DEL が押された時に実行
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -h now
...
```

3.8.2 モジュール

デバイスドライバ用のモジュールは最初のインストール中に設定されます。`modconf` はメニュ指向なモジュール設定を後に供給します。このプログラムは最小のインストール中にいくつかのモジュールを除外されたり、最初のインストール後に新しいカーネルをインストールした場合に非常に有用です。

全てのプリロードされたモジュール名は `/etc/modules` にあるリストに追加される必要があります。モジュールの制御を手動で行うために `lsmod` や `depmod` も使用します。

またもしまだの場合には `/etc/modules`に数行書き加えて IP masquerading (FTP等)が可能なように設定することを確実にしましょう。

3.8.3 CD-RW の基本的な設定

IDE で接続された CD-RW ドライブが 2.4 カーネルとともに使われる場合には、次に示すファイルを編集してください。

```
/etc/lilo.conf  (append="hdc=ide-scsi ignore=hdc" を追加し、
                 設定を有効にするために lilo を実行する)
/dev/cdrom      (シンボリックリンク # cd /dev; ln -sf scd0 cdrom)
/etc/modules    ("ide-scsi" および "sg" を追加する。"sr" が必要なら
                その後に追加する。)
```

詳細は 'CD ライター' on page 137 をごらんください。

3.8.4 大容量メモリおよび自動電源 OFF

大規模メモリ (2.2 kernel 用)と自動電源 OFF (APM 用) のためのブートプロンプトパラメータを設定するため、`/etc/lilo.conf` を次のように編集してください。

```
append="mem=128M apm=on apm=power-off noapic"
```

これらの設定をインストールするには `lilo` を実行してください。 SMP カーネルには `apm=power-off` の設定が必要であり、buggy な SMP ハードウェアの問題を回避するには `noapic` と設定する必要があります。同じことはブートプロンプトでオプションを入力することによっても可能です。'ブートプロンプトを使った他のブートトリック' on page 106 をごらんください。

APM がモジュールとしてコンパイルされている場合、Debian のデフォルトの version 2.4 カーネルでは、`insmod apm power-off=1` とブート後に実行するか、`/etc/modules` に次のように設定してください。

```
# echo "apm power_off=1" >>/etc/modules
```

又は、ACPI のサポートを組み込んでコンパイルすることにより、より新しい kernel を用いて同じ目標を達成出来ますし、より SMP フレンドリーにもなるようです (これには新しいマザーボードが必要です)。 新しいマザーボード上で 2.4 kernel を次の設定で動かすと大容量メモリを正確に認識できます。

```
CONFIG_PM=y
CONFIG_ACPI=y
...
CONFIG_ACPI_BUSMGR=m
CONFIG_ACPI_SYS=m
```

そして、次の 2 行をこの順番で `/etc/modules` に追加してください。

```
ospm_busmgr
ospm_system
```

又は、上記に示したオプションを "y" にセットして kernel を再構築してください。 どの場合でも、ACPI を用いた場合にはブートプロンプトパラメタは必要ありません。

3.8.5 いくつかのウェブサイトにアクセスした時の奇妙な挙動

最近の Linux kernel は通常 ECN を有効化でき、これが腐ったルータ上にあるいくつかのウェブサイトにアクセスした際に問題を引き起こします。ECN の状況を確認するには。次を実行します。

```
# cat /proc/sys/net/ipv4/tcp_ecn
 ... 又は
# sysctl net.ipv4.tcp_ecn
を実行してください。
```

ECN を無効化するには、次のコマンドを実行してください。

```
# echo "0" > /proc/sys/net/ipv4/tcp_ecn
 ... 又は
# sysctl -w net.ipv4.tcp_ecn=0
```

ブート時に TCP ECN を無効化するには、`/etc/sysctl.conf` を 次のように編集してください。

```
net.ipv4.tcp_ecn = 0
```

3.8.6 ダイアルアップ PPP 接続の設定

`pppconfig` パッケージをインストールすることにより ダイアルアップ PPP 接続の設定ができます。

```
# apt-get install pppconfig
# pppconfig
 ... ダイアルアップ PPP のための指示に従う
# adduser user_name dip
 ... user_name のダイアルアップ PPP へのアクセスを許可
```

ユーザ (*user_name*) によりダイアルアップ PPP を次のように 起動できます。

```
$ pon ISP_name  # プロバイダへの PPP 接続を開始。
 ... Internet を愉しむ
$ poff ISP_name # PPP 接続を切断。 ISP_name は任意。
```

詳細は 'PPP インターフェイスの設定' on page 182 をごらんください。

3.8.7 `/etc/` 下のファイルを微調整するための他の設定ファイル

標準的な Debian のインストールでは欠けている `/etc/cron.deny` ファイルを追加したいかもしれません。(`/etc/at.deny` をコピー可能です。)

章 4

Debian チュートリアル

本章は本当の初心者に Debian ワールドへの基本的なオリエンテーションを供給します。しばらく Unix ライクなシステムを使っていたならば、ここで説明することは多分ほとんど知っているでしょう。常識の確認としてこれを使ってください。

4.1 始め方

PC に Debian システムをインストールした後、有用ないくつかのことを学ぶ必要があります。素早い説明をします。

4.1.1 root としてシェルにログインする

システムのリブート後、パッケージの最初の選択により、グラフィカルなログインスクリーン又は文字ベースのログインスクリーンが表れます。単純化のため、グラフィカルなログインスクリーンが表れた場合、キャラクタベースのログインスクリーンを表示させるために Ctrl-Alt-F1 を押してください。[1]

あなたのホスト名が *foo* と仮定すると、ログインプロンプトは次のようになります。

```
foo login:
```

`root` とタイプし、Enter キーを押してインストール手順の間に選んだパスワードをタイプしてください。Debian システムでは、Unix の伝統に従い、パスワードは大文字と小文字が区別されます。ログインに成功すると、システムは歓迎メッセージを表示し、root コマンドプロンプトを表示してあなたの入力を待ちます。[2]

```
foo login: root
```

---

[1] 左側の Ctrl キー、左側の Alt キー、そして F1 キーを同時に押すことです。

[2] `/etc/motd` にある歓迎メッセージを編集した場合、メッセージは異なることに注意。

```
Password:
Last login: Sun Oct 26 19:04:09 2003 on tty3
Linux foo 2.4.22-1-686 #6 Sat Oct 4 14:09:08 EST 2003 i686 GNU/Linux

Most of the programs included with the Debian GNU/Linux system are
freely redistributable; the exact distribution terms for each program
are described in the individual files in /usr/share/doc/*/copyright

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

root@foo:root#
```

あなたは既にこの root コマンドプロンプトからシステム管理を行う準備ができています。この root アカウントは、スーパーユーザや特権ユーザとも呼ばれています。このアカウントからは、次のようなことができます。

- ファイルパーミッションに依らずあらゆるファイルの読み書き削除
- システム上のあらゆるファイルのファイル所有者やパーミッションの設定
- システム上の非特権ユーザのパスワードの設定
- パスワード無しでの全アカウントへのログイン

パスワードを共有することにより root アカウントへのアクセスを共有することは非常にまずい考えです。`sudo(8)` のようなプログラムを使って管理者権限を共有するのが良い考えです。

管理行動を行おうとする場合でも、まず自分の非特権者アカウントでログインするのは、良い Unix 的習慣であると考えられていることに注意してください。必要な際に制限された root 権限を得るために、`sudo`, `super`, や `su -c` コマンドを使いましょう。'– `sudo` を使ってより安全に作業をする' on page 135 をご覧ください。[3]

4.1.2 最小限の初心者用環境をセットアップ

コンピュータシステムを学ぶことは新しい外国語を学ぶことに似ていると考えます。チュートリアルブックは有用ですが、ヘルパーツールを用いて実際に学ぶことが必要です。こういう背景から、`mc`, `vim`, `lynx`, `doc-linux-text`, そして `debian-policy` のようないくつかの追加パッケージをインストールするのは良い考えです。[4]

```
# apt-get update
 ...
# apt-get install mc vim lynx doc-linux-text debian-policy
 ...
```

---

[3] 簡単でだらしがないだけの理由で必要以上にスーパーユーザアカウントを使っていたことを認めなくてはなりません。

[4] `gpm`, `emacs21`, や `doc-linux-html` をインストールするのも良い考えかもしれません。'マウスの設定' on page 37 や 'エディタ' on page 207 参照。

これらのパッケージを既にインストールしている場合、何もインストール しなくても良いです。

4.1.3 ユーザアカウントを追加

インストール中に、通常 root アカウントへのメールを受け取る非特権ユーザ アカウントを作成します。[5] この特別なユーザアカウントを以降の訓練用にも使いたくないならば、他の新ユーザアカウントを作成すべきです。

新しいユーザ名を `penguin` とすると、次のように タイプするとアカウントが作成されます。

```
root@foo:root# adduser penguin
... 質問全てに答える
```

[6] 先に進む前に、いくつかのことをまず学びましょう。

4.1.4 仮想コンソールを切替える

Debian システムには、6 個の独立した仮想端末が使えます。すなわち、PC の VGA キャラクタコンソールスクリーンを 6 個の切替え可能な VT100 端末として使えるということです。スクリーンの切替えは左側の Alt キー と F1–F6 キーを同時に押すことにより行えます。仮想ターミナル それぞれに独立したアカウントでログインすることができます。 マルチユーザ環境は偉大な Unix の機能であり、非常に中毒性があります。

X Window System を起動中のシステムにおいて Alt-F7 を間違ってタイプし、 コンソールスクリーンがグラフィカルなスクリーンを表示していた場合、 Ctrl-Alt-F1 を押すことによりキャラクタコンソールへのアクセスを回復します。 この現象に慣れるため、違うコンソールに移動して元に戻ってみてください。

4.1.5 シャットダウンのやりかた

ファイル操作がメモリへのキャッシュにかかわる他の現代的な OS と同様に、 Debian システムは電源を OFF にする前にファイルの完全性を維持するため、 適切なシャットダウン手順を取る必要があります。 システムをシャットダウンするためには、root コマンドプロンプトから 次に示すコマンドを実行してください。

```
# shutdown -h now
```

---

[5] 私はいつもインストール中に作成されるアカウントを `admin` としますが、 任意の名前でかまいません。

[6] この *penguin* ユーザを `adm` グループに追加 して `/var/log/` にある多くのログファイルを読めるようにしたい かもしれません。 `passwd(5)`, `group(5)`, `shadow(5)`, `group(5)`, `vipw(8)`, そして `vigr(8)` をご覧ください。 ユーザやグループの公式な意味は、 Users and Groups (`/usr/share/doc/base-passwd/users-and-groups.html`) 文書の最近のバージョンをご覧ください。

これは通常のマルチユーザモード用です。シングルユーザモードの場合は、 root コマンドプロンプトから次のコマンドを実行してください。

```
# poweroff -i -f
```

これの代わりに、Ctrl-Alt-Delete を押してシャットダウンできるかもしれません。[7]

システムが “System halted” と表示するまで待ち、そして電源を OFF にしましょう。BIOS と Linux により APM 又は ACPI の機能が有効化された場合、システムは何もせずに電源を OFF にします。詳細は ‘大容量メモリおよび自動電源 OFF’ on page 44 を ご覧ください。

4.1.6 プレイ時間

ついに Debian システムを非特権ユーザアカウント *penguin* を使ってできるだけリスクを負わずに楽しむ準備ができました。[8]

penguin にログインしましょう。まだ root シェルプロンプトにいるなら、Ctrl-D をタイプしましょう。[9] こうして root シェルでの行動を止めるために root コマンドプロンプトから 出て、ログインプロンプトに戻ります。新しく作ったユーザ名 *penguin* と対応したパスワードを入力しましょう。[10] するとこのようなコマンドプロンプトが表示されるでしょう。

```
penguin@foo:penguin$
```

ここから、単純化のために、シンプルにしたコマンドプロンプトの例を下に挙げます。

- `#` : root シェルプロンプト
- `$` : 非特権ユーザシェルプロンプト

まず簡単な方法 ‘Midnight Commander (MC)’ on page 50 を使って Debian システムを学び、後で 適切な方法 (‘Unix ライクな作業環境’ on page 53) を用いて学ぶことにします。

4.2 Midnight Commander (MC)

Midnight Commander (MC) は Linux コンソールや他の端末環境のための GNU 製 “スイス軍ナイフ” です。初心者にメニュドリブンなコンソール経験を与え、 標準的な Unix コマンドを習うよりもより簡単です。

---

[7] コンソールから左 Ctrl キー、左 Alt キーと Delete キーを同時に押します。 標準のシステムでは、これはシステムをリブートさせます。‘最初のインストール後にさらにいくつかパッケージを インストールする’ on page 43 に記述しているように、`shutdown` コマンドのオプションが `-h` となるように `/etc/inittab` を修正する必要があります。

[8] これは Debian システムでは、標準のインストール直後でさえも、 非特権ユーザがシステムに損害を与えるのを防ぐように適切なファイルアクセス権が設定されているからです。もちろん、まだ利用される可能性のある穴がいくつか残っているかもしれませんが、この問題を心配している人は本書ではなく、Securing Debian Manual (`http://www.debian.org/doc/manuals/securing-debian-howto/`) を読むべきです。

[9] 左 Ctrl キーと d キーを同時に押すこと。これらのコントロールキーが “control D” のように大文字で書かれていても、Shift キーを押す必要は ありません。

[10] ここで *penguin* の代わりに `root` と対応したパスワードを入力すると、`root` アカウントに ログインします。この手順は `root` アカウントへのアクセスを 回復するために必要です。

Debian システムを探検するためにこのコマンドを使いましょう。これは学習するための最良の方法です。カーソルキーとエンターキーを使うだけでいくつかの主要な場所を探検してください。

- `/etc` とサブディレクトリ。
- `/var/log` とサブディレクトリ。
- `/usr/share/doc` とサブディレクトリ。
- `/sbin` と `/bin`

4.2.1 強化型 MC

終了時に作業ディレクトリを MC に変更させるためには、`~/.bashrc` (又は `.bashrc` から呼ばれる `/etc/bash.bashrc`) をマニュアルページ `mc(1)` の中の `-P` オプションの所に記述されているように修正する必要があります。[11]

4.2.2 MC の開始

```
$ mc
```

MC はメニュを通じて、最小限のユーザの苦労しか要求せずに全てのファイル操作を注意深く行います。ヘルプ表示を出すには、ただ F1 を押すだけす。カーソルキーとファンクションキーの操作だけで MC を楽しめます。[12]

4.2.3 MC でのファイル操作

MC の標準ではファイルリストを含む 2 つのディレクトリパネルがあります。他の役立つモードとして、右側のウィンドウにファイルアクセス権情報などを表示する “information” をセットするモードがあります。次にいくつかの不可欠なキーストロークを示します。`gpm` デーモンを起動させると、マウスも使えます。(MC でのカットアンドペーストの通常の挙動を回復するためには、shift キーを押してください。)

- F1: ヘルプメニュ
- F3: 内部ファイルビューア
- F4: 内部エディタ
- F9: プルダウンメニュを有効にする
- F10: Midnight Commander を終了する
- Tab: 二つのウィンドウの間を移動する
- Insert: コピーのような複数のファイル操作のため、ファイルにマークする
- Del: ファイルの削除 (気をつけて — MC を安全削除モードに設定)
- カーソルキー: 自明な動きをします

---

[11] たとえここで説明したことを正確に理解できなくとも、この作業は後でできます。

[12] 日本人用の `kon` や `kterm` のような端末上の場合は、特定のグラフィック文字に関する問題が発生します。MC のコマンドラインオプションに `-a` を追加すると、問題を防止できるかもしれません。

4.2.4 MC のコマンドライントリック

- `cd` コマンドは選択されたスクリーンに表示されたディレクトリを変更します。
- Ctrl-Enter 又は Alt-Enter はファイル名をコマンドラインにコピーします。コマンドライン編集と一緒に `cp` や `mv` コマンドで御使用ください。
- Alt-Tab はシェルファイル名の拡張子の選択肢を表示します。
- MC への引数として、両ウィンドウが開くディレクトリの指定ができます。例えば、`mc /etc/ /root` などです。
- Esc + *numberkey* == F*n* (すなわち、 Esc + '1' = F1,同様に; Esc + '0' = F10)
- Esc キー == Alt キー (= Meta, M-); すなわち、 Esc + 'c' は Alt-C と同じです。

4.2.5 MC 内のエディタ

MC の内部エディタは興味深いカットアンドペースト機構を持ちます。F3 キーを押すと、選択範囲の開始としてマークし、次に F3 を押すと、選択範囲の終了としてマークし、選択範囲を強調します。そしてカーソルを動かすことができます。F6 を押すと、選択範囲はカーソルの位置に移動します。F5 を押すと、選択範囲はコピーされ、カーソルの位置に挿入されます。F2 を押すとファイルをセーブします。F10 を押すと選択範囲はなくなります。ほとんどのカーソルキーは直感的に働きます。

このエディタはファイル上で直接起動できます。

```
$ mc -e filename_to_edit
$ mcedit filename_to_edit
```

これはマルチモードエディタではありませんが、同じ効果を発揮させるため、複数の Linux コンソール上で使用できます。ウィンドウ間のコピーを行うには、 Alt-F*n* キーを押して仮想コンソールを切替えて、ファイルの一部を他のファイルに動かすために "File->Insert file" や "File->Copy to file" を用います。

この内部エディタは他の外部エディタと置き換えが可能です。

また、多くのプログラムは使用するエディタを決定するために環境変数 `EDITOR` や `VISUAL` を使用します。 `vim` が不快な場合、`~/.bashrc` に次に示す行を追加してエディタを `mcedit` にセットします。

```
...
export EDITOR=mcedit
export VISUAL=mcedit
...
```

可能ならばこれらを `vim` にセットすることをお薦めします。 `vim` コマンドに慣れることは正しいことです。なぜならば、Vi エディタは常に Linux/Unix 世界に存在するからです。[13]

---

[13] 実際には、`vi` や `nvi` のようなプログラムも見つかります。私は、初心者のためには `vim` を代わりに選びます。 vim は F1 キーを通したヘルプを提供し、同じ位十分かつより力強いからです。もし `vim` がしっくりしない場合は、ほとんどのシステムメインテナンス作業に関して `mcedit` を使い続けて良いです。`mcedit` は 8 ビットクリーンで、あまり賢くない (テキストエンコーディングに無関心) ので、エンコーディング不明のファイルの編集の際には、むしろ有利です。`mcedit` は UTF-8 のファイルを正確に表示できません。

4.2.6 MC 内のビューア

非常に小さなビューアです。文書内の単語を検索するための素晴らしいツールです。`/usr/share/doc` ディレクトリ内のファイルに対していつもこれを使います。これは大量の Linux 情報をブラウズするための最速の方法です。このビューアはこのように直接起動できます。

```
$ mc -v filename_to_view
```

4.2.7 MC の自動起動機能

ファイルの上で Enter を押すと、適切なプログラムがファイルの内容を操作します。これは非常に便利な MC の機能です。

```
実行ファイル：コマンドを実行
man, html ファイル:ビューアソフトに内容をパイプする
tar.gz, deb ファイル: サブディレクトリであるかのように内容を表示
```

これらのビューアや仮想ファイルの機能を有効にするため、閲覧可能なファイルには実行権が設定されていなくてはなりません。`chmod` コマンドを使うか、MC のファイルメニュ経由で状態を変更してください。

4.2.8 MC の FTP 仮想ファイルシステム

MC は FTP を用いた Internet 越しのファイルアクセスに使えます。F9 を押してメニュに行き、'p' を押して FTP 仮想ファイルシステムを有効にします。`username:passwd@hostname.domainname` の形式で URL を入力すると、それがローカルにあるかのようにリモートディレクトリを取得します。

`http.us.debian.org/debian` を URL として試し、Debian ファイルアーカイブの閲覧をしてみてください。Debian アーカイブがどのように構成されているかについては 'Debian アーカイブ' on page 5 をご覧ください。

4.3 Unix ライクな作業環境

mc はほとんど全てのことを可能にしますが、シェルプロンプトから実行されるコマンドラインツールの使用方法について学び、Unix ライクな作業環境に親しむのは非常に重要なことです。[14]

---

[14] チュートリアルの章では、シェルは `bash` を意味します。他種のシェルの真実については、'シェル' on page 221 参照。

4.3.1 特別なキーストローク

Unix ライクな環境では、特別な意味を持ついくつかのキーストロークがあります。[15]

- Ctrl-U: カーソルの前の行を削除する。
- Ctrl-H: カーソルの前の文字を削除する。
- Ctrl-D: 入力を終了する。(シェルを使用中の場合、シェルを終了する)
- Ctrl-C: 起動中のプログラムを終了する。
- Ctrl-Z: 一時的にプログラムを停止する。(バックグラウンドジョブに入れる。'`command &`' on page 61 参照)
- Ctrl-S: スクリーンへの出力を止める[16]
- Ctrl-Q: スクリーンへの出力を再び有効にする。

標準のシェルである `bash` には対話的な利用を手助けする 履歴編集機能とタブ補完機能があります。

- 右矢印: コマンド履歴検索を開始。
- Ctrl-R: インクリメンタルなコマンド履歴検索を開始。
- TAB: コマンドラインに入力ファイルを補完。
- Ctrl-V TAB: コマンドラインに拡張無しで TAB を入力。

覚えておく価値のある他の重要なキーストロークを挙げます。

- Ctrl-Alt-Del: システムをリブート/停止する。'最初のインストール後にさらにいくつかパッケージを インストールする' on page 43 参照。
- マウスを左クリックしながらドラッグする:選択領域をクリップボードにコピー。
- マウスの中ボタンを押す:カーソルの位置にクリップボードの中身をペースト。
- メタキー (Emacs の専門用語) は伝統的に左側 Alt キーに割り当てられている。 あるシステムでは Window キーがメタキーに設定されている。

ここで、Linux キャラクタコンソール上でマウスを使うためには、`gpm` をデーモンとして起動する必要があります。[17] 'マウスの設定' on page 37 参照。

4.3.2 基本的な Unix コマンド

基本的な Unix コマンドを学びましょう。[18] 非特権ユーザアカウント *`penguin`* から次に挙げる コマンドを全て試してみてください。

- `pwd`
  - カレント/ワーキングディレクトリの名前を表示。
- `whoami`
  - ユーザ名を表示。
- `file` *`foo`*
  - ファイル *foo* のファイル形式を表示。
- `type -p` *`commandname`*

---

[15] 通常の Linux キャラクタコンソールでは、左側の Ctrl と Alt キーを使う ことを想定しています。

[16] `stty(1)` を使ってこれらのターミナル機能を無効にできます。

[17] X Window 環境では、マウスは Xterm プログラムを用いると同様に機能します。

[18] ここで "Unix" を純粋な意味で使用しています。Unix クローン OS は通常 等価なコマンドを提供します。Debian システムも例外ではありません。 望んだようにコマンドが働かなくても心配しないでください。 これらの例はこの順序で実行されることを意味していません。

  - *commandname* コマンドの場所を表示。
  - `which` *commandname* も同じ。[19]
- `type` *commandname*
  - *commandname* コマンドに関する情報を表示。
- `apropos` *key-word*
  - *key-word* に関するコマンドを検索。
  - `man -k` *key-word* も同様。
- `whatis` *commandname*
  - *commandname* コマンドの一行説明を表示。
- `man -a` *commandname*
  - *commandname* コマンドの説明を表示。 (Unix スタイル)
- `info` *commandname*
  - *commandname* コマンドのかなり長い説明を表示。 (GNU スタイル)
- `ls`
  - ディレクトリの内容を表示 (非ドットファイルおよびディレクトリを表示) [20]
- `ls -a`
  - ディレクトリの内容を表示。(全ファイルおよびディレクトリ)
- `ls -A`
  - ディレクトリの内容を表示。(ほとんど全てのファイルとディレクトリを表示。 すなわり、“`/..` と ”`.`“ をのぞく。)
- `ls -la`
  - 詳細な情報を含むディレクトリの全内容を表示。‘Debian でのファイルシステムの概念’ on page 67 参照。
- `ls -d`
  - カレントディレクトリ下の全ディレクトリを表示。
- `lsof` *foo*
  - ファイル *foo* のオープンされている状況を表示。
- `mkdir` *foo*
  - カレントディレクトリに *foo* という新ディレクトリ を作成。
- `rmdir` *foo*
  - カレントディレクトリにある *foo* ディレクトリを 削除。
- `cd` *foo*
  - カレントディレクトリ又は変数 `CDPATH` にリストされた ディレクトリを *foo* ディレクトリに変更する。 `builtins(7)` の `cd` コマンドに関する 部分を参照。
- `cd /`
  - root ディレクトリにディレクトリを変更。
- `cd`
  - カレントユーザのホームディレクトリにディレクトリを変更。
- `cd /`*foo*
  - ディレクトリを絶対パス指定でディレクトリ /*foo* に変更。
- `cd ..`
  - ディレクトリを親ディレクトリに変更。
- `cd ~`*foo*

---

[19] もし`Ialias`がシェル中で用いられた場合は、これらの出力は異なります。

[20] Unix は伝統的に、“`.`” で始まるファイル名を隠します。これらは 伝統的に設定情報やユーザプリファレンスを含むファイルです。

  - ディレクトリをユーザ *foo* のホームディレクトリに変更。 *foo*.
- `cd -`
  - ディレクトリを前いたディレクトリに変更。
- `</etc/motd pager`
  - 標準のページャを使って `/etc/motd` の内容を表示。‘`command < ` *foo*’ on page 62 参照。[21]
- `touch` *junkfile*
  - 空の *junkfile* ファイルを作成。
- `cp` *foo bar*
  - 既存のファイル *foo* を新規ファイル *bar* にコピーする。
- `rm` *junkfile*
  - ファイル *junkfile* を削除する。
- `mv` *foo bar*
  - 既存のファイル *foo* の名前を新しい名前 *bar* に変更する。
- `mv` *foo bar/baz*
  - 既存のファイル *foo* を新しい場所 *bar* に移動させ、名前を *baz* に変更する。ディレクトリ *bar* が存在している必要があります。
- `chmod 600` *foo*
  - 既存のファイル *foo* を他の人に読み書きできなくする。(全ての人が実行不可)
- `chmod 644` *foo*
  - 既存のファイル *foo* を他の人に読めるが、書き込みができないようにする。(全ての人が実行不可)
- `chmod 755` *foo*
  - 既存のファイル *foo* を他の人に読めるが、書き込みができないようにする。(全ての人が実行可能)
- `top`
  - フルスクリーンを用いてプロセス情報を表示。”q“ を押すと終了。
- `ps aux | pager`
  - 起動中の全プロセスの情報を BSD スタイルの出力を用いて表示。‘`command1 | command2`’ on page 61 参照。
- `ps -ef | pager`
  - 起動中の全プロセスの情報を System-V スタイルの出力を用いて表示。
- `ps aux | grep -e ”[e]xim4⋆“`
  - `exim` 又は `exim4` を起動中の全プロセスを表示。 `man grep` をタイプして、`grep(1)` マニュアルページから正規表現について学びましょう。[22]
- `ps axf | pager`
  - 起動中の全プロセスの情報を ASCII アート出力を用いて表示。
- `kill` *1234*
  - プロセス ID *1234* により識別されるプロセスを殺す。‘プロセスを殺す’ on page 111 参照。

---

[21] 素の Debian システムの標準のページャは `more` であるため、スクロールバックができません。コマンドライン `apt-get install less` を用いて `less` を インストールすることにより、`less` が標準のページャ になり、カーソルキーでスクロールバックできるようになります。

[22] 正規表現中の `[` と `[` は `grep` が 自分自身にマッチするのを避けることを可能とします。正規表現中の `4⋆` は数字 4 の 0 回以上の繰り返しを意味するので、`grep` が `exim` と `exim4` の両方にマッチすることが可能になります。 `⋆` はシェルのファイルネームのワイルドカードや正規表現で使われますが、これらの意味は異なります。

- `grep -e "`*pattern*`" *.html`
  - カレントディレクトリにある `.html` で終わる全ファイルから *"pattern"* のパターンを検索し、該当する全ファイルを表示する。
- `gzip` *foo*
  - Lempel-Ziv コーディング (L777) を用いて *foo* を圧縮し、*foo*`.gz` を作成する。
- `gunzip` *foo*`.gz`
  - *foo*`.gz` を解凍して *foo* を作成する。
- `bzip2` *foo*
  - Burrows-Wheeler ブロックソートテキスト圧縮アルゴリズムとハフマン コーディング (`gzip` より高圧縮率) を用いて *foo* を圧縮し、*foo*`.bz2` を作成する。
- `bunzip2` *foo*`.bz2`
  - *foo*`.bz2` を解凍して *foo* を作成する。
- `tar -xvvf` *foo.tar*
  - *foo*`.tar` アーカイブからファイルを展開する。
- `tar -xvvzf` *foo*`.tar.gz`
  - gzip 圧縮された *foo*`.tar.gz` アーカイブから ファイルを展開する。
- `tar -xvvf --bzip2` *foo.tar.bz2*
  - bzip2 圧縮された *foo*`.tar.bz2` アーカイブから ファイルを展開する。[23]
- `tar -cvvf` *foo*`.tar` *bar/*
  - フォルダ *bar/* の内容を *foo*`.tar` アーカイブにアーカイブする。
- `tar -cvvzf` *foo*`.tar.gz` *bar/*
  - フォルダ *bar/* の内容を *foo*`.tar.gz` アーカイブに圧縮アーカイブする
- `tar -cvvf --bzip2` *foo*`.tar.bz2` *bar/*
  - フォルダ *bar/* の内容を *foo*`.tar.bz2` アーカイブに圧縮アーカイブする [24]
- `zcat README.gz | pager`
  - 標準のページャを用いて圧縮された `README.gz` の 内容を表示する。
- `zcat README.gz > foo`
  - `README.gz` の内容を解凍してファイル `foo` を作成する。
- `zcat README.gz >> foo`
  - 圧縮された `README.gz` の内容をファイル `foo` の 末尾に追加する。(`foo` が存在しない場合、先頭に作成される。)
- `find . -name` *pattern*
  - シェルを用いて *pattern*. にマッチするファイル名を 検索する。(低速)
- `locate -d .` *pattern*
  - シェルを用いて *pattern*. にマッチするファイル名を 検索する。(定期的に作成されたデータベースを使うため高速)

訓練として、上記のコマンドを用いてディレクトリを横断し、システムの中を のぞきこんでみてください。コンソールのコマンドに関して質問がある場合は、 必ずマニュアルページを読んでみてください。例えば、これらのコマンドは 良い開始点でしょう。

```
$ man man
```

---

[23] Potato にある旧版の `tar` でも動くように、新しい短縮形の オプション `-j` の代わりに `--bzip2` をここでは用いています。

[24] 互換性を保つためにここでは `--bzip2` を用いられています。

```
$ man bash
$ man ls
```

また、`vim` を始め、F1 キーを押す良い時期でしょう。 少なくとも最初の 35 行を読むべきです。そしてカーソルを |`tutor`| まで動かし、Ctrl-] を押してオンラインコースを受けましょう。 エディタについてより詳しく学ぶには、'エディタ' on page 207 をご覧ください。

GNU や BSD 由来の多くの Unix コマンドは次のような手順 (又は ある場合には引数無し) を行えば主要なヘルプ情報を表示できます。

```
$ commandname --help
$ commandname -h
```

自習のために、 'Debian tips' on page 103 も例として試してみてください。

4.3.3 コマンドの実行

ついに Debian システムの使用方法を幾らか感じとれました。 Debian システムでのコマンド実行機構について深く見ていきましょう。 [25]

4.3.4 シンプルなコマンド

シンプルなコマンドは、次の順序となります。

1 変数の割り当て (任意)
2 コマンド名
3 引数 (任意)
4 リダイレクト (任意: >, >>, <, <<, など)
5 制御演算子 (任意: `&&`, `||`, <newline>, `;`, `&`, `(`, `)`)

引用や置換を持つより複雑なコマンドについては、'コマンドライン処理' on page 225 参照。

4.3.5 コマンド実行と環境変数

典型的なコマンドの実行は次のようなシェル行シーケンスを使用します。 [26]

```
$ date
Sun Oct 26 08:17:20 CET 2003
$ LC_ALL=fr_FR date
dim oct 26 08:17:39 CET 2003
```

---

[25] ここで、初心者のために事実を単純化しています。正確な説明は `bash(1)` 参照。

[26] 次のような出力を得るには、フランス語ロケールのインストールが必要です。'ロケール' on page 167 参照。これは必ずしもチュートリアルには必要ありません。 これはその潜在的な効果を示すためだけに行われました。

ここで、プログラム `date` はフォアグランドジョブで実行されています。 環境変数 `LC_ALL` は次のようになっています。

- 最初のコマンドでは unset (システムの標準。`C` と同等)
- 次のコマンドでは `fr_FR` (フランス語ロケール) にセット。

ほとんどのコマンドの実行は通常前の環境変数の定義を使用しません。 上の例では、次のように実行することもできます。

```
$ LC_ALL=fr_FR
$ date
dim oct 26 08:17:39 CET 2003
```

ここで見たように、コマンドの出力は環境変数により影響され、フランス語の 出力が行われました。環境変数をサブプロセスに継承されるようにしたい (すなわち、シェルスクリプトを呼ぶ時) 場合は、代わりに次のように "export" する必要があります。

```
$ export LC_ALL
```

4.3.6 コマンド検索パス

シェル上でコマンドをタイプした場合、シェルはコマンドを `PATH` 環境変数内にリストされたディレクトリのリストから検索します。`PATH` 環境変数の値は、シェルの検索パスとも呼ばれます。

標準の Debian インストールでは、ユーザアカウントの `PATH` 環境変数 には `/sbin/` が含まれないかもしれません。それゆえ、 `/sbin/` にある `ifconfig` のようなコマンドを 実行したい場合は、`PATH` 環境変数を、このディレクトリを含むように 変更する必要があります。`PATH` 環境変数は通常初期化ファイル `~/.bash_profile` に含まれます。'Bash 設定' on page 36 を ご覧ください。

4.3.7 コマンドラインオプション

いくつかのコマンドは引数を取ります。`-` 又は `--` で始まる 引数はオプションと呼ばれ、コマンドの挙動を制御します。

```
$ date
Mon Oct 27 23:02:09 CET 2003
$ date -R
Mon, 27 Oct 2003 23:02:40 +0100
```

ここでコマンドライン引数 `-R` は `date` コマンドが RFC-2822 互換のデータ文字列を出力するように挙動を変更させます。

4.3.8 シェルのワイルドカード

処理したいファイル名を全てタイプせずにファイルをコマンド処理したいことがよくあります。シェルの **ワイルドカード** を用いたファイル名のパターン展開はこのニーズを促進します。

- `*`
  - 空列を含む任意の文字列にマッチする。
  - これは ”.“ で始まるファイル名にはマッチしません。
- `?`
  - 任意の 1 文字にマッチする。
- `[...]`
  - 括弧中の 1 文字にマッチする。
- `[a-z]`
  - `a` と `z` の範囲にある文字とマッチする。
- `[^...]`
  - 括弧内 (”^“ 以外) に含まれる文字以外の文字にマッチ。

例えば、次の例を試してみて、自分で考えてみましょう。

```
$ mkdir junk; cd junk; touch 1.txt 2.txt 3.c 4.h .5.txt
$ echo *.txt
1.txt 2.txt
$ echo *
1.txt 2.txt 3.c 4.h
$ echo *.[hc]
3.c 4.h
$ echo .*
. .. .5.txt
$ echo .[^.]*
.5.txt
$ echo [^1-3]*
4.h
$ cd ..; rm -rf junk
```

4.3.9 コマンドの戻り値

コマンドは戻り値として終了ステータスを返します。

- コマンドが正常に終了した場合、戻り値は 0。
- コマンドがエラーで終了した場合、戻り値は非 0。

この戻り値は実行後すぐに `$?` シェル変数によりアクセスできます。

```
$ [ 1 = 1 ] ; echo $?
0
$ [ 1 = 2 ] ; echo $?
1
```

戻り値はシェルのために論理的な観点で使用される場合、**成功** は論理的に **真** として扱われることに注意してください。 これは **正常終了** は **ゼロ** の値を持つため、いくらか非直感的です。

'シェルの条件文' on page 224 をご覧ください。

4.3.10 典型的なコマンドシーケンス

次に挙げるシェルコマンドのイディオムを覚えてみましょう。 これらのイディオムを読んだ後に、'シェルパラメータ' on page 223, 'シェルリダイレクト' on page 223, 'シェルの条件文' on page 224, そして 'コマンドライン処理' on page 225 をご覧ください。

`command &`

`command` は **バックグラウンド** で サブシェルにて実行されます。バックグラウンドジョブにより、 複数のプログラムを単独のシェルで実行することが可能となります。

バックグラウンドプロセスの管理は次に挙げるシェル組込みコマンドに関連します。 `jobs`, `fg`, `bg`, および `kill`。 `bash(1)` マニュアルページの "SIGNALS"、 "JOB CONTROLS" および "SHELL BUILTIN COMMANDS" の章を 読んでみてください。 [27]

`command1 | command2`

`command1` の標準出力は `command2` の標準入力 に導かれます。両方のコマンドは **同時に**動くかも しれません。これは **パイプライン** と呼ばれます。

`command1 ; command2`

`command1` と `command2` は **連続して** 実行されます。

`command1 && command2`

`command1` がまず実行されます。コマンドの実行が 正常終了した場合は、`command2` も **連続して** 実行されます。`command1` と `command2` の **両方** が正常終了した場合に正常終了を返します。

`command1 || command2`

`command1` がまず実行されます。失敗した場合、 `command2` も **連続して** 実行されます。`command1` **又は** `command2` のどちらかが正常終了した場合に正常終了を返します。

---

[27] Debian システムはマルチタスクシステムです。 >

command > *foo*

`command` の標準出力をファイル *`foo`* にリダイレクト (上書き) します。

command >> *foo*

`command` の標準出力をファイル *`foo`* にリダイレクト (追記) します。

command > *foo* 2>&1

`command` の標準出力と標準エラー出力の両方をファイル *`foo`* にリダイレクトします。

command < *foo*

`command` の標準入力をファイル *`foo`* にリダイレクトします。試してみましょう。

```
$ </etc/motd pager
 ... (グリーティングメッセージを表示)
$ pager </etc/motd
 ... (グリーティングメッセージを表示)
$ pager /etc/motd
 ... (グリーティングメッセージを表示)
$ cat /etc/motd | pager
 ... (グリーティングメッセージを表示)
```

この 4 つの構文は同じ出力を表示しますが、最後の例は特別な `cat` コマンドを起動し、無駄なリソースを消費しています。

4.3.11 コマンドエイリアス

良く使うコマンドにエイリアスを設定できます。例えば、

```
$ alias la='ls -la'
```

ここで、`la` は 1 列形式で全ファイルをリストする `ls -la` の短縮形として働きます。

`type` コマンドを使って、正確なパス又はコマンドの正体を識別できます。例えば、

```
$ type ls
ls is hashed /bin/ls)
$ type la
la is aliased to `ls -la'
```

```
$ type echo
echo is a shell builtin
$ type file
file is /usr/bin/file
```

ここで `ls` は最近検索されましたが、`file` はそうではないので、`ls` は”ハッシュされた“といいます。 すなわち、シェルは `ls` コマンドの場所への高速なアクセス のために、内部記録を取っています。

4.4 Unix ライクなテキスト処理

頻繁に Unix ライクシ ステムで使用される標準的なテキスト処理ツールがいくつかあります。

- 正規表現未使用ツール。
  - `head` はファイルの先頭部分を出力します。
  - `tail` はファイルの末尾の部分を出力します。
  - `sort` はテキストファイルの行をソートします。
  - `uniq` はソートされたファイルの重複行を削除します。
  - `tr` は文字を変換したり削除します。
  - `diff` は 1 行ごとにファイルを比較します。
- 基本的な正規表現 (BRE) を使用するツール。
  - `grep` はテキストにパターンをマッチさせます。
  - `ed` は原始的なラインエディタです。
  - `sed` はストリームエディタです。
  - `vi` はスクリーンエディタです。
  - `emacs` はスクリーンエディタです。
- 拡張正規表現 (ERE) を使用するツール。
  - `egrep` はテキストにパターンをマッチさせます。
  - `awk` はシンプルなテキスト処理を行います。‘Awk’ on page 226 参照。
  - `perl` は考えられる全てのテキスト処理を行います。 ‘Perl’ on page 227 参照。

いくつかのスクリプト例は ‘正規表現による置換’ on page 117, ‘コマンドをパイプするためのわずかなスクリプト’ on page 119, そして ‘Perl ショートスクリプトの狂気’ on page 121 をご覧ください。

4.4.1 正規表現

正規表現は多くのテキスト処理ツールで使われています。シェルのワイルドカード (‘シェルのワイルドカード’ on page 60 参照) に似ていますが、より複雑かつパワフルです。

正規表現はパターンマッチを記述し、テキスト文字と *メタ文字* からなります。メタ文字は単なる特別な意味を持つ文字です。BRE と ERE の 2つの主要な形式がありますが、どちらを使うかは ‘Unix ライクなテキスト処理’ on page 63 に記述されているテキストツールの種類に依ります。

ERE の場合、**メタ文字** には ”`\ . [ ] ^ $ * + ? ( ) { } |`“ が含まれます。 正規表現は次の意味を持ちます。

- `c`
  - 非メタ文字の ”`c`“ にマッチ。
- `\c`
  - 文字通り ”`c`“ にマッチ。
- `.`
  - 空白を含む全文字にマッチ。
- `^`
  - 文字列の先頭にマッチ。
- `$`
  - 文字列の末尾にマッチ。
- `\<`
  - 単語の先頭にマッチ。
- `\>`
  - 単語の末尾にマッチ。
- `[abc...]`
  - ”`abc...`“ のいずれかの文字にマッチ。
- `[^abc...]`
  - ”`abc...`“ 以外のいずれかの文字にマッチ。
- `r*`
  - ”`r`“ で認識される 0 回を含む正規表現の繰り返しにマッチ。
- `r+`
  - ”`r`“ で認識される 1 回以上の正規表現の繰り返しにマッチ。
- `r?`
  - ”`r`“ で認識される 0 回を含む正規表現の繰り返しにマッチ。
- `r1|r2`
  - ”`r1`“ 又は ”`r2`“ で認識される正規表現のいずれか にマッチ。
- `(r1|r2)`
  - ”`r1`“ 又は ”`r2`“ で認識される正規表現のいずれか にマッチし、**ひとまとめの** 正規表現として扱われます。

BRE では、**メタ文字** ”`+ ? ( ) { } |`“ はその特別な意味を失っています。その代わり、バックスラッシュされた ”`\+ \? \( \) \{ \} \|`“ を使います。それゆえ、grouping construct `(r1|r2)` は BRE では `\(r1|r2\)` の ように引用符で囲む必要があります。`emacs` は、基本的に BRE を使いますが、”`+ ?`“ を **メタ文字** として 扱います。それゆえ、引用符で囲む必要はありません。grouping construct がどのように使われるかについては ‘置換表現’ on page 65 をご覧ください。

例えば、`grep` は正規表現を用いてテキスト検索を行うために使えます。

```
$ egrep 'GNU.*LICENSE|Yoyodyne' /usr/share/common-licenses/GPL
                    GNU GENERAL PUBLIC LICENSE
                    GNU GENERAL PUBLIC LICENSE
  Yoyodyne, Inc., hereby disclaims all copyright interest in the program
```

4.4.2 置換表現

置換表現の場合、次の文字には特別な意味があります。

- `&`
  - 正規表現がマッチするものを示す。(`emacs` では `\&` を使用)
- `\`*n*
  - *n* 番目に **括弧で囲まれた** 正規表現に マッチするものを示す。

Perl の置換文字列の場合、`$`*n* は `\`*n* の代わりに用いられ、`&` は 特別な意味を持ちません。

例を示します。

```
$ echo zzz1abc2efg3hij4 | \
  sed -e ’s/\(1[a-z]*\)[0-9]*\(.*\)$/=&=/’
zzz=1abc2efg3hij4=
$ echo zzz1abc2efg3hij4 | \
  sed -e ’s/\(1[a-z]*\)[0-9]*\(.*\)$/\2===\1/’
zzzefg3hij4===1abc
$ echo zzz1abc2efg3hij4 | \
  perl -pe ’s/(1[a-z]*)[0-9]*(.*)$/$2===$1/’
zzzefg3hij4===1abc
$ echo zzz1abc2efg3hij4 | \
  perl -pe ’s/(1[a-z]*)[0-9]*(.*)$/=&=/’
zzz=&=
```

ここで、**括弧で囲まれた** 正規表現と、マッチした文字列が異なったツールでテキスト置換処理にどのように使われるかに特別な注意を払ってください。

これらの正規表現はエディタ内でカーソルの動きとテキスト置換アクションの両方 に対して使えます。

これらのコマンドを習うために、関連するマニュアルページを全て読んでください。

4.5 Unix ライクなファイルシステム

GNU/Linux や他の Unix ライク OS システムでは、**ファイル** は **ディレクトリ** に組織化されています。[28] 全ての **ファイル** と **ディレクトリ** は `/` を根とする 1 つの大きなツリー、すなわちファイル階層に配置 されています。

これらのファイルは複数のデバイスに展開することができます。`mount(8)` コマンドはあるデバイス上にあるファイルシステムを大きなファイルツリーに所属させます。逆に、`umount(8)` コマンドは再び分離させます。

---

[28] **ディレクトリ** はいくつかのシステムでは **フォルダ** と呼ばれています。

4.5.1 Unix ファイルの基礎

次に挙げることは基礎知識です。

- ファイル名は大文字と小文字を区別します。すなわち、`MYFILE` と `MyFile` は **異なる** ファイルです。
- root ディレクトリは単に `/` として参照されます。 root ユーザとこの ”root“ を混同しないでください。‘root としてシェルにログインする’ on page 47 参照。
- 全てのディレクトリ名には `/` **を除く**全ての 文字と記号を使用できます。[29] root ディレクトリは例外です。名前は `/` (”スラッシュ“ や ”root ディレクトリ“ と呼ぶ) であり、変えられません。
- 各ファイルおよびディレクトリは **完全に記述したファイル名**、**絶対ファイル名** や **パス** により 指定され、たどっていくとファイルに到達するディレクトリの列を与えます。 この 3 つの用語は同意語です。全ての絶対ファイル名は `/` ディレクトリで始まり、ファイル名内のディレクトリやファイルの間には `/` がはさまります。最初の `/` はディレクトリ名 ですが、その他はファイル名の部分部分を区別するための単なるセパレータです。 ここで使われる単語は混乱を招きます。次の例をご覧ください。

  ```
  /usr/share/keytables/us.map.gz
  ```

  これは完全に記述されたファイル名です。ある人はこれを パス と呼びますが、`us.map.gz` だけをファイル名として呼ぶ人もいます。[30]
- root ファイルシステムは `/etc/` や `/usr/` のような複数の枝を持ちます。これらのサブディレクトリもまた `/etc/init.d/` や `/usr/local/` のように、さらに サブディレクトリに枝別れします。これらの全体をまとめて **ディレクトリツリー** と呼びます。絶対ファイル名はツリーの根元 (/) から枝の先 (ファイル) までの経路として考えることもできます。また、あたかもディレクトリツリーを **家系図** のように人が話すのを聞いたことがあるでしょう。 あたかもそれぞれのサブディレクトリに **親** があるとし、 パスはファイルの完全な祖先の系図のように表現します。 root ディレクトリではない他の場所から始まる相対パスもあります。 `../` は親ディレクトリを参照していることを覚えておきましょう。
- ハードディスクのような物理デバイスに対応したディレクトリは存在しません。 ここが 全てのパスが `C:` のようなデバイス名から始まる CP/M、DOS、そして Windows と違う点です。‘Debian でのファイルシステムの概念’ on page 67 をご覧ください。

ファイル階層について詳細に学ぶ最も良い方法は、 Filesystem Hierarchy Standard (`/usr/share/doc/debian-policy/fhs/fhs.txt.gz`) に記述されています。 手始めとして次の事実を覚えるべきです。

- `/`
  - 単なる `/` は root ディレクトリを表す。
- `/etc/`
  - ここはシステム全体の設定ファイルを置く場所です。
- `/var/log/`

---

[29] ファイル名にはほとんど全ての文字や記号を**使えます**が、 実際には使うべきではない文字や記号があります。コマンドラインで特別な意味 を持つ文字を使うのを避けるのが良いでしょう。それらには、スペース、タブ、 改行や他の特別な文字: `` { } ( ) [ ] ’ ` " \ / > < | ; ! # & ^ * % @ $ `` が含まれます。 複数の単語からなるファイル名を付けたい場合、ピリオド、ハイフンや下線を 使うのが良い選択です。各単語の先頭文字を、`LikeThis` のように 大文字にするのも良いでしょう。

[30] **パス** という単語の他の使いかたもあります。 ‘コマンド検索パス’ on page 59 をご覧ください。意図された意味は通常文脈から 判断するとはっきりします。

  - ここはシステムのログファイルを置く場所です。
- `/home/`
  - ここは全ての非特権ユーザのホームディレクトリがあるディレクトリです。

4.5.2 Debian でのファイルシステムの概念

Unix の伝統に従い、Debian システムはハードディスクや他のストレージデバイス上の物理データ、そしてコンソールスクリーンなどのハードウェアデバイスとリモートのシリアルコンソールとの相互作用を統一された形式で表されるファイルシステムを供給します。

Debian での各ファイル、ディレクトリ、名前付きパイプ、そして物理データはそのデータの所有者 (owner)、データが所属するグループ、最後にアクセスされた時間などの付随する属性を記述する **inode** と呼ばれるデータ構造を持ちます。Debian GNU/Linux システムでの `inode` 構造の正確な定義を知るには、`/usr/include/linux/fs.h` をご覧ください。

物理エントリの統一された表現は非常にパワフルなので、完全に異なるデバイスに同種の操作を行うために、同じコマンドを使用することを可能とします。

すべてのファイルは 1 つのディスク上に置けますし、20 個のディスクに分けることも、ネットワーク上のどこかにある違うコンピュータにあって接続することもできます。ファイルの物理的位置がどこであれ、ディレクトリツリーを見るだけでは見分けがつきませんし、またほとんど全てのコマンドは同じように動きます。

4.5.3 ファイルとディレクトリのアクセス権

ファイルとディレクトリのアクセス権はそれぞれ次に挙げる3つの影響されるユーザのカテゴリのために定義されています。
- ファイルを所有する **ユーザ** (u)。
- ファイルが所属する **グループ** のユーザ (g)。そして
- 全ての **他の** ユーザ (o)。

ファイルでは、それぞれに対応する権限は次のようになります。
- **読み込み** (r): ファイルの内容を検査する権限。
- **書き込み** (w): ファイルの修正を行う権限。そして
- **実行** (x): ファイルをコマンドとして実行する権限。

ディレクトリでは、対応する権限はそれぞれ次のようになります。
- **読み込み** (r): ディレクトリの内容をリストする権限。
- **書き込み** (w): ディレクトリにファイルを追加したり削除する権限。そして
- **実行** (x): ディレクトリ内のファイルにアクセスする権限。

ここで、ディレクトリでの **実行** 権はディレクトリにあるファイルの読み込みを許可するだけでなく、サイズや修正時間などの属性を見ることも許可します。

ファイルやディレクトリのアクセス権情報 (やそれ以上) を表示するには、`ls` が使われます。`ls(1)` をご覧ください。`ls` を `-l` オプション付きで実行すると、次の情報がこの順序で表示されます。
- **ファイルタイプ** (先頭文字)

  - `-`: 通常のファイル
  - `d`: ディレクトリ
  - `l`: シンボリックリンク
  - `c`: キャラクタデバイス node
  - `b`: ブロックデバイス node
  - `p`: 名前付きパイプ
  - `s`: ソケット
- ファイルの **アクセス権** (次の 9文字。ユーザ、グループ、そして他者それぞれに対して 3つの文字から構成)
- そのファイルへの **ハードリンク数**
- ファイルを所有する **ユーザ** の名前
- ファイルが所属する **グループ** の名前
- ファイルのバイト単位での **サイズ**
- ファイルの **タイムスタンプ** (mtime)
- ファイルの **名前**

ファイルの所有者を変更するには、root アカウントから `chown` を 使用します。ファイルのグループを変更するには、ファイルの所有者又は root アカウントから `chgrp` を使用します。ファイルやディレクトリの アクセス権を変更するには、ファイルの所有者又は root アカウントから `chmod` を使います。`foo` ファイルの操作を行う 基本的な文法は次の通り。

```
# chown newowner foo
# chgrp newgroup foo
# chmod  [ugoa][+-=][rwx][,...] foo
```

詳細は `chown(1)`,`chgrp(1)`,および `chmod(1)` をご覧ください。

例えば、ディレクトリツリーの所有者をユーザ *foo* に変更し、 グループ *bar* で共有する場合、root アカウントから次のコマンドを 実行します。

```
# cd /some/location/
# chown -R foo:bar .
# chmod -R ug+rwX,o=rX .
```

3つのさらに専門的なアクセス権ビットが存在します。

- **setuid** ビット (ユーザの x の代わりに s 又は S)
- **setgid** ビット (グループの x の代わりに s 又は S)
- **sticky** ビット (他者の x の代わりに t 又は T)

ここで `ls -l` のこれらのビットを表す出力は、これらのビットが出力されることにより隠される実行ビットがセットされていない場合、大文字になります。

setuid ビット を実行ファイルにセットすると、ユーザは ファイルの所有者 ID (例えば、**root**) により実行ファイルを 実行することを許可します。同様に、 **setgid** ビット を実行ファイルにセットすると、ユーザは ファイルのグループ ID (例えば、**root**) により実行ファイルを 実行することを許可します。これらの設定はセキュリティを破るリスクを引き起こす ので、これらのビットを有効にするには特別な注意が必要です。

setgid ビット をディレクトリに対して有効にすると、ディレクトリに作成した全ファイルがディレクトリの **グループ** に所属するという BSD ライクなファイル生成手法が有効になります。

sticky ビット をディレクトリに対して有効にすると、ディレクトリにあるファイルがファイルの所有者以外から削除されるのを防ぎます。`/tmp` やグループの書き込み可能なディレクトリなどの world-writable なディレクトリにあるファイルの内容を安全にするためには、**書き込み** 権を無効にするだけでなく、ディレクトリに **sticky** ビット もセットする必要があります。さもなければ、ディレクトリに書き込みできるユーザにより、ファイルが削除され、同じ名前で新しいファイルが作成されることを許してしまいます。

次にファイルアクセス権の興味深い例を挙げます。

```
$ ls -l /etc/passwd /etc/shadow /dev/ppp /usr/sbin/pppd
crw-rw----    1 root     dip      108,   0 Jan 18 13:32 /dev/ppp
-rw-r--r--    1 root     root         1051 Jan 26 08:29 /etc/passwd
-rw-r-----    1 root     shadow        746 Jan 26 08:29 /etc/shadow
-rwsr-xr--    1 root     dip        234504 Nov 24 03:58 /usr/sbin/pppd
$ ls -ld /tmp /var/tmp /usr/local /var/mail /usr/src
drwxrwxrwt    4 root     root         4096 Feb  9 16:35 /tmp
drwxrwsr-x   10 root     staff        4096 Jan 18 13:31 /usr/local
drwxrwsr-x    3 root     src          4096 Jan 19 08:36 /usr/src
drwxrwsr-x    2 root     mail         4096 Feb  2 22:19 /var/mail
drwxrwxrwt    3 root     root         4096 Jan 25 02:48 /var/tmp
```

`chmod(1)` コマンドを用いて、ファイルアクセス権を記述するためのもう一つの数字モードが存在します。この数字モードは8進数 を使った4桁の数字を用います。各桁は次のように対応します。

- 最初の任意桁:: **setuid** (=4)、**setgid** (=2)、**sticky** ビット (=1) の合計。
- 2 番目の桁: 所有者に対する **読み込み** (=4)、**書き込み** (=2)、**実行** (=1) 権の合計。
- 3 番目の桁: **グループ** に対する同様の合計。
- 4 番目の桁: **他者** に対する同様の合計。

これは複雑に聞こえるかもしれませんが、実際は本当にシンプルです。`ls -l` コマンドの出力の最初の数行を見て、それをファイルアクセス権の バイナリ (2進数) 表現 ("-" を "0"、"rwx" を "1) として読むと、この数字モードの値はファイルアクセス権の 8 進数表現として意味を持ちます。[31] 例えば、次を試してみてください。

```
$ touch foo bar
$ chmod u=rw,go=r foo
$ chmod 644 bar
$ ls -l foo bar
-rw-r--r--    1 penguin  penguin  0 Nov  3 23:30  foo
-rw-r--r--    1 penguin  penguin  0 Nov  3 23:30  bar
```

標準のファイルアクセス権マスクは `umask` シェル組込み コマンドを使うことによりセットできます。`builtins(7)` をご覧ください。

---

[31] もちろんこの方法は 3 桁の数字モードの場合のみ機能します。

4.5.4 タイムスタンプ

GNU/Linuxのファイルには、3種類のタイプスタンプが存在します。

- **mtime**: 修正時刻 (`ls -l`)
- **ctime**: ステータス変更時刻 (`ls -lc`)
- **atime**: 最後にアクセスした時刻 (`ls -lu`)

ctimeが作成時刻ではないことに注意してください。

- ファイルを上書きすると、そのファイルの**mtime**、 **ctime**、**atime**の全てが変更されます。
- ファイルのアクセス権や所有者を変更すると、**ctime**、 **atime**が変更されます。
- ファイルを読み込むと、そのファイルの**atime**が 変更されます。

Debianシステム上では、ファイルをただ読み込むだけで通常ファイルの書き込み操作 が発生し、**inode**の **atime**情報が更新される ことに注意してください。ファイルシステムを`noatime`オプション付きで マウントすると、システムはこの操作を飛ばし、ファイル読み込みアクセスが より高速になります。`mount(8)`をご覧ください。

既存のファイルのタイムスタンプを変更するには、`touch(1)`をご覧ください。

4.5.5 リンク

異なるファイル*bar*とファイル*foo*を連携させる2つの方法があります。

- ハードリンク は既存のファイルに対する重複した名前です。 (`ln foo bar`)
- シンボリックリンク 又は “symlink” は他のファイルを 名前により指示する特別なファイルです。 (`ln -s foo bar`)

リンク数の変更と`rm`コマンドの結果の厳密な違いについての次の例 をご覧ください。

```
$ echo "Original Content" > foo
$ ls -l foo
-rw-r--r--    1 osamu    osamu           4 Feb  9 22:26 foo
$ ln foo bar     # hard link
$ ln -s foo baz  # symlink
$ ls -l foo bar baz
-rw-r--r--    2 osamu    osamu           4 Feb  9 22:26 bar
lrwxrwxrwx    1 osamu    osamu           3 Feb  9 22:28 baz -> foo
-rw-r--r--    2 osamu    osamu           4 Feb  9 22:26 foo
$ rm foo
$ echo "New Content" > foo
$ cat bar
Original Content
$ cat baz
New Content
```

シンボリックリンクは上の例に示したように、常にファイルアクセス権 “rwxrwxrwx” を持ちますので、シンボリックリンクが指すファイルのアクセス権により指示される ファイルアクセス権が実際の効果を持ちます。

. ディレクトリは、それ自身が中にあるディレクトリとリンクしていますので、新しいディレクトリリンク数は 2 から始まります。**..** ディレクトリ は親ディレクトリとリンクしているので、ディレクトリのリンク数は 新しいサブディレクトリの増加に伴い増加します。

4.5.6 名前付きパイプ (FIFO)

名前付きパイプは、パイプのように動くファイルです。何かをファイルに 入れ、もう一方から出ます。こうしてこれは FIFO 又は First-In-First-Out と呼ばれます。つまり、最初にパイプに入れるものが最初にもう一方から出てきます。

名前付きパイプに書き込んだ場合、パイプに書き込んだプロセスは情報が パイプから読み込まれるまで終了しません。名前付きパイプを読み込んだ場合、 読み込んだプロセスは何か読み込むものが存在するまで終了するのを待ちます。 パイプのサイズは常に 0 です。– 名前付きパイプはデータを保存せず、 シェルの | のように 2 つのプロセスをリンクするだけです。しかし、このパイプは名前を持つので、2 つのプロセスは同じコマンドライン にあったり、同じユーザにより実行される必要さえありません。

次を実行することにより、名前つきパイプを試せます。

```
$ cd; mkfifo mypipe
$ echo "hello" >mypipe & # バックグラウンドに入れる
[1] 5952
$ ls -l mypipe
prw-r--r--    1 penguin penguin  0 2003-11-06 23:18 mypipe
$ cat mypipe
hello
[1]+  Done                     echo hello >mypipe
$ ls mypipe
prw-r--r--    1 penguin penguin  0 2003-11-06 23:20 mypipe
$ rm mypipe
```

4.5.7 ソケット

ソケットは名前つきパイプ (FIFO) に似ており、プロセス間の情報交換を 行えます。ソケットにとって、これらのプロセスは同時に実行する必要 はありませんし、同じ祖先プロセスの子供である必要もありません。 これはプロセス間通信の終端点です。情報の交換は異なるホストの間の ネットワーク越しでさえ発生します。

4.5.8 デバイス名

デバイスファイルは、システム上のハードディスク、ビデオカード、ディスプレイ、キーボードなどの物理デバイス又は仮想デバイスに関連します。仮想デバイスの 例として、`/dev/console` として表されるコンソールがあります。

デバイスには 2つのタイプがあります。

- キャラクタデバイス
  - 一度に 1 つの文字にアクセス、すなわち、デバイスから書き込みや読み込みを 行うデータの最小単位が文字 (バイト) であるデバイスです。
- ブロックデバイス
  - ブロックと呼ばれる、多量の文字を含むより大きな単位でアクセスされる デバイスです。ハードディスクはブロックデバイスです。

デバイスファイルの読み書きが可能ですが、ファイルには、人間にとっては ちんぷんかんぷんで理解できないバイナリデータが多分含まれます。 データを直接デバイスファイルに書き込むことは時々ハードウェアの 接続に関するトラブルシュートに役立ちます。例えば、プリンタデバイス `/dev/lp0` にテキストデータをダンプしたり、 適切なシリアルポート `/dev/ttyS0` にモデムコマンドを送る ことができます。しかし、注意深くやらないと、大災害をもたらすことがあります。 ですから気をつけてください。

`/dev/null` など

`/dev/null` は書き込む全てを捨てる特別なデバイスファイルです。 何かをしたくない場合、それを `/dev/null` に投げ込みます。 これは基本的に底無し穴です。`/dev/null` を読み込むと、 ファイル終端 (EOF) 文字がすぐさま得られます。

`/dev/zero` は同様に、ここから読み込むと、`\0` 文字 を取得します。(ASCII コードの 0 とは異なる) 'ダミーファイル' on page 126 を ご覧ください。

デバイスノード数

デバイスノード数は `ls` を次のように実行すると表示されます。

```
$ ls -l /dev/hda /dev/ttyS0 /dev/zero
brw-rw----    1 root     disk       3,   0 Mar 14  2002 /dev/hda
crw-rw----    1 root     dialout    4,  64 Nov 15 09:51 /dev/ttyS0
crw-rw-rw-    1 root     root       1,   5 Aug 31 03:03 /dev/zero
```

ここで、

- `/dev/hda` はmajor device number 3 と minor device number 0 を持ちます。これは `disk` グループに所属するユーザにより、読み書き が可能です。
- `/dev/ttyS0` は major device number 4 と minor device number 64 を持ちます。これは `dialout` グループに所属するユーザにより読み書き が可能です。
- `/dev/zero` は major device number 1 と minor device number 5 を持ちます。これはだれでも読み書き可能です。

古いシステムでは、インストール過程で `/sbin/MAKEDEV` コマンド を使ってデバイスノードを作成します。 `MAKEDEV(8)` をご覧ください。

新しいシステムでは、`/dev` 下のファイルシステムは `/proc` ファイルシステムと同様のデバイスファイルシステムにより 自動的に生成されます。

4.5.9 `/proc` ファイルシステム

`/proc` ファイルシステムは仮想ファイルシステムであり、システムや起動中のプロセスに関する情報を含みます。

特にあるファイル - `/proc/kcore` に気づくと、パニックになる人が良くいます。これは一般に巨大です。これは (おおよそ) コンピュータのメモリの内容のコピーです。これは kernel をデバッグするのに用いられます。実際にはどこにも存在しないので、そのサイズに関して心配する必要は全くありません。

'proc filesystem により Linux kernel をチューニングする' on page 100 と `proc(5)` をご覧ください。

4.6 X Window System

'X' on page 141 をご覧ください。

4.6.1 X Window System を起動する

X Window System は `xdm` のようなグラフィカルなログインデーモンにより自動的に起動できますし、コンソールから次をタイプしても起動できます。

```
$ exec startx
```

4.6.2 X Window System におけるメニュ

X 環境は多くのウィンドウマネージャを提供するので、ユーザインターフェースはかなり変化します。root ウィンドウを右クリックするとメニュインターフェースが上がって来ます。これはいつも使えます。

- シェルコマンドプロンプトを得るには、メニュから Xterm を起動します。
  - "XShells" –> "XTerm".
- web ページのグラフィカルなブラウズを行うには、メニュから Mozilla を起動します。
  - "Apps" –> "Net" –> "Mozilla Navigator".
- PDF ファイルのグラフィカルなブラウズを行うには、メニュから Xpdf を起動します。
  - "Apps" –> "Viewers" –> "Xpdf".

メニュエントリが見付からない場合、関連するパッケージをインストールしてください。'Debian パッケージ管理の基礎' on page 80 をご覧ください。

4.6.3 X Window System でのキーボードシーケンス

X Window System を起動中に覚えておくべき重要なキーストロークを次に挙げます。

- Ctrl-Alt-F1 から F6: 他の仮想端末に切替える (X window、DOSEMU などから)
- Alt-F7: X window に戻る
- Ctrl-Alt-minus: 画面の解像度を変更 (マイナスは数字キーパッド にあるキー)
- Ctrl-Alt-plus: 画面の解像度を逆向きに変更 (プラスは数字 キーパッドにあるキー)
- Ctrl-Alt-Backspace: X サーバプロ グラムを終了
- Alt-X, Alt-C, Alt-V: 良く使う Window/Mac のカット、コピー、ペーストキー の組合せで使う Ctrl- キーはこれらの Alt- キーで Netscape Composer のような いくつかのプログラムでは置き換えられる。

4.7 Further study

現時点では、The Linux Documentation Project: Guides (`http://www.tldp.org/guides.html`) から引用する主要なガイドブックを読むことをお薦めします。

- "The Linux System Administrators' Guide",
  - システムの起動、ユーザアカウント、バックアップ、システム設定を 維持する見地を全てカバーしている。
  - パッケージ: `sysadmin-guide`
  - ファイル: `/usr/share/doc/sysadmin-guide/html/index.html`
  - web: `http://www.tldp.org/LDP/sag/index.html`
- "The Linux Network Administrator's Guide, Second Edition",
  - Linux 環境でのネットワーク管理のためのただ一つのリファレンスです。
  - パッケージ: `(not available)`
  - ファイル: `(notapplicable)`
  - web: `http://www.tldp.org/LDP/nag2/index.html`
- *Linux: Rute User*の入門解説書
  - GNU/Linuxシステム管理に関する非常に良い オンラインとハードカバー本
  - 著者: Paul Sheer
  - 出版社: Prentice Hall
  - パッケージ: `rutebook` (from `non-free`)
  - ファイル: `/usr/share/doc/rutebook/`

さらに学ぶためのリソースは 'Debian のサポート' on page 247 をご覧ください。

章 5

stable, testing, 又は unstable ディストリビューションへのアップグレード

アップグレード用の公式リリースノートは `http://www.debian.org/releases/stable/releasenotes` および `http://www.debian.org/releases/testing/releasenotes` (作業中) にあります。

`stable`, `testing`, 又は `unstable` ディストリビューションにシステムをアップグレードするには、次に示す数段階の手順を踏むことが必要となるかもしれません。

- Woody へのアップグレード (システムが Woody より前の場合)
- `stable` へのアップグレード
- `testing` へのアップグレード
- `unstable` へのアップグレード

Debian は中間のリリースを飛ばしたアップグレードをサポートしていません。

5.1 Potato から Woody へのアップグレード

この手順は Potato の APT が現在の `apt_preferences(5)` マニュアルページに記述 されている全ての機能をサポートしていないために分けて記述されています。

`/etc/apt/sources.list` に `stable` ソースだけを 含めた後に、次を実行することにより、`stable` 版に必要な APT と主要パッケージをアップグレードします。

```
# apt-get update
# apt-get install libc6 perl libdb2 debconf
# apt-get install apt apt-utils dselect dpkg
```

次にシステムの残りの部分を Woody にアップグレードします。

```
# apt-get upgrade
# apt-get dist-upgrade
```

5.2 アップグレードの準備

ネットワーク経由でパッケージを取得することにより、あるディストリビューション から他のディストリビューションにアップグレードすることができます。 アップグレード作業は次のように行います。

まず `stable` 用のリポジトリのクリーンなリストを取得します。

```
# cd /etc/apt
# cp -f sources.list sources.list.old
# :>sources.list
# apt-setup noprobe
```

`testing` にアップグレードしたい場合、この新しいリストに `testing` ソースを追加します。 `unstable` にアップグレードしたい場合には、`unstable` ソースも追加します。

```
# cd /etc/apt
# grep -e "^deb " sources.list >srcs
# :>sources.list
# cp -f srcs sources.list
# sed -e "s/stable/testing/" srcs >>sources.list
# sed -e "s/stable/unstable/" srcs >>sources.list
# apt-get update
# apt-get install apt apt-utils
```

`/etc/apt/sources.list` と `/etc/apt/preferences` のチューニング技術については 'Debian パッケージ管理の基礎' on page 80 をご覧ください。

5.3 アップグレード

上記に示したように `/etc/apt/sources.list` と `/etc/apt/preferences` を適切に設定した後、システムの アップグレードを開始できます。

しかしながら、Debian の `testing` ディストリビューションを 追いかけることには、更新されたパッケージはまず `unstable` に アップロードされ、後で `testing` に移って来るために、 セキュリティ修正を含むパッケージのインストールが 遅れるという副作用がありえるということに注意してください。

基本的なことは 'Debian パッケージ管理' on page 79 をご覧ください。そして、問題に 直面した場合は 'APT アップグレードのトラブルシュート' on page 86 をご覧ください。

5.3.1 **dselect** を使用する

システムに `-dev` パッケージなどを含む多くのパッケージが存在する場合、`dselect` を使用した次のような手段を用いたきめの細かいパッケージ 制御がお薦めです。

```
# dselect update   # アップグレード前には常にこれを実行する
# dselect select   # 追加パッケージを選択する
```

`dselect`を起動した時、現在の全パッケージが選択されています。 `Depend`, `Suggest`, そして `Recommends` に基づく 追加パッケージがある場合、`dselect` が入力を促すかもしれせん。 パッケージを追加したくない場合は、ただ `Q` を押せば、`dselect` は再び終了します。

```
# dselect install
```

`dselect` の install 作業中、パッケージの設定に関する質問に いくつか答える必要があるでしょう。ですから、この過程のために時間とノート を用意しておきましょう。'`dselect`' on page 82 をご覧ください。

`dselect` を使用してください。 **常に動作します :)**

5.3.2 **apt-get** を使用する

```
# apt-get update
# apt-get -t stable upgrade
# apt-get -t stable dist-upgrade
# apt-get -t testing upgrade
# apt-get -t testing dist-upgrade
# apt-get -t unstable upgrade
# apt-get -t unstable dist-upgrade
```

いったんシステムが Sarge まで到達したら、`apt-get` の代わりに `aptitude` を使うことをお薦めします。 (`aptitude` は 上に示したものを含めた `apt-get` が受け取るオプションの多くを受け取ります。)

アップグレードを行い、現在の `dselect` 設定を保持するには、次を実行します。

```
# apt-get dselect-upgrade
```

'パッケージの依存性' on page 16 をご覧下さい。

章 6

Debian パッケージ管理

現在 `aptitude` は APT のコンソール版のフロントエンドと みなされています。過去にインストールしたパッケージと依存性を通じて 引っ張って来たパッケージを故意に記憶しています。 故意にインストールしたパッケージによりもはや必要とされなくなった場合、 後者のタイプのパッケージは `aptitude` により自動的に 削除されます。先進的なパッケージフィルタ機能を持ちますが、設定するのは 難しいでしょう。

現在 `synaptic` は APT の Gtk 版 GUI フロントエンドと みなされています。 `aptitude` よりも設定しやすいパッケージフィルタ機能を持ちます。 又、Debian Package Tags (`http://debtags.alioth.debian.org/`) を実験的にサポート しています。

Debian のリポジトリへのネットワーク負荷を低減し、ダウンロードを 高速化するためには、パッケージを Debian のミラーサイトから ダウンロードすべきです。

ローカルネットワーク内の複数のマシンに同じパッケージをインストール する必要がある場合、APT を通じたパッケージのダウンロードのために `squid` を使ってローカルの HTTP プロキシを立ち上げる ことができます。必要ならば、`http_proxy` 環境変数を設定 したり、`/etc/apt/apt.conf` に `http` 値を 設定してください。

`apt_preferences(5)` に記載された pin 機能は強力ですが、その影響について理解や管理がしずらい場合が あります。これを先進的な機能とみなすべきです。

'`chroot`' on page 126 に記載された手法を用いることは、システムの安定性と最新版のソフトウエアへのアクセスの両方を確保できる ために、望ましい手法です。

本章は Woody 以後のシステムに準拠していますが。 いくつかの機能は Sarge 以後であることを要求するかもしれません。

6.1 イントロダクション

もし全てのデベロッパー向けのドキュメンテーションを読むのが辛いなら、本章を まず読み、Debian の `testing/unstable` によるフルパワーを 楽しみ始めてください :-)

6.1.1 主要なパッケージ管理ツール

`dpkg` - Debian パッケージファイル インストーラ
`apt-get` - APT のコマンドラインフロントエンド
`aptitude` - APT の先進的なコンソール版コマンドラインフロントエンド
`synaptic` - APT の Gtk GUI フロントエンド
`dselect` - メニュドリブンなパッケージマネージャ
`tasksel` - タスクインストーラ

これらはお互いに置き換わるものではありません。 例えば、`dselect` は APT も `dpkg` も使用しています。

APT は 取得可能なパッケージを追跡するために `/var/lib/apt/lists/*` を使いますが、`dpkg` は `/var/lib/dpkg/available` を使います。 `aptitude` や他の APT のフロントエンドを使ってパッケージを インストールしてきて、インストールに `dselect` を使いたい場合、まず 最初にすべきことは `dselect` のメニュから `[U]pdate` を選択する (又は "`dselect update`" を起動する) ことです。

`apt-get` は自動的にインストールを要求したパッケージに Depends する パッケージをインストールします。 インストールを要求したパッケージに単に Recommends や Suggests するパッケージはインストールしません。

一方、`aptitude` はインストールを要求したパッケージに Recommends や Suggests するパッケージをインストールするように設定できます。

`dselect` は選択したパッケージが Recommends や Suggests するパッケージ のリストをユーザに提供し、それらを個別にインストールするかどうかを選択できます。 'パッケージの依存性' on page 16 参照。

6.1.2 便利なツール

`dpkg-reconfigure` - 既にインストールしたパッケージを再設定する (`debconf` を使っている場合)
`dpkg-source` - source パッケージファイルを管理する
`dpkg-buildpackage` - パッケージファイルの構築を自動化する
`apt-cache` - ローカルキャッシュにあるパッケージアーカイブをチェックする

6.2 Debian パッケージ管理の基礎

'アップグレードの準備' on page 76 に記述されているように `sources.list` を設定しましょう。[1] 'Debian システムインストールのヒント' on page 27, '`stable`, `testing`, 又は

---

[1] `testing` や `unstable` を追いかけている場合、`/etc/apt/sources.list` や `/etc/apt/preferences` から `stable` への参照を削除できます。その理由は `testing` が `stable` のコピーとして始まっているからです。

`unstable` ディストリビューションへのアップグレード’ on page 75, や ‘緊急用エディタ’ on page 208 でも言及しています。

6.2.1 task のインストール

特定の使用目的で Debian システムに入れ込むために要求される主要なパッケージ群をインストールできます。 これらのパッケージ群は “task” と呼ばれます。

最初のインストール時に task をインストールする最も単純な方法は `tasksel` を使用することです。 使用前に

```
dselect update
```

を起動させる必要がある ことに注意してください。

`aptitude` も task をインストールでき、この目的に 用いるために推奨できるツールです。インストールの手順に進む前に task 内の個別のパッケージを削除 することを可能とします。

6.2.2 `aptitude`

`aptitude` は `dselect` に似たメニュドリブンな新しい パッケージ インストーラですが、APT の上に 1 から構築されています。 `apt-get` の代わりのコマンドラインコマンドとしても使えます。 `aptitude(1)` と `/usr/share/doc/aptitude/README` をご覧ください。

いったん `aptitude` を使いはじめたら、他のパッケージを インストールする手段を用いずに使い続けるのが良いでしょう。 そうでないと意図してインストールしたパッケージを追跡するという `aptitude` の利点を失ってしまいます。

`aptitude` はフルスクリーンモードで通常小文字の 1文字の キーコマンドを受け付けます。 言及しておくべきキーストロークをいくつか 示します。

キーストローク	アクション
F10	メニュ
?	キーストロークヘルプを表示
u	パッケージアーカイブ情報を更新
+	パッケージをアップグレードするか新たにインストールするとしてマークする
-	パッケージを削除するとしてマークする (設定はそのまま)
_	パッケージを purge するとしてマークする (設定も削除)
=	パッケージを hold する
U	全てのアップグレード可能なパッケージをアップグレードするとしてマークする
g	選択パッケージのダウンロードおよびインストール

`q`	現在のスクリーンを終了し、変更点を保存する
`x`	現在のスクリーンを終了し、変更点を無視する
`Enter`	パッケージの情報を閲覧
`C`	パッケージの変更履歴を閲覧
`l`	表示されたパッケージの制限を変更
`/`	単語検索
`\`	最後の検索を繰り返す

`apt-get` のように、`aptitude` は選択したパッケージが Depends するパッケージをインストールできます。`aptitude` も インストールされるパッケージが Recommends したり Suggests する パッケージを引っ張って来るようなオプションを提供します。 メニュで `F10 -> Options -> Dependency handling` を選択することにより、 標準の挙動を変更できます。

`aptitude` の他の利点には次があります。

- `aptitude` はパッケージの全バージョンにアクセスできます。
- `aptitude` は `/var/log/aptitude` に行動を全て記録しています。
- `aptitude` は “Obsolete and Locally Created Packages” の下に リストすることにより、時代遅れのソフトウェアを追跡するのを簡単にします。
- `aptitude` には特定のパッケージを検索したりパッケージ表示を 制限するためのかなり力強いシステムが含まれます。`mutt` に親しんだ ユーザは、mutt から expression syntax の着想を取り込んだので、すぐに使える ようになるでしょう。 `/usr/share/doc/aptitude/README` の “SEARCHING, LIMITING, AND EXPRESSIONS” をご覧下さい。
- `aptitude` はフルスクリーンモードでは `su` の機能が 組み込んであり、本当に管理者権限が必要となるまでは通常のユーザから起動 できます。

6.2.3 `dselect`

Potato を含む stable リリースまでは、`dselect` は主要なパッケージ管理ツールでした。Sarge では、`aptitude` を代わりに使うことを考慮すべきです。

`dselect` を起動すると、自動的に “Required”, “Important”, および “Standard” パッケージの全てを選択します。

`dselect` はいくらか奇妙なユーザインターフェースを持ちます。 ほとんどの人は慣れますが、4 個のコマンドがあります。 (大文字は大文字で入力 することを示します!)

キーストローク	アクション
`Q`	終了。現在の選択を認め、とにかく終了する。 (依存性を上書きする)
`R`	戻る! 悪気はありません。
`D`	しまった! `dselect` が考えていることは分かりません。
`U`	全てを推奨される状態にセットする。

`D` や `Q` により、リスクを負って衝突した選択を 選ぶことができます。これらのコマンドを注意深く扱ってください。

雑音を減らすために、“expert” オプションを含む行を `/etc/dpkg/dselect.cfg` に追加してください。

あなたのマシンでは `dselect` の動きが遅い場合、インストールしたいパッケージを決定するために他の (高速な) マシンで `dselect` を起動して、実際にインストールするためには低速なマシンで `apt-get install` を使うことを考えた方が良いかもしれません。

6.2.4 APT を用いてディストリビューションを追いかける

`testing` ディストリビューションの変更を追いかけることには、セキュリティ修正を含むパッケージのインストールが遅れるという副作用がありうることに注意してください。そのようなパッケージはまず `unstable` にアップロードされ、遅れて `testing` に移動してきます。

例えば、`unstable` から選択したパッケージをインストールしながら `testing` を追いかけることを許可するようなより複雑な例については `apt_preferences(5)` をご覧ください。

他のパッケージを追いかけながら特定のバージョンであるパッケージをロックする例については、examples subdirectory (`http://www.debian.org/doc/manuals/debian-reference/examples/`) に `preferences.testing` や `preferences.unstable` として得られます。

ディストリビューションを混在させる、すなわち `stable` と `testing` を一緒にしたり、`stable` と `unstable` を一緒にすると、結果的には `testing` や `unstable` から `libc6` のような主要なパッケージを引っ張って来てしまうでしょう。これらにはバグが含まれていないと保証できません。警告しておきます。

もう一つの例として、`preferences.stable` は全てのパッケージの `stable` へのダウングレードを強制します。

新しいリリースの **package** を古いリリースのものにダウングレードすることは Debian では公式にサポートされていません。しかしながら、新しいバージョンがうまく動かないので、うまく動くバージョンのパッケージを再インストールする目的で特定のパッケージをダウングレードする必要があることが判明するかもしれません。これらの以前インストールしていたパッケージファイルはローカルの `/var/cache/apt/archives/` 又はリモートでは `http://snapshot.debian.net/` に見つかるかもしれません。‘`dpkg` を用いたレスキュー’ on page 86 もご覧下さい。

新しいリリースの **ディストリビューション** を古いリリースのものにダウングレードすることも Debian では公式にサポートされていませんし、問題を引き起こすでしょう。しかしながら、絶望に駆られた場合に最後の手段として試す価値はあるかもしれません。

6.2.5 `aptitude`, `apt-get` と `apt-cache` コマンド

上の例に記述されているように、`testing` を追いかける間、次のコマンドを用いてシステムを管理できます。

- `aptitude update` (又は `apt-get update`)

この作業により、リポジトリで利用可能なパッケージリストが更新されます。

- `aptitude upgrade` (又は `apt-get upgrade` 又は `aptitude dist-upgrade` 又は `apt-get dist-upgrade`)

  これらのコマンドにより、`testing` ディストリビューションを追いかけます — システム上のパッケージをそれぞれアップグレードし、その後インストールパッケージが依存するパッケージを `testing` ディストリビューションからのバージョンをインストールします。[2]

- `apt-get dselect-upgrade`

  これは `testing` ディストリビューションを追いかけます。— システムの各パッケージを `dselect` の選択に従って更新します。

- `aptitude install` *`package`*`/unstable`

  `unstable` ディストリビューションから *package* をインストールし、`testing` ディストリビューションから 依存するパッケージをインストールします。

- `aptitude install -t unstable` *`package`*

  `unstable` の Pin-Priority を 990 にセットすることにより、`unstable` ディストリビューションから *package* をインストールし、`unstable` ディストリビューションからこのパッケージに依存するパッケージをインストールします。

- `apt-cache policy` *`foo bar ...`*

  これは *foo bar ...* パッケージのステータスをチェックします。

- `aptitude show` *`foo bar ...`* `| less` (又は `apt-cache show` *`foo bar ...`* `| less`)

  パッケージ *foo bar ...* に関する情報をチェックします。

- `aptitude install` *`foo=2.2.4-1`*

  foo パッケージの特定のバージョン *2.2.4-1* をインストールします。

- `aptitude install` *`foo bar-`*

  foo パッケージをインストールし、*bar* パッケージを 削除します。

- `aptitude remove` *`bar`*

  bar パッケージを削除しますが、設定ファイルは削除しません。

- `aptitude purge` *`bar`*

  全ての設定ファイルと一緒に *bar* パッケージを削除します。

---

[2] `upgrade` と `dist-upgrade` の違いは、新しいバージョンと 古いバージョンのパッケージに依存関係の違いが見られる場合にのみ現れます。 詳細は `apt-get(8)` をご覧下さい。 `aptitude upgrade` と `aptitude dist-upgrade` は コマンドラインモードで `aptitude` を起動します。 `e` キーを押してこれらをフルスクリーンモードに切替えられます。

上の例では、`apt-get` に `-u` オプションを与えると アップグレードされる全てのパッケージのリストを表示し、次に取る行動を ユーザに促します。 次の例は `apt-get` が常にこの動作を取るようにします。 `aptitude` は、この動作をデフォルトで実施します。

```
$ cat >> /etc/apt/apt.conf << .
// Always show packages to be upgraded (-u)
APT::Get::Show-Upgraded "true";
.
```

実際にインストール、削除などをあらゆるパッケージに対して行わずに 同様の行動を取るには、`--no-act` を使用してください。

6.3 Debian で生き残るためのコマンド

本章の知識により、永遠の upgrade 生活をすごせます。 :)

6.3.1 Debian のバグをチェックし、助けを求める

特定のパッケージに関する問題に直面している場合、助けを求めたり、 バグレポートを出す前にこれらのサイトをチェックしましょう。 (`lynx`, `links`, および `w3m` は 同じように機能します)。

```
   $ lynx http://bugs.debian.org/
   $ lynx http://bugs.debian.org/package-name  # パッケージ名を知っている
場合
   $ lynx http://bugs.debian.org/bugnumber     # バグ番号を知っている場
合
```

“site:debian.org” を含む検索語により Google (www.google.com) を検索して みてください。

疑問がある場合は、良質のマニュアルを読んでください。`CDPATH` を 次のように設定してください。

```
export CDPATH=.:/usr/local:/usr/share/doc
```

そして次を実行してください。

```
$ cd packagename
$ pager README.Debian # 存在する場合
$ mc
```

さらなるサポートリソースを ‘Debian のサポート’ on page 247 に挙げます。

6.3.2 APT アップグレードのトラブルシュート

パッケージの依存性問題は‘アップグレード’ on page 76 に記述したように `unstable` 又は `testing` にアップグレードする場合に 発生する可能性があります。 ほとんどの場合、これはそのパッケージがまだ得られないパッケージに Depends しているためです。 これらの問題は次の手順を用いて解決できます。

```
# aptitude dist-upgrade
```

これが動かない場合、問題が解決するまで次のコマンドを繰り返し実行して ください。

```
# aptitude -f upgrade          # エラーが起きても upgradeを続ける
... 又は
# aptitude -f dist-upgrade     # エラーが起きても dist-upgrade を続ける
```

時々本当に壊れたアップグレードスクリプトにより持続的な問題を起こすことがあります。この種の状況を解決するには、`/var/lib/dpkg/info/`*`packagename`*`.{post,pre}{inst,rm}` スクリプトを調べ、次を実行するのがよいでしょう。

```
# dpkg --configure -a     # 部分的にインストールされたパッケージを全て設定
```

スクリプトに設定ファイルが無いと言っている場合、対応する設定ファイルに対して `/etc/` を調べてください。`.new` (又は同種の) 拡張子 を持つファイルが存在する場合、`mv` して拡張子を削除してください。

パッケージの依存性問題は `unstable` 又は `testing` に インストールする場合に発生する場合があります。依存性を迂回する手段があります。

```
# aptitude -f  install package # 壊れた依存性を上書きする
```

これらの状況を修正するための代替手段として、`equivs` パッケージを使えます。 `/usr/share/doc/equivs/README.Debian` と ‘equivs パッケージ’ on page 96 をご覧ください。

6.3.3 dpkg を用いたレスキュー

APT を用いて行き詰まった場合、Debian のミラーからパッケージを ダウンロードし、`dpkg` を用いてインストールできます。 ネットワークにアクセスできない場合、`/var/cache/apt/archives/` にあるパッケージファイルのキャッシュを探せます。

```
# dpkg -i fetchmail_6.2.5-4_i386.deb
```

依存性の衝突によりこのようにパッケージをインストールするのを失敗してしまい、 本当のそのパッケージをインストールする必要がある場合、 `dpkg` の `--ignore-depends`, `--force-depends`, や他のオプションを用いて依存性のチェックを上書きすることができます。 詳細は `dpkg(8)` をご覧下さい。

6.3.4 パッケージ選択データの回復

`/var/lib/dpkg/status` がなんらかの理由で壊れた場合、Debian システムはパッケージ選択データを失い、ひどく苦しみます。 `/var/lib/dpkg/status-old` や `/var/backups/dpkg.status.*` にある古い `/var/lib/dpkg/status` ファイルを探してください。

このディレクトリが多くの重要なシステムデータを含んでいるので、 別のパーティションに `/var/backups/` を保持するのは良い考えです。

古い `/var/lib/dpkg/status` ファイルが得られない場合、まだ `/usr/share/doc/` にあるディレクトリからの情報で回復できます。

```
# ls /usr/share/doc | \
  grep -v [A-Z] | \
  grep -v ’^texmf$’ | \
  grep -v ’^debian$’ | \
  awk ’{print $1 " install"}’ | \
  dpkg --set-selections
# dselect --expert # システムを再インストールし、必要ない物を除外する
```

6.3.5 **/var** のクラッシュ後のシステム回復

`/var` ディレクトリはメールなどの定期的に更新されるデータを 含むので、汚染されやすくなっています。このディレクトリを別のパーティションに 分けることにより、リスクを限定できます。故障が発生した場合、 `/var` ディレクトリを再構築して Debian システムを回復する必要があるかもしれません。

最小限 Debian が機能する `/var` ディレクトリの基幹部分を同一又は古い Debian バージョンから、例えば `var.tar.gz` (`http://people.debian.org/~osamu/pub/`) などを取得し、壊れたシステムの root ディレクトリに置きます。そして

```
# cd /
# mv var var-old       # 役に立つ内容が残っている場合
# tar xvzf var.tar.gz # Woody の基幹ファイルを使用
# aptitude             # 又は dselect
```

を実行します。これによりシステムが機能するようになるはずです。'パッケージ選択データの回復' on page 87 に記述している技術を用いてパッケージ選択データ の回復をはかどらせることができます。 ([FIXME]: 本手順はさらなる検証が必要。)

6.3.6 ブート不能なシステムにパッケージをインストール

Debian レスキューフロッピ/CD 又は マルチブート Linux システム上の別の パーティションを用いて Linux をブートしてください。'システムをブートする' on page 103 参照。 ブート不可能なシステムを `/target` にマウントし、`dpkg` の chroot インストールモードを使用します。

```
# dpkg --root /target -i packagefile.deb
```

そして設定を行い、問題を修正します。

ところで、`lilo` が壊れてブート不能になった場合、標準の Debian レスキューディスクを用いてブートできます。 Linux をインストールしてある root パーティションを `/dev/hda12` と仮定し、ランレベル 3 で起動したいとすると、lilo のプロンプトで次を 入力してください。

```
boot: rescue root=/dev/hda12 3
```

こうしてフロッピディスク上の kernel を用いてほぼ完全に機能するシステムに ブートできます。(kernel の機能やモジュールがないことによる些細な不都合が 存在するかもしれません。)

6.3.7 dpkg コマンドが壊れた場合どうするか

`dpkg` が壊れると `.deb` ファイルがインストール不能になります。 次の手順によりこのような状況からの回復の助けになります。(第 1 行の "links" を好みのブラウザコマンドに置き換えてください。)

```
$ links http://http.us.debian.org/debian/pool/main/d/dpkg/
  ... 良好な dpkg_version_arch.deb をダウンロード
$ su
password: *****
# ar x dpkg_version_arch.deb
# mv data.tar.gz /data.tar.gz
# cd /
# tar xzfv data.tar.gz
```

`i386` に対しては、`http://packages.debian.org/dpkg` が URL としても使われます。

6.4 Debian 涅槃コマンド

これらのコマンドを **愉しむ** と、永遠のアップグレード地獄から救い出し、Debian の **涅槃** に導くことができます。:)

6.4.1 ファイルに関する情報

ある特定のファイル名のパターンがどのインストール済みパッケージに 所属するかを見つけるには以下を実行します：

```
$ dpkg {-S|--search} pattern
```

もしくは同様のことを Debian アーカイブ中で見付けるには以下を実行します：

```
$ wget http://ftp.us.debian.org/debian/dists/sarge/Contents-i386.gz
$ zgrep -e pattern Contents-i386.gz
```

もしくは特別なパッケージコマンドを使います。

```
# aptitude install dlocate
$ dlocate filename         # dpkg -L と dpkg -S の高速な代替品
...
# aptitude install auto-apt # オンデマンドのパッケージインストールツール
# auto-apt update          # autp-apt 用の db ファイルを作成
$ auto-apt search pattern
# インストールされているかに係わらず、全パッケージをパターン検索
```

6.4.2 パッケージ情報

パッケージアーカイブから情報を検索し、表示します。`/etc/apt/sources.list` を編集して APT が適切なアーカイブを 指すようにしてください。`testing` 又は `unstable` にあるパッケージが現在 インストール しているパッケージに対してどうなっているかを知るには、`apt-cache policy` を使うのが良いでしょう。

```
# apt-get   check       # キャッシュを更新し、壊れたパッケージをチェック
$ apt-cache search  pattern # テキストの説明からパッケージ検索
$ apt-cache policy  package # パッケージの priority/dist 情報を表示
$ apt-cache show -a package # 全 dists のパッケージ説明を表示
$ apt-cache showsrc package # マッチしたソースパッケージの説明を表示
$ apt-cache showpkg package # パッケージのデバッグ情報を表示
# dpkg  --audit|-C          # 部分的にインストールされたパッケージを検索
$ dpkg {-s|--status} package ...  # インストール済みのパッケージの説明を
表示
$ dpkg -l package ...     # インストール済みパッケージのステータスを表
示 (1行毎)
$ dpkg -L package ...     # 指定したパッケージに含まれるファイル名リストを
表示
```

`apt-cache showrc` は Woody リリースではドキュメント化されていませんが、使えます :)

また、次に挙げるファイルからもパッケージ情報を検索できます(これらを見るのに `mc` を使っています)。

```
/var/lib/apt/lists/*
/var/lib/dpkg/available
```

次のファイルを比較すると、最後のインストールセッションで何が起きたかが正確に分かります。

```
/var/lib/dpkg/status
/var/backups/dpkg.status*
```

6.4.3 APT によりキーボードに触らずにインストール

キーボードに触らずにインストールするには、次の行を `/etc/apt/apt.conf` に追加してください。

```
Dpkg::Options {"--force-confold";}
```

これは `aptitude -y install` *`packagename`* を起動するのと同じことです。これは全プロンプトについて自動的に “yes” で答えるので、問題が発生するかもしれません。ゆえに慎重にこのトリックを使ってください。`apt.conf(5)` および `dpkg(1)` をご覧ください。

‘インストール済みパッケージの再設定’ on page 90 に従い、特定のパッケージを後で設定することもできます。

6.4.4 インストール済みパッケージの再設定

次のコマンドを使ってインストール済みパッケージの再設定を行います。

```
# dpkg-reconfigure --priority=medium package [...]
# dpkg-reconfigure --all   # 全パッケージの再設定
# dpkg-reconfigure locales # 特別なロケールの生成
# dpkg-reconfigure --p=low xserver-xfree86 # X サーバの再設定
```

`debconf` ダイアログモードが永続的に必要な場合は、`debconf` を再設定してください。

特別な設定スクリプトを持つプログラムがいくつかあります。[3]

```
apt-setup     - /etc/apt/sources.list の生成
install-mbr   - Master Boot Record manager のインストール
tzconfig      - ローカル time zoneゾーンの設定
gpmconfig     - gpm マウスデーモンの設定
eximconfig    - Exim (MTA) の設定
texconfig     - teTeX の設定
apacheconfig  - Apache (httpd) の設定
cvsconfig     - CVS の設定
```

---

[3] いくつかの `*config` スクリプトは新しい Sarge リリースでは消えており、パッケージの設定機能は `debconf` に移動しています。

```
sndconfig     - サウンドシステムの設定
...
update-alternatives - 標準のコマンドの設定、例えば vim を vi に設定
update-rc.d         - System-V init スクリプトマネージャ
update-menus        - Debian menu システム
...
```

6.4.5 パッケージの削除及びパージ

設定ファイルを維持したままパッケージを削除します。

```
# aptitude remove package ...
# dpkg  --remove package ...
```

設定ファイルを含め、パッケージを削除します。

```
# aptitude purge package ...
# dpkg    --purge        package ...
```

6.4.6 古いパッケージを hold する

例えば、`libc6` と `libc6-dev` を `dselect` および `aptitude install` *`package`* に対して hold するには、次を実行します。

```
# echo -e "libc6 hold\nlibc6-dev hold" | dpkg --set-selections
```

`aptitude install` *`package`* はこの "hold" により隠されません。 `aptitude upgrade` *`package`* 又は `aptitude dist-upgrade` に対する自動ダウングレードの強行から パッケージを hold するには、 `/etc/apt/preferences` に次の行を追加してください。

```
Package: libc6
Pin: release a=stable
Pin-Priority: 2000
```

ここで、"`Package:`" エントリは "`libc6*`" のようなエントリを 使えません。`glibc` ソースパッケージに同期したバージョンの 全バイナリパッケージを hold する必要がある場合、それらのパッケージを明示的に 挙げる必要があります。

次のコマンドにより hold されたパッケージのリストを表示できます。

```
dpkg --get-selections "*"|grep -e "hold$"
```

6.4.7 **`stable/testing/unstable`** システムの混在

`apt-show-versions` により、特定のディストリビューションによる パッケージのバージョンをリストできます。

```
$ apt-show-versions | fgrep /testing | wc
... testing からのパッケージ数をカウント
$ apt-show-versions -u
... アップグレード可能なパッケージ数
$ aptitude install `apt-show-versions -u -b | fgrep /unstable`
... 全ての unstable パッケージを最新バージョンにアップグレード
```

6.4.8 キャッシュされたパッケージファイルを取り除く

APT でパッケージをインストールすると、キャッシュされたパッケージファイルが `/var/cache/apt/archives` に残されるので、これらを消す必要が あります。

```
# apt-get autoclean # 必要ないパッケージファイルのみ削除
# apt-get clean     # キャッシュされたパッケージファイル全てを削除
```

6.4.9 システム設定の記録/コピー

パッケージ選択ステータスのローカルコピーを取るには、次を実行します。

```
# debconf-get-selections > debconfsel.txt
# dpkg --get-selections "*" >myselections   # 又は \* を使用
```

“`*`” により、*`myselections`* が “purge” 用の パッケージエントリにも含まれるようになります。

このファイルを他のコンピュータに転送し、これを用いてインストール可能です。

```
# dselect update
# debconf-set-selections < debconfsel.txt
# dpkg --set-selections <myselections
# apt-get -u dselect-upgrade    # 又は dselect install
```

6.4.10 **`stable`** システムへのパッケージ移植

`stable` システムの部分的なアップグレードのためには、ソースパッケージを 用いて `stable` 環境でパッケージを再構築するのが望ましいです。 パッケージ再構築により、依存性による強引なパッケージアップグレードを避けることが できます。まず、次のエントリを `/etc/apt/sources.list` に追加します。

```
deb-src http://http.us.debian.org/debian testing \
 main contrib non-free
deb-src http://http.us.debian.org/debian unstable \
 main contrib non-free
```

ここで、`deb-src` に対する各エントリは印刷時の制限のために 2行に分割 されていますが、実際の `sources.list` は 1行としてください。

そしてソースを取得し、ローカルパッケージを作成します。

```
$ apt-get update  # ソースパッケージの検索リストを更新
$ apt-get source package
$ dpkg-source -x package.dsc
$ cd package-version
  ... 要求されたパッケージを検査 (.dsc ファイル中の Build-depends) し、
      それらもインストールする。"fakeroot" パッケージも必要。

$ dpkg-buildpackage -rfakeroot

  ...又は (サイン無し)
$ dpkg-buildpackage -rfakeroot -us -uc #必要ならば後で "debsign" を使
用

  ...そしてインストール
$ su -c "dpkg -i packagefile.deb"
```

普通、パッケージの依存性を満たすために "-dev" サフィックスが付く 2, 3個の パッケージをインストールする必要があります。`debsign` は `devscripts` パッケージにあります。`auto-apt` を使うと、依存性を簡単に満足させられるかもしれません。`fakeroot` を使うと、root アカウントの不必要な使用を避けられます。

Woody では、依存性の問題は単純にできます。例えば、ソースのみの `pine` パッケージをコンパイルするには、

```
# apt-get build-dep pine
# apt-get source -b pine
```

だけです。

6.4.11 ローカルパッケージのアーカイブ

APT と `dselect` システムとの互換性を持つローカルパッケージの アーカイブを作成するには、`Packages` を作成し、 特定のディレクトリツリーに置く必要があります。

公式の Debian アーカイブと同様のローカル `deb` リポジトリを次のように作成できます。

```
# aptitude install dpkg-dev
# cd /usr/local
# install -d pool # 真のパッケージはここに置きます
# install -d dists/unstable/main/binary-i386
# ls -1 pool | sed 's/_.*$/ priority section/' | uniq > override
# editor override # priority と section を調整
# dpkg-scanpackages pool override /usr/local/ \
   > dists/unstable/main/binary-i386/Packages
# cat > dists/unstable/main/Release << EOF
Archive: unstable
Version: 3.0
Component: main
Origin: Local
Label: Local
Architecture: i386
EOF
# echo "deb file:/usr/local unstable main" \
   >> /etc/apt/sources.list
```

代わりに、手早いけど汚いローカル deb リポジトリを次のように作れます。

```
# aptitude install dpkg-dev
# mkdir /usr/local/debian
# mv /some/where/package.deb /usr/local/debian
# dpkg-scanpackages /usr/local/debian /dev/null | \
  gzip - > /usr/local/debian/Packages.gz
#  echo "deb file:/usr/local/debian ./" >> /etc/apt/sources.list
```

HTTP や FTP メソッドによりディレクトリへのアクセス手段を供給し、`/etc/apt/sources.list` にエントリを追加することにより、これらのアーカイブへのリモートからのアクセスを可能にします。

6.4.12 alien バイナリパッケージへの変換又はインストール

`alien` は Red Hat `rpm` 形式、Stampede `slp` 形式、Slackware `tgz` 形式、そして Solaris `pkg` 形式で供給される バイナリパッケージを Debian `deb` パッケージ形式に変換することを 可能とします。他の Linux ディストリビューションからのパッケージを システムにインストールしているディストリビューションで使いたい場合、`alien` を使って使用中のディストリビューションのパッケージフォーマットに変換し、インストールできます。`alien` は LSB パッケージもサポートします。

6.4.13 自動でコマンドをインストールする

`auto-apt` はオンデマンドのパッケージ インストール ツールです。

```
$ sudo auto-apt update
 ... データベースを更新
$ auto-apt -x -y run
Entering auto-apt mode: /bin/bash
Exit the command to leave auto-apt mode.
$ less /usr/share/doc/med-bio/copyright # 存在しないファイルにアクセスする
 ...  このファイルを供給するパッケージをインストールする
 ... 依存するパッケージもインストールする
```

6.4.14 インストールされたパッケージファイルを検証する

`debsums` は MD5 チェックサムを用いてインストールされたパッケージファイルの検証ができます。いくつかのパッケージは MD5 チェックサムを得られません。システム管理者が可能な一時的な修正はこのようなものです。

```
# cat >>/etc/apt/apt.conf.d/90debsums
DPkg::Post-Install-Pkgs {"xargs /usr/bin/debsums -sg";};
^D
```

Joerg Wendland <`joergland@debian.org`> からのメールによる (未検証)。

6.4.15 最適化された `sources.list`

簡単に言うと、最適化された `sources.list` を作成するための手の込んだ努力は USA に住んでいる私にとってはあまり効果を生みませんでした。私は `apt-setup` を使って近くのサイトを手動で選びました。

`apt-spy` はレイテンシとバンド幅に基づいて自動で `sources.list` を作成します。`netselect-apt` はより完全な `sources.list` を作成しますが、最適なミラーサイトを選択する、より優れた方法 (ping 時間の比較) を使用しています。

```
# aptitude install apt-spy
# cd /etc/apt ; mv sources.list sources.list.org
# apt-spy -d testing -l sources.apt
```

6.5 他の Debian 独特な点

6.5.1 `dpkg-divert` コマンド

Diversions というファイルにより、`dpkg` は ファイルをインストールする時に本来意図していた場所ではなく、**退避した** 場所にインストールするようになります。**Diversions** は

Debian パッケージスクリプトで衝突が起こりうる ファイルを移動させるために使うことができます。システム管理者はパッケージ の設定ファイルや他のファイルを (**conffiles** としてマーク されていなければ) `dpkg` が新しいバージョンのパッケージを インストールする時にそれらのファイルを上書きしてしまわないようにするために diversion を使うことができます。 ('ローカル設定の保存' on page 13) 参照.

```
# dpkg-divert [--add]  filename # "diversion" を追加
# dpkg-divert --remove filename # "diversion" を削除
```

本当に必要でない限り、`dpkg-divert` を使わない方が 通常は良いでしょう。

6.5.2 **`equivs`** パッケージ

ソースからプログラムをコンパイルした場合、最も良いのは本当のローカルな debian 化したパッケージ (`.deb`) にすることです。 `equivs` を最後の手段として使います。

```
Package: equivs
Priority: extra
Section: admin
Description: Debian パッケージの依存関係を偽るためのパッケージ
 これはダミーパッケージで、依存情報だけを含んだ Debian パッケージの
 作成に使用することができます。
```

6.5.3 Alternative コマンド

`vi` コマンドが `vim` を起動するようにするには、 `update-alternatives` を使います。

```
# update-alternatives --display vi
...
# update-alternatives --config vi
  Selection    Command
-----------------------------------------------
      1        /usr/bin/elvis-tiny
      2        /usr/bin/vim
*+    3        /usr/bin/nvi

default[*] を保つために Enter を打つか、selection 番号 2 をタイプしましょう。
```

Debian alternatives システムのアイテムは `/etc/alternatives/` に シンボリックリンクとして保持されています。

好みの X Window 環境を設定するには、`update-alternatives` を `/usr/bin/x-session-manager` と `/usr/bin/x-window-manager` に適用します。詳細は、'X セッションのカスタマイズ' on page 147 を参照。

`/bin/sh` は `/bin/bash` 又は `/bin/dash` の直接のシンボリックリンクです。古い Bashism で汚染されたスクリプトとの互換性のため、`/bin/bash` を使う方が安全ですが、POSIX 互換性を強制するには、`/bin/dash` を使うのがより良い訓練となります。2.4 Linux kernel にアップグレードすると、`/bin/sh` を `/bin/dash` にセットしがちです。

6.5.4 ランレベルの使い方

インストール直後は、ほとんどの Debian パッケージはサービスを ランレベル 2 から 5 まで起動するように設定します。これゆえに、カスタマイズされていない Debian システムではランレベル 2, 3, 4, と 5 には何の違いがありません。Debian はローカルの管理者に任せています。'ランレベルのカスタマイズ' on page 22 に記述しているようにランレベルをカスタマイズすることを ローカルの管理者に任せています。これは他の有名な GNU/Linux ディストリビューションにより用いられている ランレベルの方法とは異なります。ブートシーケンスの終わりで X ディスプレイマネージャが起動されないように ランレベル 2 で `xdm` や `gdm` を無効にしたい時が 来るかもしれません。その時はランレベル 3 から起動するように変更できます。ランレベルについてのより詳しい情報は 'ランレベル' on page 21 をご覧下さい。

6.5.5 デーモンサービスを無効にする

Debian 開発者はシステムのセキュリティを深刻にとらえています。多くのデーモンサービスは最小のサービスと機能を有効にして インストールされます。

デーモンサービス (Exim, DHCP など) に疑いを持った場合、`ps aux` を起動するか、`/etc/init.d/*` や `/etc/inetd.conf` の内容を調べましょう。また、`/etc/hosts.deny` ('PAM を用いたログイン制限' on page 133 参照)も調べましょう。`pidof` コマンドも役立ちます。(`pidof(8)` 参照)

最近の Debian において、標準では X11 は TCP/IP 経由の遠隔接続を 許可していません。'TCP/IP 越しに X を使う' on page 150 参照 SSH での X フォワードも無効にされています。'リモートの X サーバに接続する – `ssh`' on page 151 参照

章 7

Debian での Linux kernel

Debian には独自の kernel と関連モジュールの再構築方法があります。 'Debian と kernel' on page 23 をご覧ください。

7.1 Kernel (再)構築

最新の Linux kernel をコンパイルする場合、Debian `unstable` から、 `gcc`、`binituls` と `module-init-tools` をインストールして使用すると役に立つかもしれません。 公式の情報を得るには、`/usr/share/doc/kernel-package/README.gz`、特に最後尾を ご覧ください。

常に変更が発生しているので、kernel のコンパイルは最も称賛される開発者 でさえも混乱を招きかねない難しい事項です。

Manoj Srivastava の発言 `--initrd` は Debian のみ cramfs パッチを必要とする。

Herbert Xu の発言 間違っている。CRAMFS 以外のファイルシステムを使うならば `/etc/mkinitrd/mkinitrd.conf` に MKIMAGE を必ず設定する必要がある。

注意して、常に Manoj と Kent による `/usr/share/doc/kernel-package/README.gz` や `http://kernel-handbook.alioth.debian.org/` に頼ってください。 最新版の kernel をコンパイルしようとしている場合、 最新の unstable 版の `kernel-package` パッケージを 確実にインストールしてください。

1 台のマシンのためにしか kernel をコンパイルする必要がないなら、`initrd` は必要ありません。kernel-image パッケージにより供給される kernel とほとんど同じにしたいため、私は initrd を用います。 もし `initrd` を使用するなら、`initrd(8)` と `mkinitrd.conf(5)` をご覧ください。

7.1.1 古典的な方法

オリジナルソースを以下から取得してください。

- Linux: `http://www.kernel.org/`

- pcmcia-cs: `http://pcmcia-cs.sourceforge.net/`

又は Debian での 同等の kernel source を用いて次の手順を実行してください。

```
# cd /usr/src
# tar xfvz linux-whatever.tar.gz
# rm -rf linux
# ln -s linux-whatever linux
# tar xfvz pcmcia-cs-whatever.tar.gz
# ln -s pcmcia-cs-whatever pcmcia
# cd linux
# make menuconfig
... 設定を行う
# make dep
# make bzImage
... lilo / grub の設定 ...
... /usr/src/linux/arch/i386/boot/bzImage を boot に移動...
... /sbin/lilo 又は grub のための好みの方法を実行
# make modules; make modules_install
# cd ../pcmcia
# make config
# make all
# make install
... 必要なモジュール名を /etc/modules に追加
# shutdown -r now
... 新しい kernel を boot
```

7.1.2 Kernel ヘッダ

ほとんどの “normal” なプログラムには kernel ヘッダは必要なく、実際直接 使用した場合動かなくなるかもしれません。その代わり、**`glibc` を構築するのに使った** ヘッダに対してコンパイルすべきです。それらは Debian システムの `/usr/include/linux` と `/usr/include/asm` にあるバージョンのものです。

それゆえ、いくつかの古びたドキュメントが推奨しているように `/usr/include/linux` および `/usr/include/asm` から `/usr/src/linux` にあるディレクトリに symlink を張らないでください。

カーネル特有のアプリケーションプログラムに特定の kernel ヘッダが 必要ならば、include path が *`dir-of-particular-kernel-headers`*`/include/linux` と *`dir-of-particular-kernel-headers`*`/include/asm` を 指すように makefile を変更してください。

7.2 proc filesystem により Linux kernel をチューニングする

Linux kernel の挙動は、proc ファイルシステムを用いるとオンザフライで変更できます。

`/proc` ファイルシステムを通して kernel パラメータを変更することに関する基本的な情報は Linux source の `Documentation/sysctl/*` をご覧ください。

`/etc/init.d/networking` と 'いくつかのウェブサイトにアクセスした時の奇妙な挙動' on page 45 には、kernel パラメータの操作方法の例がいくつかあります。

通常、`/etc/rcS.d/S30procps.sh` から起動される `/etc/init.d/procps.sh` を用いて `/proc` ファイルシステム経由でブート時に kernel を設定する手順については、`sysctl.conf(5)` をご覧ください。

7.2.1 Too many open files

Linux kernel が "Too many open files" とエラーを出力することがあります。これは `file-max` の標準値が小さい (8096) ためです。この問題を解決するには、root で次のコマンドを実行してください。

```
   # echo "65536"  > /proc/sys/fs/file-max  # 2.2 と 2.4 kernel の両方
で有効
   # echo "131072" > /proc/sys/fs/inode-max # 2.2 kernel のみ
```

変更をずっと有効にしたい場合は、`/etc/sysctl.conf` に次の行を追加してください。

```
   file-max=65536   # 2.2 と 2.4 kernel の両方で有効
   inode-max=131072 # 2.2 kernel のみ
```

7.2.2 Disk flush 間隔

proc ファイルシステムを通じて disk flush 間隔を変更できます。次のコマンドにより、間隔をデフォルトの 5 秒 から 1 秒 に変更されます。

```
   # echo "40 0 0 0 100 30000 60 0 0"  > /proc/sys/vm/bdflush
```

本変更はファイル I/O 性能にほんの少し悪影響を与えるかもしれません。しかし、本変更により、デフォルトの 5 秒より短い最後の 1 秒の間を除き、ファイル内容を 安全に保たれます。

7.2.3 鈍くて古い少メモリ容量マシン

古い少メモリ容量マシンの場合、proc ファイルシステムを通じて メモリの over-commit を有効にするとまだ役に立つかもしれません。

```
   # echo 1 > /proc/sys/vm/overcommit_memory
```

7.3 2.6 カーネルと udev

udev は `/dev/` のダイナミックな置き換えです。 デバイス名は非常に短い物にもできます。 2.4 カーネルで使われた devfs は今や過去の物となりました。

`udev` パッケージを `kernel-image-2.6.`*`NN`* で提供される新たな Debian 2.6 カーネルとともにインストールすると、 本機能が実現します。

章 8

Debian tips

8.1 システムをブートする

ブートプロンプトの詳細は LDP にある BootPrompt-HOWTO (`http://www.tldp.org/HOWTO/BootPrompt-HOWTO.html`) をご覧ください。

8.1.1 “root パスワードを忘れてしまった!” (その 1)

コンソールのキーボードにアクセスできる人に限り、root パスワードを 知らずにシステムをブートして root アカウントにログオンできます。(システムをブートするのを妨げる BIOS や `lilo` などの ブートローダからパスワードを要求されないことを仮定しています。)

これは外部の boot disk や BIOS のブート設定の変更を要求しない手順 です。ここで、“Linux” は標準の Debian インストールで Linux kernel をブートするためのラベルです。

`lilo` ブートスクリーンでは、`boot:` が表示されたら すぐに enter を押し、次のように入力します。(自動ブートを避けるため、この時点で shift キーを押す必要があるシステム があります。また、`lilo` がフレームバッファを使っている場合、 タイプしたオプションを見るには TAB を押す必要があります。)

```
boot: Linux init=/bin/sh
```

この操作により、システムが kernel をブートし、標準の `init` の代わりに `/bin/sh` を起動します。今や root 権限と root シェル を獲得しました。`/` は現在読み込みのみでマウントされ、多くのディスクパーティションはまだマウントされていないので、十分機能するシステムとするには、次を実行する必要があります。

```
init-2.03# mount -n -o remount,rw /
init-2.03# mount -avt nonfs,noproc,nosmbfs
init-2.03# cd /etc
```

```
init-2.03# vi passwd
init-2.03# vi shadow
```

(`/etc/passwd` の 2 番目のデータフィールドが全てのユーザ名に対して “x” である場合、そのシステムはシャドウパスワードを使用しているため、`/etc/shadow` を編集する必要があります。) root パスワードを無効にするには、パスワードファイルの 2 番目のフィールド が空になるように編集してください。システムをリブートして、パスワード無しで root としてログインできるようになりました。runlevel 1 でブートすると、(少なくとも Potato 以降の) Debian はパスワードを要求しますが、より古い システムは要求しませんでした。

`/usr/` にアクセスできない場合に備えて `/bin/` に最小限のエディタを置いておくのは良い考えです。(*‘緊急用エディタ’* on page 208 参照)

また、`sash` パッケージのインストールも考慮に入れてください。 システムがブート不能に陥った場合、次を実行します。

```
boot: Linux init=/bin/sash
```

`sash` は `/bin/sh` が使えなくなった場合でも `sh` 対話的な代替品として動きます。 これはスタティックリンクされており、多くの標準的な機能を組込み機能として 含んでいます。 (リファレンスリストのため、プロンプトで “help” をタイプしてみてください)

8.1.2 “root パスワードを忘れてしまった!” (その 2)

rescue disk/root disk セットからブートします。 *`/dev/hda3`* がもともとの root パーティションの場合、 次の手順は、パスワードファイルを前述の手順と同じ程度に簡単に編集できるようにします。

```
# mkdir fixit
# mount /dev/hda3 fixit
# cd fixit/etc
# vi shadow
# vi passwd
```

この手法の前述した手法に対する利点は、`lilo` パスワードを知る (ある場合) 必要がないことです。しかし、本手法を使うには、 システムがフロッピ又は CD からブートできるように BIOS セットアップに アクセスできる必要があります。

8.1.3 boot 不能

インストール中に boot ディスクを作るのをさぼっていたとしても、問題ありません。`lilo` が壊れた場合、Debian インストールセットから boot disk を持ってきて、そこからシステムを boot させます。boot プロンプトでは、 root パーティション が *`/dev/hda12`* であり、 ランレベル 3 で起動したいと仮定すると、次のコマンドを入力します。

```
boot: rescue root=/dev/hda12 3
```

そして、フロッピにある kernel を使ってほとんどフル機能のシステムにブートします。(kernel の機能やモジュールの不足によるささいな誤作動はあります。)

システムが壊れた場合、'ブート不能なシステムにパッケージをインストール' on page 87 をご覧ください。

カスタムブートフロッピが必要な場合、rescue disk の `readme.txt` に従ってください。

8.1.4 "boot 時に X を無効にさせてくれ!"

`unstable` を追いかけることは楽しいですが、ブート中に起動するバギーな `xdm`、`gdm`、`kdm` あるいは `wdm` がひどく噛みつくとも限りません。

まず boot プロンプトで次を入力して root シェルを獲得します。

```
boot: Linux vga=normal s
```

ここで、*Linux* はブートしたい kernel のためのラベルです。"vga=normal" により、`lilo` が必ず通常の VGA モードで起動します。そして、"s" (又は "S") は `init` がシングルユーザモードを呼び出すために渡されるパラメータです。

X の起動デーモンを全て無効にするためのいくつかの方法があります。

- run `update-rc.d -f ?dm remove;update-rc.d?dm stop 99 1 2 3 4 5 6 .`
- insert "exit 0" at the start of all `/etc/init.d/?dm` files.
- 全ての `/etc/rc2.d/S99?dm` ファイルを `/etc/rc2.d/K99?dm` に改名する。
- 全ての `/etc/rc2.d/S99?dm` を削除する。
- `/etc/X11/default-display-manager` を起動する。

ここで、`rc2.d` の中の数字は `/etc/inittab` で指定したランレベルと対応させなければなりません。また、`?dm` は全ての `xdm`, `gdm`, `kdm`, `wdm` を置き換えるために複数回コマンドを実行する必要があることを意味します。

このリストの最初だけが Debian での "真実の道" です。最後の方法は簡単ですが、Debian でしか動かず、`dpkg-reconfigure` を使用した後は再度ディスプレイマネージャの設定を要求します。残りは汎用のデーモンの無効化手段です。

コンソールシェルから `startx` コマンドを使って X をまだ起動できます。

8.1.5 ブートプロンプトを使った他のブートトリック

`lilo` のブートプロンプトを使ってシステムを特定のランレベルと設定でブートすることができます。詳細は BootPrompt-HOWTO (`http://www.tldp.org/HOWTO/BootPrompt-HOWTO.html`) (LDP) にあります。

システムをランレベル 4 でブートさせたい場合、`lilo` の ブートプロンプトで次を入力してください。

```
boot: Linux 4
```

root パスワードを知っている状態でシステムを普通に動くシングルユーザモード で起動したい場合、次の例のうち 1 つを `lilo` のブートプロンプト で実行してください。

```
boot: Linux S
boot: Linux 1
boot: Linux -s
```

システムを実際よりも少ないメモリ容量を用いてブートしたい場合 (例えば 64MB のシステムで 48MB を用いる場合)、`lilo` ブートプロンプト で次を実行してください。

```
boot: Linux mem=48M
```

実際より多いメモリ容量をここで絶対指定しないでください。さもないと kernel がクラッシュします。64MB より多い、例えば 128MB のメモリを持っている場合、`mem=128M` をブートプロンプトで実行するか、同様の行を `/etc/lilo.conf` に追加しないと、古い kernel や古い BIOS を 持つマザーボードは 64MB 以上のメモリを使用できません。

8.1.6 GRUB ブートパラメータの設定

GRUB は GNU Hurd プロジェクト由来の新しいブートマネージャであり、 Lilo より柔軟性がありますが、ブートパラメータの取り扱い方が lilo とはずいぶん異なります。

```
grub> find /vmlinuz
grub> root (hd0,0)
grub> kernel /vmlinuz root=/dev/hda1
grub> initrd /initrd
grub> boot
```

ここで、Hurd デバイス名に気づく必要があります。

the Hurd/GRUB	Linux	MS-DOS/Windows
(fd0)	/dev/fd0	A:
(hd0,0)	/dev/hda1	C: (通常)
(hd0,3)	/dev/hda4	F: (通常)
(hd1,3)	/dev/hdb4	?

詳細は `/usr/share/doc/grub/README.Debian.gz` と `/usr/share/doc/grub-doc/html/` をご覧ください。

8.2 行動を記録する

8.2.1 シェルでの行動を記録する

システム管理者は Unix 環境において通常の PC 環境よりもより複雑な仕事 を行います。システムトラブルから回復する必要がある場合に備えて、 設定の最も基本的な意味を確実に知る必要があります。X11 ベースの GUI 設定ツールは見栄えは良く便利ですが、たびたび緊急事態に対応できません。

このような観点で、シェルでの活動を記録することは、特に root に とっては良い習慣です。

Emacs では、M-x `shell` を実行すると、バッファに記録を開始し、 C-x C-w を使うとバッファをファイルに書き出します。

Shell では、'`screen` を使って端末を切替える' on page 123 に記述されているように、`screen` コマンドを "^A H" 付きで実行するか、`script` コマンドを 使ってください。

```
$ script
スクリプトを開始しました、ファイルは typescript です
 ... 何かを行う ...
 Ctrl-D
$ col -bx <typescript >savefile
$ vi savefile
```

次は `script` の代わりに使えます。

```
$ bash -i 2>&1 | tee typescript
```

8.2.2 X での行動を記録する

`xterm` の表示を含む X アプリケーションのグラフィックな画像を記録する 必要がある場合、`gimp` (GUI) を使いましょう。ウィンドウごとあるいは スクリーン全体をキャプチャーできます。代替品には、`xwd` (`xbase-clients`), `import` (`imagemagick`), `scrot` (`scrot`), があります。

8.3 サブディレクトリ全体のコピーとアーカイブ

これらのコピーとアーカイブコマンドはシステムやデーターのバックアップの 基本となります。簡単なバックアップスクリプト例が　スクリプト例 (`http://www.debian.org/doc/manuals/debian-reference/examples/`)に　`backup` として納められています。

8.3.1 サブディレクトリ全体をコピーする基本的なコマンド

ファイル構成を再調整する必要がある場合、ファイルのリンクを含む内容を 次のように移動します。

```
標準的方法
# cp -a /source/directory /dest/directory # GNU cp が必要
# (cd /source/directory && tar cf - . ) | \
        (cd /dest/directory && tar xvfp - )
ハードリンクを含む場合、衒学的な方法が必要となります。
# cd /path/to/old/directory
# find . -depth -print0 | afio -p -xv -0a /mount/point/of/new/directory
リモートにコピーする。
# (cd /source/directory && tar cf - . ) | \
        ssh user@host.dom (cd /dest/directory && tar xvfp - )
リンクファイルが無い場合は次を実行する。
# scp -pr user1@host1.dom:/source/directory \
          user2@host2.dom:/dest/directory
```

サブディレクトリ全体のコピーに関する次の比較情報は Manoj Srivastava (`<srivasta@debian.org>`) により debian-user@lists.debian.org で掲示されました。

8.3.2 cp

伝統的に、`cp` はシンボリックリンクを参照しませんし、ハードリンク を保つので、本当はこの仕事に向いていません。もう 1 つ考えるべきことは、 まばらなファイル (穴を持つファイル) です。

GNU `cp` はこれらの制限を越えています。しかしながら、非 GNU システムでは、`cp` はまだこの問題を抱えています。また、 `cp` を用いて、小さく移植性の高いアーカイブを作成できません。

```
% cp -a . newdir
```

8.3.3 tar

Tar は `cp` が持つシンボリックリンクの問題のいくつかを 解決していますが、`cpio` は特別なファイルを扱えるため、 伝統的な `tar` は使用されていません。

`tar` の複数のハードリンクの操作方法は、テープ上に 1つ しかコピーを置かないことですが、コピーの名前はそのファイルを取得する ために使用される 唯一の 名前です。`cpio` の方法は 全リンクに対して 1つのコピーを置きますが、全リンクの名前を使ってファイル を取得できます。

`tar` コマンドは `.bz2` ファイルに対するオプションが Potato と Woody の間で変更されました。ですから、短縮形のオプションである `-I` (Potato) や `-j` (Woody) の代わりに、スクリプトでは `--bzip2` を使ってください。

8.3.4 `pax`

新しい POSIX (IEEE Std 1003.2-1992, pages 380–388 (section 4.48) と pages 936–940 (section E.4.48)) 互換の移植性のあるアーカイブ 交換ユーティリティです。`pax` はアーカイブファイルの読み書き、 内容のリスト表示が行え、ディレクトリ階層のコピーもできます。`pax` の操作は特別なファイルフォーマットに依存せず、さまざまなアーカイブフォーマット をサポートしています。

`pax` の実装はまだ新しく、未熟です。

```
# apt-get install pax
$ pax -rw -p e . newdir
 又は
$ find . -depth  | pax -rw -p e  newdir
```

8.3.5 `cpio`

`cpio` は `cpio` 又は `tar`アーカイブへの ファイルの操作ができます。アーカイブはディスク上の他のファイルにできますし、 磁気テープやパイプにもできます。

```
$ find . -depth -print0 | cpio --null --sparse -pvd new-dir
```

8.3.6 `afio`

`afio` は `cpio` フォーマットアーカイブの操作を 行うより良い手段です。般に、`cpio` よりも高速で、データの 変造をいくらか上品な方法で扱います。マルチボリュームなアーカイブをサポートし、さらに対話的な操作もサポートします。 `afio` は `tar` や `cpio` アーカイブ を圧縮するよりもより安全な圧縮アーカイブを作成できます。`afio` はバックアップスクリプト内で “アーカイブエンジン” として使うのにベストです。

```
$ find . -depth -print0 | afio -px -0a new-dir
```

私は磁気テープへのバックアップに `afio` を使っています。

8.4 差分バックアップとデータ同期

差分バックアップとデータ同期はいくつかの方法で実装できます。

- `rcs`: バックアップと履歴、テキストのみ
- `rdiff-backup`: バックアップと履歴、シンボリックリンクも可能
- `pdumpfs`: ファイルシステム内のバックアップと履歴。シンボリックリンクも可能
- `rsync`: 片方向同期
- `unison`: 双方向同期
- `cvs`: サーババックアップおよび履歴機能を持つ多方向同期。テキストのみだが熟成している。'Concurrent Versions System (CVS)' on page 213 参照
- `arch`: サーババックアップおよび履歴機能を持つ多方向同期、"作業ディレクトリ"のようなものでは無い。
- `subversion`: サーババックアップおよび履歴機能を持つ 多方向同期、Apache と連携。

'サブディレクトリ全体のコピーとアーカイブ' on page 108 に記述されているアーカイブ手法と 'タスクのスケジューリング (`cron`、`at`)' on page 123 に記述されている規則的なジョブの自動化を組み合わせると、すばらしい バックアップシステムとなります。

3つの使いやすいユーティリティについて説明します。

8.4.1 rdiff を用いた差分バックアップ

`rdiff-backup` はあらゆるタイプのファイルに対して 差分バックアップによる素晴らしくシンプルなバックアップを提供します。 `~/` のほとんどを `/mnt/backup` にバックアップするには、 次を実行します。

```
$ rdiff-backup --include ~/tmp/keep --exclude ~/tmp  ~/ /mnt/backup
```

このアーカイブから 3 日前のデータを `~/old` に復旧するには、 次を実行します。

```
$ rdiff-backup -r 3D /mnt/backup ~/old
```

`rdiff-backup(1)` をご覧ください。

8.4.2 **`pdumpfs`** による日々のバックアップ

`pdumpfs` は 毎日のスナップショットを保存する Plan9 の `dumpfs` と似たシンプルな日々のバックアップシステムです。 ある日のファイルを取得するために、あらゆる過去のスナップショットにアクセス できます。ホームディレクトリを `pdumpfs` と `cron` でバックアップしましょう!

`pdumpfs` は送り先ディレクトリに `YYYY/MM/DD` の形 でスナップショットを構築します。全ソースファイルは `pdumpfs` が最初に起動された際にスナップショットディレクトリにコピーされます。 次回の起動からは、ディスクスペースの節約のため、`pdumpfs` は更新されたか、新たに作成されたファイルのみコピーし、未更新ファイルは 前日のスナップショットのファイルにハードリンクされます。

```
$ pdumpfs src-dir dest-dir [dest-basename]
```

`pdumpfs(8)` をご覧ください。

8.4.3 RCS による規則的な差分バックアップ

`Changetrack` はテキストベースの設定ファイルの変更点を RCS アーカイブに定期的に記録します。 `changetrack(1)` をご覧ください。

```
# apt-get install changetrack
# vi changetrack.conf
```

8.5 システムフリーズからの回復

8.5.1 プロセスを殺す

`top` を起動し、どのプロセスの動きがおかしいかを調べます。 CPU 使用量でソートするためには 'P' 、メモリ量でソートするには 'M'、 プロセスを殺すには 'k' を押します。又は、BSD スタイルの `ps aux | less` や System-V スタイルの `ps -efH | less` も使えるかもしれません。 System-V スタイルの文法では親 ID (`PPID`) を表示するので、 ゾンビ (defunct) となった子プロセスを殺すのに使えます。

プロセス ID によりプロセスを殺す (又はシグナルを送る) には `kill` を、プロセスのコマンド名により同様のことを行うには `killall` を使います。良く使われるシグナルは次の通りです。

```
 1: HUP、デーモンを再起動
15: TERM、通常の kill
 9: KILL、激しく kill する
```

8.5.2 Alt-SysRq

システム故障に対する緊急手段が kernel のコンパイルオプション “Magic SysRq key” により供給されています。i386 で Alt-SysRq を押し、`r 0 k e i s u b` キーのうち一つを続けて押してください。すると魔法が効果を発揮します。

Un‘r’aw は X のクラッシュの後などにキーボードを回復します。端末の ログレベルを ‘0’ に減らすと、エラーメッセージが減少します。 sa‘k’ (sytem attention key) は現在の仮想端末上の全プロセスを殺します。 t‘e’rminate は現在の端末上の `init` 以外の全プロセスを殺します。 k‘i’ll は `init` 以外の全プロセスを殺します。

‘S’ync、 ‘u’mount、そして re‘b’oot は本当にひどい状況から脱出するための ものです。

Debian の標準でインストールされる kernel は本文書が書かれた現時点では 本オプション付でコンパイルされていません。kernel を再構築してこの機能を 有効にしてください。詳しい情報は `/usr/share/doc/kernel-doc-`*`version`*`/Documentation/sysrq.txt.gz` 又は `/usr/src/`*`kernel-version`*`/Documentation/sysrq.txt.gz` にあります。

8.6 覚えるべき小さくて洒落たコマンド

8.6.1 Pager

`less` は標準的なページャ (ファイル内容のブラウザ) です。 ヘルプは ‘h’ を押してください。`more` よりずっと多くのことが できます。`less` は `eval $(lesspipe)` や `eval $(lessfile)` をシェルの起動スクリプトで実行することにより ブーストできます。詳細は `/usr/share/doc/lessf/LESSOPEN` をご覧ください。 `-R` オプションにより、文字をそのまま表示し、ANSI の カラーエスケープシーケンスを有効にします。`less(1)` をご覧ください。

`w3m` もいくつかのエンコードシステム (EUC) では役に立つ ページャの代替品かもしれません。

8.6.2 フリーなメモリ

`free` と `top` はメモリリソースに関する良い情報を 提供します。“Mem:” 行にある “used” のサイズは心配しなくてもよいですが、 その下を読んでください。 (下の例の 38792)

```
$ free -k # for 256MB machine
             total       used       free     shared    buffers cached
Mem:        257136     230456      26680      45736     116136 75528
-/+ buffers/cache:      38792     218344
Swap:       264996          0     264996
```

物理メモリの正確な量は `grep '^Memory' /var/log/dmesg` により 確認できます。この場合、 “Memory: 256984k/262144k available (1652k kernel code, 412k reserved, 2944k data, 152k init)” のような出力となります。

```
合計          = 262144k = 256M (1k=1024, 1M=1024k)
dmesg でのフリーなメモリ量 = 256984k = Total - kernel - reserved - data - init
シェルでのフリーなメモリ量 = 257136k = Total - kernel - reserved - data
```

約 5MB は kernel が使用するため、システムにより使用不可にされています。

8.6.3 時刻の設定 (BIOS)

```
# date MMDDhhmmCCYY
# hwclock --utc --systohc
# hwclock --show
```

これはシステム時刻とハードウェア時刻を MM/DD hh:mm, CCYY にセットします。 時刻はローカル時間で表示されますが、ハードウェア時間は UTC を用います。

もしハードウェア (BIOS) 時刻が GMT に設定されている場合は、 `/etc/default/rcS` 中で `UTC=yes` と設定を変更します。

8.6.4 時刻の設定 (NTP)

Managing Accurate Date and Time HOWTO (`http://www.tldp.org/HOWTO/TimePrecision-HOWTO/index.html`) を参照。

インターネットへの常時接続により時刻を設定する

リモートのサーバ経由自動的にシステムのクロックを設定して時刻を修正します。

```
# ntpdate server
```

システムがインターネットに常時接続している場合、`/etc/cron.daily/` に書いておくのがよいでしょう。

散発的なインターネット接続で時刻を設定する

`chrony` パッケージを使用してください。

8.6.5 スクリーンセーバなどのコンソール機能の制御方法

スクリーンセーバを無効にするには、次のコマンドを使ってください。

Linux コンソール上では、次を実行します。

```
# setterm -powersave off
```

kon2 (kanji) コンソールのスクリーンセーバを起動します。

```
# kon -SaveTime 0
```

X 起動中には次を実行します。

```
# xset s off
  又は
# xset -dpms
  又は
# xscreensaver-command -prefs
```

他のコンソール機能を制御するには、対応するマニュアルを読んでください。 ターミナルの行設定を変更したり印刷するには、`stty(1)` も読んでください。

8.6.6 管理データベースの検索

Glibc は管理データベースからエントリ、すなわち、パスワード、グループ、 ホスト、サービス、プロトコル、ネットワークなどを検索するために、 `getent(1)` を提供しています。

```
getent database [key ...]
```

8.6.7 サウンド (ビープ音) を無効にする

PC スピーカを常に抜いておくこともできます ;-) Bash シェルでは次を実行します。

```
echo "set bell-style none">> ~/.inputrc
```

8.6.8 コンソールのスクリーン上のエラーメッセージ

スクリーンに出るエラーメッセージを黙らせるには、まず `/etc/init.d/klogd` をチェックします。このスクリプトに `KLOGD=“-c` *3”* をセットし、 `/etc/init.d/klogd restart` を起動します。 代わりに、`dmesg -n`*3* を起動することもできます。

エラーレベルの意味を示します。

- 0: KERN_EMERG, システムが使用不可
- 1: KERN_ALERT, 直ちに行動をとる必要がある
- 2: KERN_CRIT, 致命的状況
- 3: KERN_ERR, エラー発生
- 4: KERN_WARNING, 警告段階

- 5: KERN_NOTICE, 通常状態だが深刻な状況
- 6: KERN_INFO, インフォメーション
- 7: KERN_DEBUG, デバッグレベルのメッセージ

特定の使えないエラーメッセージが煩わしい場合、`shutup-abit-bp6` のような些細な kernel patch を適用することを考慮に入れましょう。 (examples subdirectory (`http://www.debian.org/doc/manuals/debian-reference/examples/`) で入手可能)

他に見るべき箇所は、`/etc/syslog.conf` です。コンソールデバイスにメッセージが出力されるようになっているかをチェックしましょう。

8.6.9 正しいタイプに端末を設定する

Unix ライクシステムでの端末スクリーンは通常 (n)curses ライブラリルーチンを使ってアクセスされます。これらのライブラリはユーザに 適度な最適化による端末に依存しない文字端末の更新手段を与えます。 `ncurses(3X)` と `terminfo(5)` をご覧ください。

Debian システムでは、前もって定義された多数のエントリがあります。

```
$ toe | less                        # 全エントリ
$ toe /etc/terminfo/ | less    # ユーザが再定義したエントリ
```

環境変数 `TERM` として選択したエントリをエクスポートします。

Debian でないホストから Debian システムにリモートでログインした時に、`xterm` 用の端末エントリが Debian でない `xterm` では動かない場合、 端末のタイプを “xterm” から “xterm-r6” などの機能制限されたものに変更してください。 詳細は `/usr/share/doc/libncurses5/FAQ` をご覧ください。“dumb” は terminfo 用の最小の共通部分 です。

8.6.10 端末を正常な状態に戻す

`cat` *`some-binary-file`* を実行してしまって端末が手を付けられなくなってしまった時は、次を実行します。(タイプしてもコマンドがエコーされて見えないかもしれません)

```
$ reset
```

8.6.11 DOS から Unix スタイルにテキストファイルを変換する

(EOF = `^M^J` である) DOS テキストファイルを (EOF = `^j` である) Unix テキストファイルに変換します。

```
# apt-get install sysutils
$ dos2unix dosfile
```

8.6.12 **recode** によりテキストファイルを変換する

次のコマンドにより、テキストファイルの改行コードをDOS, MAC, そして Unix 間 で変換できます。

```
$ recode /cl../cr <dos.txt >mac.txt
$ recode /cr.. <mac.txt >unix.txt
$ recode ../cl <unix.txt >dos.txt
```

フリーな `recode` はファイルの各種文字セットや surface を次のように 変換できます。

```
$ recode charset1/surface1..charset2/surface2 \
  <input.txt >output.txt
```

次に示す文字セットがよく用いられます (‘ロケール入門’ on page 167 も参照)。[1]

- `us` — ASCII (7 ビット)
- `l1` — ISO Latin-1 (ISO-8859-1, Western Europe, 8 ビット)
- `EUCJP` — 日本語用の EUC-JP (Unix)
- `SJIS` — 日本語用の Shift-JIS (Microsoft)
- `ISO2022JP` — 日本語用メールエンコーディング (7 ビット)
- `u2` — UCS-2 (Universal Character Set, 2 バイト)
- `u8` — UTF-8 (Universal Transformation Format, 8 ビット)

次に示す surface がよく用いられます。[2]

- `/cr` — 行端としてのキャリッジリターン (Mac テキスト)
- `/cl` — 行端としてのキャリッジリターン+ラインフィード (DOS テキスト)
- `/` — 行端としてのラインフィード (Unix テキスト)
- `/d1` — 可読な bytewise 10 進ダンプ
- `/x1` — 可読な bytewise 16 進ダンプ
- `/64` — Base64 エンコードのテキスト
- `/QP` — Quoted-Printable エンコードのテキスト

詳細は、`info recode` にある関連の説明をごらんください。

より目的に特化した変換ツールもあります。

- 文字セット変換ツール
  - `iconv` — ロケールエンコーディング変換ツール
  - `konwert` — しゃれたエンコーディング変換ツール
- バイナリファイル変換ツール
  - `uuencode` と `uudecode` — Unix 用
  - `mimencode` — メール用

---

[1] `recode` は `iconv` よりずっと便利なエイリアスが可能です。

[2] 行端

- キャリッジリターンは ASCII 13, ASCII 0xD, ^M, そして \r を意味します。
- ラインフィードは ASCII 10, ASCII 0xA, ^J, そして \n を意味します。

8.6.13 正規表現による置換

FILES ファイルの全てにある *FROM_REGEX* を *TO_TEXT* に置き換えます。

```
$ perl -i -p -e 's/FROM_REGEX/TO_TEXT/g;' FILES ...
```

`-i` は “in-place editing”、`-p` は “implicit loop over *FILES* . . .” のためのオプションです。置換が複雑な場合、パラメータ `i.bak` を `-i` の代わりに 使うことで容易にエラーから回復できます。これは元のファイルをそれぞれ `.bak` をファイルの拡張子として追加して保存しておきます。

8.6.14 スクリプトを使ってその場でファイルを編集する

次のスクリプトはその場で 5–10 行と 16–20 行を削除します。

```
#!/bin/bash
ed $1 <<EOF
16,20d
5,10d
w
q
EOF
```

ここで、`ed` コマンドは `vi` コマンドモードコマンドと同じです。スクリプトがファイルの後ろから編集するのを容易にします。

8.6.15 ソースファイルから差分を取得しマージする

次に挙げる手順は 2 つのソースファイルの差分を取り、ファイルの場所に応じて 統合された差分ファイルを *file.patch0* 又は *file.patch1* という名前で作成します。

```
$ diff -u file.old file.new > file.patch0
$ diff -u old/file new/file > file.patch1
```

差分ファイル (パッチファイルとも呼ばれる) はプログラムの更新箇所を送るために 使われます。差分ファイルを受け取ったパーティはこの更新を他の *file* に次のように適用します。

```
$ patch -p0 file < file.patch0
$ patch -p1 file < file.patch1
```

3つの異なるバージョンのソースコードを持っている場合は、`diff3` を使ってより効率的にこれらをマージできます。

```
$ diff3 -m file.mine file.old file.yours > file
```

8.6.16 大きなファイルを複数の小さなファイルに変換する

```
$ split -b 650m file   # ファイルを 650MB 毎に分割
$ cat x* >largefile    # 一つの大きなファイルにマージ
```

8.6.17 テキストファイルテーブルからデータを取り出す

全ての元 Debian リーダの名前と就任日がスペースで分割されたフォーマットで リストされている `DPL` と呼ばれるファイルを考えてみましょう。

```
Ian     Murdock   August  1993
Bruce   Perens    April   1996
Ian     Jackson   January 1998
Wichert Akkerman  January 1999
Ben     Collins   April   2001
Bdale   Garbee    April   2002
Martin  Michlmayr March   2003
```

これらのタイプのファイルからデータを取り出すには、Awk が良く使われます。

```
$ awk '{ print $3 }' <DPL                      # 開始月
August
April
January
January
April
April
March
$ awk '($1=="Ian") { print }' <DPL             # Ian と呼ばれる DPL の
リスト
Ian     Murdock   August  1993
Ian     Jackson   January 1998
$ awk '($2=="Perens") { print $3,$4 }' <DPL # Perens 就任時
April 1996
```

Bash などのシェルもこれらのファイルを解釈するのに使えます。

```
$ while read first last month year; do
    echo $month
  done <DPL
... 最初の Awk の例と同じ出力
```

ここで、`read` 組込みコマンドは $IFS(内部フィールドセパレータ) を用いて行を単語単位で分割します。

IFS を “:” に変更すると、`/etc/passwd` をシェルでうまくパーズ できます。

```
$ oldIFS="$IFS"   # 旧値を保存
$ IFS=":"
$ while read user password uid gid rest_of_line; do
    if [ "$user" = "osamu" ]; then
      echo "$user’s ID is $uid"
    fi
  done < /etc/passwd
osamu’s ID is 1001
$ IFS="$oldIFS"   # 旧値に戻す
```

(同じことを Awk を使って行うには、`FS=":"` を用いてフィールド セパレータをセットします。)

IFS はパラメータ拡張、コマンド置換、算術拡張の結果を分割するためにも シェルにより使われます。これらはダブル クォートやシングル クォートされた 単語内では発生しません。IFS の標準値は <space>, <tab>, と <newline> の組合せです。

シェルの IFS トリックを注意深く使ってください。 シェルがスクリプトの一部を **入力** として解釈した場合、 奇妙なことが起きるかもしれません。

```
$ IFS=":,"                          # ":" と "," を IFS として使う
$ echo IFS=$IFS,   IFS="$IFS"       # echo は Bash 組込みコマンド
IFS=  , IFS=:,
$ date -R                           # 単なるコマンド出力
Sat, 23 Aug 2003 08:30:15 +0200
$ echo $(date -R)                   # サブシェルの出力をメインシェルに入力
Sat  23 Aug 2003 08 30 36 +0200
$ unset IFS                         # IFS を標準にリセット
$ echo $(date -R)
Sat, 23 Aug 2003 08:30:50 +0200
```

8.6.18 コマンドをパイプするためのわずかなスクリプト

次のスクリプトはパイプの一部として素晴らしいことをします。

```
find /usr | egrep -v "/usr/var|/usr/tmp|/usr/local"
                    # いくつかのファイルを除き /usr にある全ファイルを見
つける
xargs -n 1 command   # 標準入力からの全てのアイテムに対してコマンドを実行
xargs -n 1 echo |    # スペース区切りのアイテムを行単位で分割する
xargs echo      |    # 全ての行を 1 行にマージする
grep -e pattern|     # patternを含む行を取り出す
cut -d: -f3 -|
           # : で区切られた 3 番目のフィールドを取り出す (passwd ファイルな
ど)
```

```
awk '{ print $3 }' | # スペースで区切られた 3 番目のフィールドを取り出す
awk -F'\t' '{ print $3 }' |
        # タブで区切られた 3 番目のフィールドを取り出す
col -bx |            # バックスペースを削除し、タブをスペースに変換する
expand -|            # タブをスペースに変換する
sort -u|             # 入力をソートし重複箇所を削除する
tr '\n' ' '|         # 複数の行を 1 行に繋げる
tr '\r' ''|          # キャリッジリターンを削除
tr 'A-Z' 'a-z'|      # 大文字を小文字に変換する
sed 's/^/# /'|       # 各行をコメントアウトする
sed 's/\.ext//g'|    # .ext を削除する
sed  -n -e 2p|       # 2 番目の行を表示
head -n 2 -|         # 最初の 2 行を表示
tail -n 2 -|         # 最後の 2 行を表示
```

8.6.19 ファイルごとにループするためのわずかなスクリプト

次の方法は各ファイルについて `*.ext` とのマッチングを行います。スペースを持つファイル名などを適切に扱うことを保証し、どの方法も同じように処置します。

- シェルループ (これは `PS2="`"` を用いた複数行の入力スタイルです。この行を1行でタイプする場合、各行を終わらせるために、いくつかのセミコロンを追加する必要があります)

```
for x in *.ext; do
  if test -f "$x"; then
    command "$x"
  fi
done
```

- `find` と `xargs` の組合せ。

```
find . -type f -maxdepth 1 -name '*.ext' -print0 | \
 xargs -0 -n 1 command
```

- `find` の `-exec` オプションによりコマンドを使った場合。

```
find . -type f -maxdepth 1 -name '*.ext' \
 -exec command '{}' \;
```

- `find` の `-exec` オプションにより短いシェルスクリプトを使う場合。

```
find . -type f -maxdepth 1 -name '*.ext' \
 -exec sh -c "command '{}' && echo 'successful'" \;
```

8.6.20 Perl ショートスクリプトの狂気

Awk スクリプトは `a2p(1)` を用いれば自動的に Perl で書き直せますが、ワンライン Awk スクリプトは手動でワンライン Perl スクリプト に完璧に変換できます。例えば、

```
awk '($2=="1957") { print $3 }' |
```

は次の行のどれとも同じです。

```
perl -ne '@f=split; if ($f[1] eq "1957") { print "$f[2]\n"}' |
perl -ne 'if ((@f=split)[1] eq "1957") { print "$f[2]\n"}' |
perl -ne '@f=split; print $f[2] if ( $f[1]==1957 )' |
perl -lane 'print $F[2] if $F[1] eq "1957"' |
```

上の行の `perl` への引数の全てのスペースはすべて削除できますし、Perl における番号と文字列の間の自動変換の利点を利用しています。

```
perl -lane 'print$F[2]if$F[1]eq+1957' |
```

コマンドラインオプションについては `perlrun(1)` をご覧ください。より狂った Perl スクリプトについては、`http://perlgolf.sourceforge.net` が興味深いかもしれません。

8.6.21 web ページからテキストやメーリングリストのアーカイブを取得する

次のコマンドは web ページを読み込んでテキストファイルに整形します。 Web から設定をコピーするときに非常に役立ちます。

```
$ lynx -dump http://www.remote-site.com/help-info.html >textfile
```

ここで `links` と `w3m` も使えますが、描画に少し 違いがあります。

これがメーリングリストのアーカイブの場合、mime contents からテキストにするには `munpack` を使います。

8.6.22 web ページをきれいに印刷する

次のコマンドは web ページを PostScript ファイルに変換して印刷します。

```
$ apt-get install html2ps
$ html2ps URL | lpr
```

'`lpr`又は`lpd`' on page 42 をご覧ください。また、PostScript ファイルを作成するには `a2ps` および `mpage` パッケージをチェックしてみてください。

8.6.23 マニュアルページをきれいに印刷する

次のコマンドはマニュアルページを PostScript に変換して印刷します。

```
$ man -Tps some-manpage | lpr
$ man -Tps some-manpage | mpage -2 | lpr
```

8.6.24 2つの PostScript や PDF ファイルをマージする

2つの PostScript や PDF ファイルをマージできます。

```
$ gs -q -dNOPAUSE -dBATCH -sDEVICE=pswrite \
  -sOutputFile=bla.ps -f foo1.ps foo2.ps
$ gs -q -dNOPAUSE -dBATCH -sDEVICE=pdfwrite \
  -sOutputFile=bla.pdf -f foo1.pdf foo2.pdf
```

8.6.25 コマンドの実行時間を計測する

プロセスにより使用された時間を表示します。

```
# time some-command >/dev/null
real    0m0.035s       # 壁掛け時計の時間 (真の経過時間)
user    0m0.000s       # ユーザモードでの時間
sys     0m0.020s       # kernel モードでの時間
```

8.6.26 **nice** コマンド

起動時にコマンドの nice 値をセットするには、`nice` (GNU の `shellutils` パッケージに収録) を使ってください。 `renice` (`bsdutils`) や `top` も プロセスを renice できます。nice 値が 19 の場合はプロセスは最低速度 (最低の優先度) なプロセスです。負の nice 値は “not-nice” であり、 -20 では最速 (最高の優先度) なプロセスです。スーパーユーザだけが負の nice 値をセットできます。

```
# nice  -19 top                                      # 非常
に nice
# nice --20 cdrecord -v -eject speed=2 dev=0,0 disk.img # 非常に高
速
```

極端な nice 値はシステムに悪影響を及ぼすことが時々あります。注意して このコマンドを使ってください。

8.6.27 タスクのスケジューリング(`cron`、`at`)

Linux で仕事をスケジューリングするには、`cron` や `at` を使いましょう。`at(1)`、`crontab(5)`、`crontab(8)` をご覧ください。

定期的にスケジューリングするイベントを設定するには、`crontab -e` コマンドを起動して crontab ファイルを作成したり編集します。 crontab ファイルの例:

```
    # (/etc/passwd の指定に関らず) コマンド実行に /bin/sh を使用する。
    SHELL=/bin/sh
    # (この crontab の所有者に関らず) あらゆる出力を `paul' にメールする。
    MAILTO=paul
    # 分 時 月内日 月 曜日 コマンド (Day... are OR'ed)
    # 毎日、日付変更の 5 分後に実行する
    5 0 * * *       $HOME/bin/daily.job >> $HOME/tmp/out 2>&1
    # 毎月初日の 2:15pm に実行する -- 出力は paul にメールされる
    15 14 1 * *     $HOME/bin/monthly
    # 平日の午後 10 時に実行してジョーを心配させる
    0 22 * * 1-5 mail -s "午後 10時だ" joe%ジョー、%%お前の子どもはどこだ
い?%
    23 0-23/2 * * * echo "毎日 0, 2, 4..時 23 分に実行する"
    5 4 * * sun     echo "日曜 4時 5分に実行する"
    # 毎月第一日曜日の午前 3時 40分に実行する
    40 3 1-7 * *    [ "$(date +%a)" == "Mon" ] && command -args
```

一度きりの仕事をスケジューリングするには、`at` コマンドを起動します。

```
    $ echo 'command -args'| at 3:40 monday
```

8.6.28 `screen` を使って端末を切替える

`screen` プログラムにより、単独の実際の端末又は端末エミュレータウィンドウ上で複数の仮想端末を起動させ、それぞれの端末で固有のの対話型シェルを持たせることが可能となります。 Linux の仮想端末や複数の `xterm` ウィンドウを使っている場合でさえも、次に示す `screen` の豊富な機能セットを探検する価値はあります。

- スクロールバック履歴
- コピー & ペースト
- 出力のロギング
- digraph エントリ
- 端末から `screen` セッション全体を **切り離し**、後で再び取り付ける能力。

遠隔アクセスのシナリオ

リモートターミナルからや VT100 ターミナルプログラムを使って Linux マシンによくログインする場合、`screen`を使うと **detach (切り離し)** 機能により生活がより簡単になります。

1. ダイアルアップ接続経由でログインし、いくつかのウィンドウでエディタや他のプログラムを開いて複数の `screen` を起動します。
2. 突然端末を離れることが必要になりましたが、ハングアップすることにより今までやった仕事を失いたくありません。
3. `^A d` をタイプしてセッションを **デタッチ** してログアウトします (又は、より素早くやるには、`^A DD` をタイプして `screen` をデタッチして screen 自身をログアウトします。)
4. 再びログインする時に、`screen -r` コマンドを入力すると、`screen` は魔法のように前に開いていたウィンドウ全てを **リアタッチ** します。

典型的な screen コマンド

いったん `screen` を始めると、標準では `^A` であるコマンドキーストロークを除く全てのキーボード入力が現在のウィンドウに送られます。全ての `screen` コマンドは `^A` に加えてさらに一つのキー [に加えてパラメータ] をタイプすることにより入力されます。役立つコマンドを挙げます。

`^A ?`	ヘルプスクリーンを表示 (キーバインドダイアログ)
`^A c`	新規ウィンドウを作成してそこに切替える
`^A n`	次のウィンドウに移動
`^A p`	前のウィンドウに移動
`^A` *0*	ウィンドウナンバー *0* に移動
`^A w`	ウィンドウリストを表示
`^A a`	カレントウィンドウにキーボード入力として Ctrl-A を送信
`^A h`	カレントウィンドウのハードコピーをファイルに書き出す
`^A H`	カレントウィンドウのファイルへのログ記録を開始/終了
`^A ^X`	端末をロック (パスワードによる保護)
`^A d`	ターミナルからスクリーンセッションをデタッチする
`^A DD`	スクリーンセッションをデタッチしてログアウトする

これは `screen` コマンドと機能の小さなサブセットに過ぎません。`screen` にやらせたいことが何かあるなら、たぶんできるでしょう。詳細は `screen(1)` をご覧ください。

screen セッションでのバックスペースと Ctrl-H

`screen` を起動中にバックスペースや Ctrl-H が動かないことに気づいた場合、`/etc/screenrc` を編集し、次の行を見つけたらコメントアウトしてください。(すなわち、この行の先頭に “#” を追加してください)

```
bindkey -k kb stuff "\177"
```

X 用の screem と等価なプログラム

`xmove` を調べてください。`xmove(1)` をご覧ください。

8.6.29 ネットワークテストの基礎

`netkit-ping`, `traceroute`, `dnsutils`, `ipchains` (2.2 kernel 用), `iptables` (2.4 kernel 用), そして `net-tools` パッケージをインストールして次を実行します。

```
   $ ping yahoo.com              # インターネット接続をチェック
   $ traceroute yahoo.com        # IP パケットを追跡
   $ ifconfig                    # ホストの設定をチェック
   $ route -n                    # ルーティング設定をチェック
   $ dig [@dns-server.com] host.dom [{mx|any}] |less
         # dns-server.com による {mx|any} レコード
         # に対する host.dom DNS レコードをチェックする
   $ ipchains  -L -n |less       # パケットフィルタをチェッ
ク (2.2 kernel 用)
   $ iptables -L -n |less        # パケットフィルタをチェッ
ク (2.4 kernel 用)
   $ netstat -a                  # 開いているポート全てを見つける
   $ netstat -l --inet           # 接続待ちのポートを見つける
   $ netstat -ln --tcp           # 接続待ちの TCP ポート番号を見つける
```

8.6.30 ローカルのスプールからメールをフラッシュする

ローカルスプールからメールをフラッシュするには、次を実行します。

```
   # exim4 -q    # 送信待ちメールをフラッシュする
   # exim4 -qf   # 全メールをフラッシュする
   # exim4 -qff  # frozen したメールもフラッシュする
```

`-qff` オプションを `/etc/ppp/ip-up.d/exim` スクリプト にオプションとして入れるのが良いかもしれません。 Woody 以前では、`exim4` を `exim` に置き換えてください。

8.6.31 frozen したメールをローカルスプールから削除する

ローカルスプールから frozen したメールを配送エラーメッセージを付けて削除 するには、次を実行します。

```
   # exim4 -Mg `mailq | grep frozen | awk '{ print $3 }'`
```

Woody 以前では、`exim4` を `exim` に置き換えてください。

8.6.32 **`mbox`** の内容を再配送する

ホームディレクトリが満杯になり、`procmail` が失敗する場合、`/var/mail/`*`username`* からホームディレクトリにあるソートされたメールボックスにメールを手動で送信する必要があります。ホームディレクトリにディスクスペースを確保してから、次を実行します。

```
# /etc/init.d/exim4 stop
# formail -s procmail </var/mail/username
# /etc/init.d/exim4 start
```

Woody 以前では、`exim4` を `exim` に置き換えてください。

8.6.33 ファイルの内容をクリアする

ログファイルなどのファイルの内容をクリアにするには、`rm` を使ってファイルを削除してから新しい空のファイルを作成しないでください。ファイルは削除して作成する間にまだアクセスされるかもしれないからです。次の方法はファイルの内容をクリアにします。

```
$ :>file-to-be-cleared
```

8.6.34 ダミーファイル

次のコマンドはダミー又は空のファイルを作成します。

```
$ dd if=/dev/zero    of=filename bs=1k count=5 # 5KB の空の内容
$ dd if=/dev/urandom of=filename bs=1M count=7 # 7MB のランダムな内
容
$ touch filename # 0B ファイルを作成 (既にある場合、mtime を更新)
```

例えば、最も実用的な使いかたとして、Debian のブートフロッピから実行される次のコマンドはハードディスク `/dev/hda` の内容を完全に削除します。

```
# dd if=/dev/urandom of=/dev/hda ; dd if=/dev/zero of=/dev/hda
```

8.6.35 **`chroot`**

`chroot` プログラム (`chroot(8)`) はリブートせずに単独のシステムで同時に異なる GNU/Linux 環境を起動することを可能にします。

また、低速なサテライトマシンを NFS マウントし、そこをより高速なマシンが chroot ポイントとして `apt-get` や `dselect` のようなリソース食いなプログラムを実行することもできます。

chroot を使って異なる Debian ディストリビューションを起動する

Sarge にある `debootstrap` コマンドを使うと、chroot した Debian 環境を容易に構築できます。 Sarge 以降のディストリビューションでは、適切なオプションを付ける代わりに `cdebootstrap` コマンドを使えるかもしれません。 例えば、高速なインターネットアクセス環境を持っており、*/sid-root* に Sid の chroot を作成するには、 次を実行します。

```
   main # cd / ; mkdir /sid-root
   main # debootstrap sid /sid-root http://ftp.debian.org/debian/
   ... システム全体がダウンロードされるのを監視する
   main # echo "proc /sid-root/proc proc none 0 0" >> /etc/fstab
   main # mount proc /sid-root/proc
   main # mount /dev/ /sid-root/dev -o bind
   main # cp /etc/hosts /sid-root/etc/hosts
   main # chroot /sid-root /bin/bash
   chroot # cd /dev; /sbin/MAKEDEV generic ; cd -
   chroot # apt-setup # set-up /etc/apt/sources.list
   chroot # vi /etc/apt/sources.list # source を unstable が指すように
編集
   chroot # dselect  # aptitude を使って、mc や vim をインストールしてもよ
い :)
```

この時点で完全に機能する Debian システムを持っているはずです。 ここではメインの Debian 環境に影響を与える恐れなく動きまわれます。

この `debootstrap` のトリックは Debian ではなく他の GNU/Linux ディストリビューションのインストールディスクを使って Debian をインストール するのにも使えます。

chroot 用のログイン設定

`chroot` *`/sid-root`* `/bin/bash` をタイプするのは簡単ですが、 意図せずに環境変数を全て引き継いでしまったりします。chroot に直接ログイン できる別の仮想端末で他のログインプロセスを起動するのがよりよい方法です。

標準の Debian システムでは、`tty1` から `tty6` は Linux 端末を起動し、`tty7` は X Window を起動します。例えば `tty8` を chroot 化された端末に設定しましょう。 '`chroot` を使って異なる Debian ディストリビューションを起動する' on page 127 に記述したように chroot システムを作成した後、 メインシステムの root シェルから次をタイプします。

```
   main # echo "8:23:respawn:/usr/sbin/chroot /sid-root "\
         "/sbin/getty 38400 tty8"  >> /etc/inittab
   main # init q    # init のリロード
```

`chroot` 用の X 設定

chroot で最新の X や GNOME を安全に起動したいですか? これは可能です。次の例は仮想端末 `vt9` で GDM を起動させます。

まず '`chroot` を使って異なる Debian ディストリビューションを起動する' on page 127 に記述している方法を用いて chroot システムをインストールします。メインシステムの root から chroot システムに主要な設定ファイルをコピーします。

```
main # cp /etc/X11/XF86Config-4 /sid-root/etc/X11/XF86Config-4
main # chroot /sid-root # 又は chroot の端末を使う
chroot # cd /dev; /sbin/MAKEDEV generic ; cd -
chroot # apt-get install gdm gnome x-window-system
chroot # vi /etc/gdm/gdm.conf # [servers] セクションの vt7 を vt9 に
書き換える
chroot # /etc/init.d/gdm start
```

ここで、`/etc/gdm/gdm.conf` は最初の仮想端末を `vt7` から `vt9` に変更しました。

今 chroot での完全な X 環境とメインシステムの間を Linux 仮想端末を切替えること、例えば Ctrl-Alt-F7 や Ctrl-Alt-F9 を使うことにより簡単に切替えることができます。お楽しみに!

[FIXME] Add a comment and link to the init script of the chrooted `gdm`.

`chroot` を使って他のディストリビューションを起動する

他の Linux ディストリビューションのための chroot 環境も簡単に作れます。他のディストリビューションのインストーラを使って別のパーティションにシステムをインストールします。その root パーティションが */dev/hda9* とすると、次のようになります。

```
main # cd / ; mkdir /other-dist
main # mount -t ext3 /dev/hda9 /other-dist
main # chroot /other-dist /bin/bash
```

そして '`chroot` を使って異なる Debian ディストリビューションを起動する' on page 127, '`chroot` 用のログイン設定' on page 127, や '`chroot` 用の X 設定' on page 128 にある手順を進めます。

`chroot` によりパッケージを作成する

より特化した chroot パッケージである `pbuilder` は chroot システムを構築し、chroot 内でパッケージを作成します。パッケージの build-dependencies が正しいかどうかをチェックしたり、不必要で間違った構築依存性がパッケージに存在しないかを確認するための理想的なシステムです。

8.6.36 ハードリンクのチェック方法

2つのファイルがハードリンクを使った同じファイルかどうかチェックするには次を実行します。

```
$ ls -li file1 file2
```

8.6.37 ハードディスクイメージファイルを mount する

`file.img` がハードディスクの内容のイメージを含んでおり、オリジナルのハードディスクのディスク設定が *xxxx* = (bytes/sector) * (sectors/cylinder) であった場合、次のコマンドはこのイメージを `/mnt` にマウントします。

```
# mount -o loop,offset=xxxx file.img /mnt
```

ほとんどのハードディスクは一セクタあたり 512 バイトであることに注意してください。

8.6.38 Samba

Windows からファイルを取得する基本的な方法は次の通りです。

```
# mount -t smbfs -o username=myname,uid=my_uid,gid=my_gid \
        //server/share /mnt/smb  # Windows ファイルを Linux にマウン
ト
# smbmount //server/share /mnt/smb \
        -o "username=myname,uid=my_uid,gid=my_gid"
# smbclient -L 192.168.1.2 # コンピュータ上の共有のリスト
```

Linux から Samba の隣人は次のようにチェックできます。

```
# smbclient -N -L ip_address_of_your_PC | less
# nmblookup -T "*"
```

8.6.39 他の OS のファイルシステムを扱うためのユーティリティ

他の OS のファイルシステムの多くを Linux kernel はサポートしており、単にマウントすることによりファイルシステムを含むデバイスにアクセスできます。特定のファイルシステムのために、デバイスをマウントせずにファイルシステムにアクセスするための専用のツールが存在します。これは kernel のファイルシステムのサポートを必要としないようにユーザスペースのプログラムとして実装されています。

- `mtools`: MS-DOS ファイルシステム(MS-DOS, Windows) 用
- `cpmtools`: CP/M ファイルシステム用
- `hfsutils`: HFS ファイルシステム (昔の Macintosh) 用
- `hfsplus`: HFS+ ファイルシステム (最近の Macintosh) 用

MS-DOS FAT ファイルシステムの作成やチェックを行うには、`dosfstools` が役立ちます。

8.7 注意すべき典型的な間違い

ここに危険な行動の例をいくつか挙げます。特権アカウント `root` を使う場合、副作用はさらに増大します。

8.7.1 `rm -rf .*`

“`.*`” は “`.`” と “`..`” を含むように拡張されるので、ワイルドカードファイル名をコマンド行引数で “`rm -rf .*`” のように使うと危険な結果をもたらす危険があります。幸いにも、Debian ディストリビューションに含まれている “rm” コマンドの最新版では、引数ファイル名の健全性を検査し、“`.`” と “`..`” を削除します。しかし、これはいつも成り立つとは限りません。ワイルドカードファイル名がどのように機能するかを知るには、次をお試しください。

- “`echo *`”:カレントディレクトリにある全ての非ドットファイルと非ドットディレクトリをリストする
- >“`echo .[^.]*`”:カレントディレクトリにある全てのドットファイルとドットディレクトリをリストする
- “`echo .*`”:親ディレクトリにある全てと親ディレクトリ自身をリストする

8.7.2 `rm /etc/passwd`

Debian システムは `/etc/passwd` のような重要なファイルを間違いにより失うことに対して大丈夫です。Debian システムはこれらを `/var/backups/` に定期的にバックアップします。これらのファイルを回復させたら、適切な権限を手動でセットしましょう。

```
# cp /var/backups/passwd /etc/passwd
# chmod 644 /etc/passwd
```

‘パッケージ選択データの回復’ on page 87 をご覧ください。

章 9

Debian システムのチューニング

本章はシステム設定の基本をコマンドラインインターフェースのみで説明します。本章を読む前に ‘Debian システムインストールのヒント’ on page 27 を読んで下さい。

セキュリティに関して関心があるなら、`harden-doc` パッケージにある Securing Debian Manual (`http://www.debian.org/doc/manuals/securing-debian-howto/`) を読んでください。

9.1 システムの初期化

Debian は Systen V 形式の init スクリプトシステムを採用しています。イントロダクションとして ‘`init` プログラム’ on page 21 をご覧ください。

9.1.1 init スクリプトのカスタマイズ

init スクリプトの挙動を制御する最も易しい方法は `/etc/default/` ディレクトリにある init スクリプトのような名前のファイルにある環境変数の指定を変更することです。[1] 例えば、`/etc/default/hotplug` は `/etc/init.d/hotplug` の挙動を制御するために使えます。`/etc/default/rcS` ファイルは `motd`、`sulogin` などのブート時の標準をカスタマイズするために使えます。

これらの変数を変更することにより望みの挙動を実現できない場合は、init スクリプト自身を修正できます。これらは全て設定ファイルです。

---

[1] `/etc/default/` にあるディレクトリは環境変数の割り当て**のみ**を含みます。各ファイルはこれらの割り当てが init スクリプト自身にある標準の変数の設定を上書きするような方法で、対応する init スクリプトにより読み込まれます。このディレクトリ名の選択は Debian 独特 (`http://lists.debian.org/debian-devel/2003/debian-devel-200308/msg02114.html`) のものです。大まかに言うと、Red Hat や他のディストリビューションで見られる `/etc/sysconfig` と同じものです。

9.1.2 システムのログ記録をカスタマイズする

システムのログモードは `/etc/syslog.conf` を使って設定可能です。システムのログファイルに色づけするためのプログラムについては `colorize` パッケージをチェックしてください。又、`syslogd(8)` や `syslog.conf(5)` もご覧ください。

9.1.3 ハードウェアの最適化

Debian でシステム管理者に管理が任されているハードウェアの最適化設定はあまりありません。

- `hdparm`
  - ハードディスクアクセスの最適化ツール。非常に効果的。
  - 危険です。まず `hdparm(8)` を読む必要があります。
  - `hdparm -tT` *`/dev/hda`* によりディスクのアクセス速度をテストします。
  - `hdparm -q -c3 -d1 -u1 -m16` *`/dev/hda`* により最近の IDE システムではディスク速度を高速化できます。(危険かもしれません。)
- `setcd`
  - コンパクトディスクドライブのアクセスの最適化ツール。
  - `setcd -x` *`2`* により速度を 2 倍速に落とします。
  - `setcd(1)` をご覧ください。
- `setserial`
  - シリアルポートの管理ツール集。
- `scsitools`
  - SCSI ハードウェアの管理ツール集。
- `memtest86`
  - メモリデバイスの管理ツール集。
- `hwtools`
  - 低レベルハードウェア管理ツール集。
    - `irqtune`: 優先度が高く高速なサービスを要求するデバイス (例: シリアルポートやモデム) にデバイスの IRQ 優先度を変更する。シリアルポート/モデムの 3 倍の高速化が可能。
    - `scanport`: 装着した ISA デバイスを探すために 0x100 から 0x3ff までの I/O 空間を検索する。
    - `inb`: I/O ポートを読み込み、8 進数又はバイナリ形式で値をダンプする小さなクイックハック。
- `schedutils`
  - Linux スケジューラユーティリティ。
  - `taskset`,`irqset`,`lsrt`, と `rt` が含まれます。

  - (これには含まれていない) `nice` や `renice` と組み合わせて、プロセスのスケジューリングパラメータの完全な制御が行えます。

`noatime` オプション付でファイルシステムをマウントするのもファイルへの読み込みアクセスが高速化するには非常に効率的です。 `fstab(5)` と `mount(8)` をご覧ください。

proc ファイルシステムを通じて Linux kernel 自身により直接チューンできるハードウェアもあります。'proc filesystem により Linux kernel をチューニングする' on page 100 をご覧ください。

Debian には特定のハードウェアに特化した設定ユーティリティが数多く存在します。 これらの多くはノート PC に特定したニーズに対応しています。ここに Debian で得られるいくつかの興味深いパッケージを挙げます。

- `tpconfig` - タッチパッドデバイスの設定プログラム
- `apmd` - Advanced Power Management (APM) 用ユーティリティ
- `acpi` - ACPI デバイスの情報を表示
- `acpid` - ACPI を使うためのユーティリティ
- `lphdisk` - Phoenix NoteBIOS 用のハイパネーション パーティションの準備を行う
- `sleepd` - 使わない間ノート PC をスリープさせる
- `noflushd` - アイドルになっているハードディスクの回転を止める
- `big-cursor` - X 用の大きいマウスカーソル
- `acme` - ノート PC にある "マルチメディアボタン" を有効にする
- `tpctl` - IBM ThinkPad ハードウェア設定ツール
- `mwavem` - Mwave/ACP モデムのサポート
- `toshset` - 東芝ノート PC のインターフェースの多くにアクセスする
- `toshutils` - 東芝ノート PC 用ユーティリティ
- `sjog` - Sony VAIO ノートの "ジョグダイアル" を使うためのプログラム
- `spicctrl` - LCD のバックライトの輝度設定用の Sony VAIO コントローラプログラム

ここで、ACPI は APM より新しい電源管理システムのフレームワークです。

これらのパッケージのいくつかは特別な kernel モジュールを要求します。 これらは多くの場合、既に最新の kernel ソースに含まれています。 問題が発生した場合、kernel に最新のパッチを適用する必要があるかもしれません。

9.2 アクセス制限

9.2.1 PAM を用いたログイン制限

PAM (Pluggable Authentication Modules) により、ユーザのログイン方法を制御できます。

```
/etc/pam.d/*              # PAM 制御ファイル
/etc/pam.d/login          # ログイン用の PAM 制御ファイル
/etc/security/*           # PAM モジュールパラメータ
/etc/securetty            # 端末による root でのログインを制御
/etc/login.defs           # ログインの挙動を制御
```

自分でリスクを負って、安全ではないがパスワード無しで端末にログインしたい場合は、`/etc/pam.d/login`の内容を次のように変えてください。

```
#auth       required   pam_unix.so nullok
auth       required   pam_permit.so
```

パスワード無しで X コンソールにログインするために同様のトリックを `xdm`, `gdm` などに適用できます。

一方、良いパスワードポリシーを実施したい場合には、次のように `cracklib2` をインストールして `/etc/pam.d/passwd`をセットしてください。

```
password required       pam_cracklib.so retry=3 minlen=6 difok=3
```

ワンタイムログインパスワードも役に立つかもしれません。この目的には、`passwd` コマンドを `-e` オプション付で使ってください。`passwd(1)` をご覧ください。

Bash シェルではプロセスの最大数を `ulimit -u 1000` としてセットできますし、`/etc/security/limits.conf` で PAM から設定することもできます。`core` のような他のパラメータも同様に設定できます。`PATH` の初期値はシェル起動スクリプトの起動前に `/etc/login.defs` によりセットできます。

PAM のドキュメントは `libpam-doc` パッケージでパッケージングされています。*Linux-PAM システム管理者ガイド* は PAM の設定、必要なモジュールなどについてカバーしています。このドキュメントは *Linux-PAM アプリケーション開発ガイド* と *Linux-PAM モジュール開発者ガイド* も含んでいます。

9.2.2 “なぜ GNU su は wheel グループをサポートしていないのか”

これは Richard M. Stallman による古い `info su` ページの末尾にある有名なフレーズです。心配ありません。Debian にある現在の `su` は PAM を使っているので、`su` が使える能力を `/etc/pam.d/su` にある `pam_wheel.so` を使ってあらゆるグループに対して制限できます。次の設定は Debian システムの `adm` グループを BSD の `wheel` グループと同じようにセットし、このグループに属するメンバーがパスワード無しで `su` を使用することを許可します。

```
# /etc/pam.d/su において反 RMS 設定を行う
auth       required   pam_wheel.so group=adm

# Wheel メンバーがパスワード無しで suを使えるようにする
auth       sufficient pam_wheel.so trust group=adm
```

9.2.3 標準で存在するグループの目的

興味深いグループがいくつか存在します。

- `root` グループは `pam_wheel.so` が `group=` 引数無しで使用されている場合の標準的な wheel グループです。
- `adm` グループはログファイルを読むことが可能です。
- `cdrom` グループは CD-ROM ドライブへのユーザのアクセスを 局所的に許すために使われます。
- `floppy` グループはフロッピドライブへのユーザのアクセスを 局所的に許すために使われます。
- `audio` グループはオーディオデバイスへのユーザのアクセスを 局所的に許すために使われます。
- `src` グループは `/usr/src/` にあるソースコードを 所有しています。システムのソースコードを管理する能力をユーザに局所的に 与えるために使われます。
- `staff` メンバは ヘルプデスクやシステム管理者補に対して `/use/local/` で作業を行い、`/home/` に ディレクトリを作成する能力を与えるのに役立ちます。

完全なリストは、Woody では `harden-doc` パッケージでも見つかる Securing Debian Manual (`http://www.debian.org/doc/manuals/securing-debian-howto/`) の “FAQ” セクションをご覧ください。 又、新しい `base-passwd` (3.4.6 以降) にも信頼できる リスト: `/usr/share/doc/base-passwd/users-and-groups.html` が含まれます。

9.2.4 – sudo を使ってより安全に作業をする

私の `sudo` の使いかたはほとんど自分のミスからの防御が目的です。 個人的には、`sudo` を使うことは、常に root としてシステムを使う ことに対するより良い代替手段と考えます。

`sudo` をインストールして `/etc/sudoers`(`http://www.debian.org/doc/manuals/debian-reference/examples/`) にあるオプションを 設定して有効にします。また、`/usr/share/doc/sudo/OPTIONS` にある `sudo` グループ機能を調べます。

設定サンプルは “staff” グループのメンバが `sudo` の元で root としてあらゆるコマンドを起動することと “src” メンバが `sudo` の元で root として選ばれたコマンドを起動することを許可します。

`sudo` の利点は、ユーザの普通のログインパスワードしか必要とせず、 その行動がモニタできるということです。これはシステム管理者補にいくらかの 権限を与える良い方法です。例えば、

```
$ sudo chown -R myself:mygrp .
```

もちろん root パスワードを知っている場合、(ほとんどのホームユーザがしているように) ユーザアカウントから root としてコマンドを起動できます。

```
$ su -c "shutdown -h now"
Password:
```

(管理者アカウントの `sudo` 権限を強化すべきだと知っていますが、 これは私のホームサーバなので、まだ困っていません。)

通常のユーザが root 権限でコマンドを実行することを許可するプログラムが 他にあります。`super` パッケージをご覧ください。

9.2.5 サービスへのアクセスを制限する

インターネット スーパーサーバ すなわち `inetd` は ブート時に `/etc/init.d/inetd` のシンボリックリンクである `/etc/rc2.d/S20inetd`(ランレベルが 2 の場合) により起動されます。 基本的に、`inetd` は 1 つの起動中のデーモンが複数の他のデーモン を起動し、システムの負荷を低減することを許可します。

サービス要求が届いた時には常に `/etc/protocols` と `/etc/services` にあるデータベース を検索することによりプロトコル並びにサービスが識別されます。 そして `inetd` は通常のインターネットサービスを `/etc/inetd.conf` データベース、 Sun-RPC ベースのサービスを `/etc/rpc.conf` にあるデータベースから 捜し出します。

システムのセキュリティのために、`/etc/inetd.conf` にある 使用しないサービスを必ず無効化してください。 Sun-RPC サービスは NFS や他の RPC ベースのプログラムが必要とします。

時々、`inetd` は意図したサーバを直接起動しませんが、 `tcpd` TCP/IP デーモンラッパプログラムが意図したサーバ名を用いて `/etc/inetd.conf` にある引数として起動されます。

この場合、`tcpd` は要求を記録した後に適切なサーバプログラムを 起動し、`/etc/hosts.deny` と `/etc/hosts.allow` を使ったいくつかの付随的な検査を行います。

最近の Debian システムで遠隔アクセスに問題が発生した場合、 `/etc/hosts.deny` に “ALL: PARANOID” という行が存在する 場合はこれをコメントアウトしてください。

詳細については `inetd(8)`, `inetd.conf(5)`, `protocols(5)`, `services(5)`, `tcpd(8)`, `hosts_access(5)`,そして `hosts_options(5)` をご覧ください。

Sun-RPC についての詳細は、 `rpcinfo(8)`, `portmap(8)`, そして `/usr/share/doc/portmap/portmapper.txt.gz` をご覧ください。

9.2.6 集権的認証 – LDAP

Lightweight Directory Access Protocol (LDAP) を使いましょう。 以下を参照してください。

- OpenLDAP (`http://www.openldap.org/`)
- `openldap-guide` パッケージにある OpenLDAP 管理者ガイド
- LDP: LDAP Linux HOWTO (`http://www.tldp.org/HOWTO/LDAP-HOWTO/index.html`)
- LDP: LDAP Implementation HOWTO (`http://www.tldp.org/HOWTO/LDAP-Implementation-HOWTO/index.html`)
- OpenLDAP, extensive use reports (`http://portal.aphroland.org/~aphro/ldap-docs/ldap.html`)

- Open LDAP with Courier IMAP and Postfix (`http://alinux.washcoll.edu/docs/plc/postfix-courier-howto.html`)

9.3 CD ライター

ATAPI/IDE インターフェースを用いた CD ライターは最近非常にポピュラーな 選択肢になっています。640 MB 以下の容量しか必要としないホームユーザ にとってのシステムバックアップとアーカイブのための素晴らしい媒体です。 CD ライターに関する最も信頼できる情報は、LDP CD-Writing-HOWTO (`http://www.tldp.org/HOWTO/CD-Writing-HOWTO.html`) をご覧ください。

9.3.1 導入

まず、CD ライターに送るデータが中断すると CD に回復できない被害が発生します。 できるだけ大きなバッファを CD ライターに与えて下さい。 お金を問わないならば、ATAPI/IDE に拘らず、SCSI を使ってください。 接続する IDE インターフェースを選択する場合、ISA バス上 (SB16 カード など) にあるものよりも PCI バス上 (マザーボードにあるものなど) にあるものを 使ってください。

CD ライターが IDE に接続されていると、Linux 2.2 kernel と 2.4 kernel では 通常の IDE CD ドライバの代わりに IDE-SCSI ドライバが働きます。 又、IDE-SCCI ドライバを有効にするには SCSI generic ドライバが必要となります。 最新のディストリビューション (2001 年 3 月現在)により配布されている kernel の使用を仮定すると、CD を焼くためには 2つの可能なアプローチが存在します。

Linux 2.6 カーネルの場合には、通常の IDE ドライバーを使い CD-RW デバイスに `/dev/hdx` と言ったデバイス名で直接アクセスするべきです。 こうすれば DMA が使えます。

9.3.2 アプローチ 1: モジュール + `lilo`

Debian パッケージの kernel を使っている場合は、`/etc/lilo.conf` に次の行を追加してください。複数のオプションを使う場合、スペースでそれらを 区切ってください。

```
append="hdx=ide-scsi ignore=hdx"
```

ここで CD ライターの場所は、ide-scsi ドライバを通してアクセスする場合、 `hdx` として示されます。ここで *x* は、次のいずれかを表します。

hda	プライマリ IDE ポートのマスター
hdb	プライマリ IDE ポートのスレーブ
hdc	セカンダリ IDE ポートのマスター
hdd	セカンダリ IDE ポートのスレーブ
hde から hdh	外部 IDE ポート又は ATA66/100 IDE ポートに接続されたドライブ

root で次のコマンドをタイプすることにより、全設定が有効になります。

```
# lilo
# shutdown -h now
```

9.3.3 アプローチ 2: kernel 再構築

Debian は kernel を作るために `make-kpkg` を用います。 複数の kernel イメージを構築するため、`make-kpkg` の 新しい `--append_to_version` オプションを用います。 'Debian での Linux kernel' on page 99 をご覧ください。

`make menuconfig` を通じて次のセットアップを使用してください。

- bzImage
- IDE CD ドライバを除く (必須ではないがシンプルになります)
- ide-scsi と sg を組み込むかモジュールとしてコンパイルする

9.3.4 設定後のステップ

次を実行することによりブート中に kernel の CD ライターのサポートを有効に できます。

```
# echo ide-scsi >>/etc/modules
# echo sg       >>/etc/modules
# cd /dev; ln -sf scd0 cdrom
```

手動により CD ライターを有効にするには次を実行します。

```
# modprobe ide-scsi
# modprobe sg
```

リブート後、このように CD ライターのインストールをチェックできます。

```
$ dmesg|less
# apt-get install cdrecord
# cdrecord -scanbus
```

[Warren Dodge による] CD-ROM と CD-R/RW がシステムに同時に存在する場合、時々 `ide-scsi` と `ide-cd` の間で衝突が発生することがありえます。 `/etc/modutils/aliaese` に次の行を追加して、 `update-modules` を起動してからリブートしてみてください。

```
pre-install      ide-scsi      modprobe ide-cd
```

この設定は IDE ドライバを `ide-scsi` の前にロードさせます。 IDE ドライバ `ide-cd` は ATAPI CD-ROM を制御します。 — これは `ide-scsi` が制御するデバイスを無視するだけです。

9.3.5 CD イメージファイル (ブータブル)

`target-directory/` にあるファイルの CD イメージを `cd-image.raw` として作成するには、フロッピドライブにブートフロッピを挿入して次を 実行します。 (ブータブルな Joilet TRANS.TBL が有効化されたフォーマット。 ブート不能な場合は、`-b` と `-c` オプションを取る)

```
# dd if=/dev/fd0 target-directory/boot.img
# mkisofs -r -V volume_id -b boot.img -c bootcatalog -J -T \
        -o cd-image.raw target_directory/
```

ブータブル DOS CD-ROM を作るのは面白いハックです。通常の DOS ブートフロッピ ディスクイメージは *`boot.img`* にあり、CD-ROM は DOS フロッピが A ドライブにあるかのようにブートします。これを freeDOS を使って行うとより面白いかもしれません。

この CD イメージはループバックデバイスによりマウントして検査できます。

```
# mount -t iso9660 -o ro,loop cd-image.raw /cdrom
# cd /cdrom
# mc
# umount /cdrom
```

9.3.6 CD ライター (R,RW) に書き込む

まず次のようにテストを行います。 (倍速と仮定)

```
# nice --10 cdrecord -dummy speed=2 dev=0,0 disk.img
```

結果が OK ならば、CD-R に次のように書き込みます。

```
# nice --10 cdrecord -v -eject speed=2 dev=0,0 disk.img
```

又は CD-RW ディスクに次のように書き込みます。

```
# nice --10 cdrecord -v -eject blank=fast speed=2 dev=0,0 disk.img
```

次のようにした方が良い CD-RW ドライブもあります。

```
# nice --10 cdrecord -v blank=all speed=2 dev=0,0 disk.img
```

に続いて

```
# nice --10 cdrecord -v -eject speed=2 dev=0,0 disk.img
```

を実行します。 この 2 ステップは消去作業中に、書き込み作業を妨害するのを防ぐために必要となります。`nice` の引数は幾らか調整が必要となる かもしれません。

9.3.7 CD イメージファイルの作成

ある CD-R や商用 CD は `dd` によるコピーができなくなるように終わりに壊れたセクタを持っています。(Windows 98 CD はこの 1 つです) `cdrecord` パッケージには `readcd` コマンドが付随しています。このコマンドを使うと CD の内容をコピーして イメージファイルを作成できます。データディスクの場合、マウントして `df` を実行することにより実際のサイズがわかります。そのブロック数を 2 で割って実際の CD セクター数がわかります。(1セクタ = 2048 byte) そして `readcd` を次に示すオプション付きで実行し、作成されたディスクイメージを使って CD-R/RW に書き込めます。

```
# readcd dev=target,lun,scsibusno # 機能 11 を選択
```

ここで、ほとんどの場合はこれら 3 つのパラメータは全て 0 にセットします。通常 `readcd` により与えられるセクタ数は過大です! 実際にマウントして得られる数を使う方がより良い結果をもたらします。

`dd` を CD に対して使うといくつか問題が発生することを言及すべきです。`dd` コマンドを最初に実行すると、エラーが発生し、末尾が失われた短いディスクイメージができるかもしれません。次に `dd` コマンドを実行すると、ディスクサイズが指定されていない場合、後ろにごみデータが付随した大きなディスクイメージができるかもしれないシステムがあるかもしれません。引数として正しいディスクサイズを与えて `dd` コマンドを 2 回実行し、エラーメッセージの後に CD をイジェクトしなければこれらの問題を避けられるようです。例えば、`df` により表示されるイメージサイズが `46301184` であった場合、次のコマンドを 2 回実行すると正しいイメージを得られます。(これは経験的な情報です)

```
# dd if=/dev/cdrom of=cd.img bs=2048 count=$((46301184/2))
```

9.3.8 Debian CD イメージ

Debian CD の最新情報を得るには、Debian CD site (`http://www.debian.org/CD/`) を訪れてください。

高速なインターネット接続を持っている場合は、次を用いてネットワーク越しに インストールすることを考えてみてください。

- 数枚の floppy images (`http://www.debian.org/distrib/floppyinst`) 又は
- minimal bootable CD image (`http://www.debian.org/CD/netinst/`)。

高速なインターネット接続を持っていない場合は、CD vendor (`http://www.debian.org/CD/vendors/`) から CD を購入することを 考慮してください。

CD イメージのテスタで無い限り、(たとえ新しい jigdo を使ったとしても) 標準の CD イメージをダウンロードしてバンド幅を無駄に消費しないでください。

言及すべき CD イメージは KNOPPIX - Live Linux Filesystem On CD (`http://www.knopper.net/knoppix/index-en.html`) です。この CD はハードディスクにインストールせずに機能する Debian システムを ブートします。

9.3.9 システムを CD-R にバックアップ

主要な設定ファイルとデータファイルを CD-R にコピーするには、`backup`(`http://www.debian.org/doc/manuals/debian-reference/examples/`) にあるサンプルバックアップスクリプトを使います。又、'サブディレクトリ全体のコピーとアーカイブ' on page 108 と '差分バックアップとデータ同期' on page 110 もご覧下さい。

9.3.10 音楽 CD を CD-R にコピーする

テストしていません。

```
   # apt-get install cdrecord cdparanoia
   # cdparanoia -s -B
   # cdrecord dev=0,0,0 speed=2 -v -dao -eject defpregap=1 -audio *.wav
```

又は

```
   # apt-get install cdrdao #disk at once
   # cdrdao read-cd --device /dev/cdrom --paranoia-mode 3 my_cd # CD を
読み込む
   # cdrdao write --device /dev/cdrom --speed 8 my_cd    # 新         し
い CD を書き込む
```

`cdrdao` は (ギャップなどが無い) 真のコピーを作ります。

9.3.11 DVD-R や DVD-RW や DVD+RW への書き込み

DVD に書き込む場合, 2 つのアプローチがあります:

- `growisofs` を `mkisofs` とともに使う。
- `/usr/share/doc/cdrecord/README.DVD.Debian` に書かれた手順で `cdrecord` を dvd オプションでリコンパイルしローカルパッケージを作る。

9.4 X

W Window System は XFree86 (`http://www.xfree86.org/`) により供給されています。Debian システムでは 2つのメジャーバージョンの X サーバ、すなわち XFree86 Version 3.3 (XF3) と XFree86 Version 4.x シリーズ (XF4) が得られますが、どちらも X.Org (`http://www.x.org/`) による X11R6 規格に基づいています。

X の基礎は、`X(7)`、LDP XWindow-User-HOWTO (`http://www.tldp.org/HOWTO/XWindow-User-HOWTO.html`) や Remote X Apps mini-HOWTO (`http://www.tldp.org/HOWTO/mini/Remote-X-Apps.html`) を参照してください。Debian 特有のユー

ザガイドは、`xfree86-common` パッケージにより供給される `/usr/share/doc/xfree86-common/FAQ.gz` をご覧下さい。これには Branded Robinson による興味深く信頼が置けるキーバインディング問題の報告が含まれます。

'X サーバ' on page 143 ユーザのモニタ (CDT,液晶) に X window やデスクトップを 表示し、キーボードやマウスの入力を受け付けるローカルホストでのプログラム。

'X クライアント' on page 146 X 互換なアプリケーションソフトを起動する (ローカル又はリモートでの) プログラム

これは他の文脈での通常の "サーバ" と "クライアント" の逆の使いかたです。

"X クライアント" (アプリケーション側) から "X サーバ" (ディスプレイ側) がリモート接続を得るための複数の方法があります。

- `xhost`
  - ホストリストを用いた方法 (非常に危険)
  - 暗号化されていないプロトコル (盗聴されやすい)
  - できるならば使わないでください。
  - 'リモートの X サーバに接続する – `xhost`' on page 150 と `xhost(1x)` 参照。
- **`xauth`**
  - MIT magic cookie を使った方法 (安全ではないが `xhost` よりは良い)
  - 暗号化されていないプロトコル (盗聴されやすい)
  - `ssh -X` よりは CPU 負荷が少ないのでローカル接続に限り 使ってください。
  - 'X で root 権限を取得する' on page 153 と `xauth(1x)` 参照。
- `xdm`, `wdm`, `gdm`, `kdm` などの方法
  - MIT magic cookie を用いた方法 (`xauth` と同様に安全では無い)
  - X ディスプレイアクセス制御については `xdm(1x)` と `Xsecurity(7)` を参照。
  - インストールしている場合、詳細は `wdm(1x)`, `gdm(8)`, と `kdm.options(5)` を参照。
  - `xdm` パッケージをパージせずにブート時に Linux コンソール を得るために `xdm` を無効にする方法は 'ランレベルのカスタマイズ' on page 22 を参照。
- **`ssh -X`**
  - 安全なシェル (**ssh**) によるポートフォワードを用いる 方法
  - 暗号化されたプロトコル (ローカル接続の場合リソースの無駄)
  - リモート接続の場合はこれを使ってください。
  - 'リモートの X サーバに接続する – `ssh`' on page 151 参照。

`ssh` を除いて、全てのリモート接続は X サーバ上で可能な TCP/IP 接続を要求します。'TCP/IP 越しに X を使う' on page 150 をご覧下さい。

9.4.1 X パッケージ

Woody に X システムをインストールするのを簡単にするためにいくつかの (メタ)パッケージが供給されています。

x-window-system-core このメタパッケージは X Window System を起動するスタンドアロンの ワークステーションに不可欠な構成要素を供給します。X ライブラリ、X サーバ (`xserver-xfree86`)、フォント群、そして 基本的な X クライアントおよびユーティリティ群を供給します。

x-window-system このメタパッケージは XFree86 プロジェクトにより開発されている X Window System としての実質上全ての構成要素を供給するだけでなく、 歴史的に有名なアクセサリプログラムのセットを供給します。(とりわけ、`x-window-system-core`, `twm`, そして `xdm` などに依存します。これをインストールする場合 `x-window-system-core` をインストールする必要はありません。)

xserver-common-v3 XFree86 3.x X サーバ (XF3) に共通のファイルやユーティリティです。

xserver-* 何らかの理由で新しい XF4 サーバ (`xserver-xfree86`) によりサポートされていないハードウェアをサポートするための補助的な XF3 サーバパッケージです。いくつかの古い ATI mach64 カードは XF4 でサポートされていませんし、他のカードも Woody でのバージョンの XF4 ではひどくハングします。(取得可能なパッケージは、`apt-cache search xserver-|less` により判明します。 これら XF3 サーバの全ては `xserver-common-v3` に 依存します。)

ほとんどの場合、`x-window-system` パッケージをインストール すべきです。(コンソールでログインしたい場合は、"'boot 時に X を無効にさせてくれ!"' on page 105 で言及されているように `xdm` を無効にすることを忘れないで ください。)

9.4.2 X 用のハードウェア検出

X の設定段階でハードウェアの検出を有効にするには、X システムをインストールする前に次のパッケージをインストールしてください。

- `discover` – ハードウェア認識システム。
- `mdetect` – マウスデバイス自動検知ツール。
- `read-edid` – VESA PnP モニタ用のハードウェア 情報収集ツール。

9.4.3 X サーバ

X サーバに関する情報は `XFree86(1x)` をご覧下さい。

ローカルの端末から X サーバを起動するには、次を実行します。

```
$ startx -- :<display> vtXX
例えば、
$ startx -- :1 vt8 -bpp 16
... vt8 で 16bpp モードを用いて X サーバを起動し、localhost:1 に接続
```

-- の後に与えられた引数は X サーバのためのものです。

X サーバ起動スクリプトをカスタマイズするために `~/.xserverrc` スクリプトを使用している場合、本当の X サーバを必ず exec してください。 こうしないと X サーバの起動が遅くなり、exit することに注意してください。 例えば、

```
#!/bin/sh
exec /usr/bin/X11/X -dpi 100 -nolisten tcp
```

X サーバの設定 (version 4)

XF4 サーバの(再)設定を行うには、

```
# dpkg-reconfigure --priority=low xserver-common
# dpkg-reconfigure --priority=low xserver-xfree86
```

を実行して `/etc/X11/XF86Config-4` ファイルを生成し、 `dexconf` スクリプトを用いて X の設定を行います。

X サーバの設定 (version 3)

XF3 サーバの(再)設定を行うには、例えば、ATI mach64 の場合、

```
# dpkg-reconfigure --priority=low xserver-common-v3
# dpkg-reconfigure --priority=low xserver-mach64
```

を実行して `/etc/X11/XF86Config` ファイルを生成し、 `xf86config-v3` スクリプトを用いて X の設定を行います。

X サーバを手動で設定する

Woody の場合、`/etc/X11/XF86Config-4` ファイルにユーザの カスタマイズを追加します。 **設定ファイル中の次の行を決して編集しないでください。**

```
### BEGIN DEBCONF SECTION
[snip]
### END DEBCONF SECTION
```

その代わり、テキストの前にカスタマイズを追加してください。 例えば、ビデオデバイスをカスタマイズするには、ファイルの 先頭 に次のテキストに似た行を追加してください。

```
Section "Device"
  Identifier        "Custom Device"
  Driver            "ati"
  Option            "NoAccel"
EndSection

Section "Screen"
  Identifier  "Custom Screen"
  Device      "Custom Device"
  Monitor     "Generic Monitor"
  DefaultDepth 24
  Subsection "Display"
    Depth       8
    Modes       "1280x960" "1152x864" "1024x768" "800x600" "640x480"
  EndSubsection
  Subsection "Display"
    Depth       16
    Modes       "1280x960" "1152x864" "1024x768" "800x600" "640x480"
  EndSubsection
  Subsection "Display"
    Depth       24
    Modes       "1280x960" "1152x864" "1024x768" "800x600" "640x480"
  EndSubsection
EndSection

Section "ServerLayout"
    Identifier       "Custom"
    Screen           "Custom Screen"
    InputDevice      "Generic Keyboard" "CoreKeyboard"
    InputDevice      "Configured Mouse" "CorePointer"
EndSection
```

Sarge (この文書の作成時は `testing`) の場合、 アップグレードを通じてカスタマイズした設定を `/etc/X11/XF86Config` 保ちたい場合、次のコマンドを root として実行してください。

```
# cp /etc/X11/XF86Config-4 /etc/X11/XF86Config-4.custom
# md5sum /etc/X11/XF86Config-4 > /var/lib/xfree86/XF86Config-4.md5sum
# dpkg-reconfigure xserver-xfree86
```

フォントのアンチエイリアス を行うには、'X での TrueType フォント' on page 154 に記述されているように `/etc/X11/XF86Config-4` を編集する必要があります。

X の設定ファイルの他の部分もチェックしてみてください。ひどいモニタ設定はひどいフォントよりも頭痛をもたらします。ですのでリフレッシュレートが使っているモニタが発揮できるレートと同じ位高いことを確認してください。(85 Hz なら素晴らしい、75 Hz なら大丈夫、60Hz だと苦痛です)

9.4.4 X クライアント

X クライアントプログラムのほとんどは次のようなコマンドで起動されます。

```
client $ xterm -geometry 80x24+30+200 -fn 6x10 -display hostname:0 &
```

ここで、任意のコマンドライン引数は次を意味します。

- `-geometry` *WIDTH*x*HEIGHT*+*XOFF*+*YOFF*: 起動時のウィンドウのサイズと位置
- `-fn` *FONTNAME*: テキスト表示用のフォント *FONTNAME* は次を設定可能です。
  - a14: 通常サイズのフォント
  - a24: 大きなサイズのフォント
  - など (`xlsfont` でフォントが使えるかをチェックしてください。)
- `-display` *displayname*: 使用する X サーバ名 *displayname* は次のようになります。
  - *hostname*:*D.S* はホスト *hostname* のディスプレイ *D* 上の *S* スクリーンを意味します。このディスプレイ用の X サーバは UNIX domain socket `/tmp/.X11-unix/XD` を listen しています。(それゆえ *host* からしか読めません)
  - *:D.S* は *host*/unix:*D.S* と等価です。ここで *host* はローカルのホスト名です。

X クライアントプログラム (アプリケーション側) 用の標準の *displayname* は DISPLAY 環境変数によりセットできます。例えば、環境変数を設定するために X クライアントプログラムより先に次のコマンドのうちひとつを実行します。

```
$ export DISPLAY=:0
        # 標準、ローカルマシンの第一スクリーン
$ export DISPLAY=hostname.fulldomain.name:0.2
$ export DISPLAY=localhost:0
```

この設定は `~/.xinitrc` によりカスタマイズできます。例えば、

```
xrdb -load $HOME/.Xresources
xsetroot -solid gray &
xclock -g 50x50-0+0 -bw 0 &
xload -g 50x50-50+0 -bw 0 &
xterm -g 80x24+0+0 &
xterm -g 80x24+0-0 &
twm
```

'X セッションのカスタマイズ' on page 147 に記述したように、この設定は `startx` から実行された場合、`Xsession` の通常の実行結果を全て上書きします。 その代わり、`~/.xsession` を苦肉の策としてのみ使ってください。 `xsetroot(1x)`、`xset(1x)` と 'X リソース' on page 151 をご覧下さい。

9.4.5 X セッション

X セッション (X サーバ + X クライアント) は次のように起動できます。

- `startx`: Linux 文字端末から X サーバと X クライアントを 起動するための `xinit` のラッパースクリプトコマンド。 `~/.xinitrc` が存在しない場合、`/etc/X11/Xsession` が `/etc/X11/xinit/xinitrc` を通じて実行される。
- `xdm`, `gdm`, `kdm`, や `wdm`: X サーバと X クライアントを起動し、GUI スクリーン 経由でログインを制御するためのX ディスプレ イマネージャデーモン。 `/etc/X11/Xsession` が直接実行される。

コンソールは "'boot 時に X を無効にさせてくれ!'' on page 105 に記述されている方法で取得できます。

X セッションのカスタマイズ

標準の起動スクリプト `/etc/X11/Xsession` は `/etc/X11/Xsession.d/50xfree86-common_determine-startup` と `/etc/X11/Xsession.d/99xfree86-common_start` が組み合わさったもの です。

`/etc/X11/Xsession` の実行は `/etc/X11/Xsession.options` にいくらか影響を受け、本質的には 次の順序で最初に見付かったプログラムを `exec` コマンドで 実行することになります。

1 定義されている場合、`~/.xsession` か `~/.Xsession`
2 定義されている場合、 `/usr/bin/x-session-manager`
3 定義されている場合、`/usr/bin/x-window-manager`
4 定義されている場合、`/usr/bin/x-terminal-emulator`

これらのコマンドの正確な意味は 'Alternative コマンド' on page 96 に記述されている Debian alternative システムにより定義されています。 例えば、

```
# update-alternatives --config x-session-manager
又は
# update-alternatives --config x-window-manager
```

GNOME や KDE のセッションマネージャをインストールしたまま他の X ウィンドウマネージャを標準にするには、 `/etc/X11/Xsession.d/50xfree86-common_determine-startup` を `http://bugs.debian.org/168347` の 2 番目のバグレポートに添付されているもの に置き換え (これがすぐに含まれればいいのですが)、X セッションマネージャを禁じるために次のように `/etc/X11/Xsession.options` を編集します。

```
# /etc/X11/Xsession.options
#
# configuration options for /etc/X11/Xsession
# See Xsession.options(5) for an explanation of the available options.
# Default enabled
allow-failsafe
allow-user-resources
allow-user-xsession
use-ssh-agent
# Default disabled (enable them by uncommenting)
do-not-use-x-session-manager
#do-not-use-x-window-manager
```

上で述べた修正をシステムに行わないと、`gnome-session` と `kdebase` はこれらの X セッションマネージャを含むパッケージとなります。これらを削除すると X ウィンドウマネージャが標準になります。(Yack, any better idea?)

`/etc/X11/Xsession.options` に `allow-user-xsession` という行が有効になっている場合、`~/.xsession` や `~/.Xsession` を定義したユーザは `/etc/X11/Xsession` のアクションをカスタマイズできます。

お好みの X ウィンドウ/セッションマネージャを起動するために、`~/.xsession` ファイルの最後のコマンドは `exec` *`some-window/session-manager`* の形にすべきです。

`~/.xsession` スクリプトの良い例は `/usr/share/doc/xfree86-common/examples/xsession.gz` に置いておきます。

私はこれをウィンドウマネージャ、スクリーンへのアクセス、ユーザアカウント毎の 言語サポートの設定のために使っています。'ユーザ用に X セッションを開始する' on page 148, 'X で root 権限を取得する' on page 153, そして '多言語 X window システムの例' on page 171 をご覧下さい。

複数の X クライアントプログラムを同時に起動したい場合、'X クライアント' on page 146 の例を見て、それらを `~/.xinitrc` ではなく `~/.xsession` から起動してください。

ユーザが指定する追加的な X リソースは `~/.Xresources` に 保存できます。'X リソース' on page 151 をご覧下さい。

X においてユーザがカスタマイズしたキーマップとポインタデバイスのマッピングも ユーザの起動スクリプトで指定できます。'X における keymap とポインタボタンマッピング' on page 152 をご覧下さい。

ユーザ用に X セッションを開始する

'X セッションのカスタマイズ' on page 147 に記述されている原則に従い、ユーザ指定の X セッション/ウィンドウマネージャは、指示されたパッケージをインストールし、 次のように `~/.xsession` の最後に設定することにより有効化 できます。(私はシンプルなスタイルと高速性から `blackbox`/ `fluxbox` が好きです。)

- 標準の X セッションマネージャ
  - ‘Alternative コマンド’ on page 96 参照
  - `exec /usr/bin/x-session-manager`

- 標準の X ウィンドウマネージャ
  - ‘Alternative コマンド’ on page 96 参照
  - `exec /usr/bin/x-window-manager`

- GNOME セッションマネージャ (重い)
  - インストールするパッケージ: `gnome-session`
  - `exec /usr/bin/gnome-session`

- KDE セッションマネージャ (重い)
  - インストールするパッケージ: `kdebase` (又は KDE3 の場合 `kdebase3`)
  - `exec /usr/bin/kde2`

- Blackbox ウィンドウマネージャ (軽量でしなやか)
  - インストールするパッケージ: `blackbox`
  - `exec /usr/bin/blackbox`

- Fluxbox ウィンドウマネージャ (軽量、新しい blackbox)
  - インストールするパッケージ: `fluxbox`
  - `exec /usr/bin/fluxbox`

- Xfce ウィンドウマネージャ (Mac OS-X や SUN CDE ライク)
  - インストールするパッケージ: `xfce`
  - `exec /usr/bin/xfwm`

- IceWM ウィンドウマネージャ (軽量、GNOME の代替品)
  - インストールするパッケージ: `icewm`
  - `exec /usr/bin/X11/icewm`

- FVWM2 仮想ウィンドウマネージャ (軽量、Win95 ライク)
  - インストールするパッケージ: `fvwm`
  - `exec /usr/bin/fvwm2`

- Windowmaker ウィンドウマネージャ (幾らか NexT ライク)
  - インストールするパッケージ: `wmaker`
  - `exec /usr/bin/wmaker`

- Enlightenment ウィンドウマネージャ (重い)
  - インストールするパッケージ: `enlightenment`
  - `exec /usr/bin/enlightenment`

Window Managers for X (`http://www.xwinman.org`) をご覧下さい。

KDE と GNOME の設定

KDE 又は GNOME 環境を完全に設定するためには、次のメタパッケージが役立ちます。

- KDE: `kde` パッケージをインストール
- GNOME: `gnome` パッケージをインストール

これらのパッケージを `Recommends` を操作できる `dselect` や `aptitude` のようなツールでインストールすると、ただ `apt-get` でインストールするよりも適切なソフトウェアの選択を行えます。

端末からログインをしたい場合、`kdm`、`gdm` のような X ディスプレイマネージャを必ず無効にしてください。これらは "'boot 時に X を無効にさせてくれ!'" on page 105 に記述されているように、メタパッケージといっしょに依存性に従いインストールされるかもしれません。

KDE ではなく、GNOME をシステムの標準にしたい場合、'Alternative コマンド' on page 96 にあるように `x-session-manager` の 設定を必ず行ってください。

9.4.6 TCP/IP 越しに X を使う

暗号化を行わずにリモートの TCP/IP ソケット接続を行うのは盗聴攻撃を受けやすいので、最近の Debian における標準の X の設定は TCP/IP ソケットを無効にしています。リモートから X へ接続するには、 `ssh` の使用を考慮してください。('リモートの X サーバに接続する – `ssh`' on page 151 参照)

ここで記述する方法は信頼できるユーザしか存在しない良質のファイアーウォールシステムの後ろに隠れた非常に安全な環境にいるので無ければ推奨されません。現在の TCP/IP ソケットに関する X の設定を検証するために、次のコマンドを 使用してください。

```
# find /etc/X11 -type f -print0 | xargs -0 grep nolisten
/etc/X11/xinit/xserverrc:exec /usr/bin/X11/X -dpi 100 -nolisten tcp
```

X サーバが TCP/IP を listen されるには、`-nolisten` を削除してください。

9.4.7 リモートの X サーバに接続する – xhost

`xhost` を使ってホスト名に基づいてリモートホストにアクセスできる ようになります。これは非常に危険です。次の手順によりホストのチェックを 無効にし、TCP/IP ソケット接続が許されている所からならどこからでも 接続を許すようになります。('TCP/IP 越しに X を使う' on page 150 参照)

```
$ xhost +
```

ホストのチェックを次の手順で再び有効にできます

```
$ xhost -
```

`xhost` はリモートホストの異なるユーザを区別できません。 又、ホスト名 (実際はアドレス) はだまされ得ます。

この方法は信頼できないネットワーク (例えばインターネットへダイアルアップ PPP 接続している場合など) では、より制限されたホスト基準を用いていたとしても避けるべきです。`xhost(1x)` をご覧下さい。

9.4.8 リモートの X サーバに接続する – `ssh`

`ssh` を使うと、ローカルの X サーバからリモートのアプリケーション サーバに安全な接続を行うことが可能になります。

- コマンドラインオプションを省略したい場合、リモートホストの `/etc/ssh/sshd_config` にある `X11Forwarding` および `AllowTcpForwarding` エントリを `yes` にセットしてください。
- ローカルホストで X サーバを起動する。
- ローカルホストで `xterm` を開く。
- `ssh` を起動してリモートサイトへの接続を確立する。

```
localname @ localhost $ ssh -q -X -l loginname remotehost.domain
Password:
.....
```

- リモートサイトで X アプリケーションを起動する。

```
loginname @ remotehost $ gimp &
```

この方法はローカルの UNIX ドメインソケットを通してローカルで接続しているかのようにリモートの X クライアントの出力を表示できます。

9.4.9 X ターミナルエミュレータ – `xterm`

`http://dickey.his.com/xterm/xterm.faq.html` で `xterm` についての全てを学んでください。

9.4.10 X リソース

`xterm` のような多くの古い X プログラムは、見栄えを設定するために X リソースデータベースを用います。 ユーザのリソース指定を保存するため、`~/.Xresources` ファイルを用います。このファイルはログイン時に標準の X リソースと自動的にマージされます。システムの標準となっている X リソースは `/etc/X11/Xresources/*` に、アプリケーション

の標準リソースは `/etc/X11/app-defaults/*` に保存されます。 これらの設定を開始点として使用します。

`~/.Xresources` ファイルに追加すべき役立つ設定をいくつか示します。

```
! フォントを読みやすい 9x15 にセット
XTerm*font: 9x15

! スクロールバーを表示
XTerm*scrollBar: true

! バッファサイズを 1000 行にセット
XTerm*saveLines: 1000

! 大きな kterm スクリーン
KTerm*VT100*fontList: -*-fixed-medium-r-normal--24-*,\
 -*-gothic-medium-r-normal--24-*,\
 -*-mincho-medium-r-normal--24-*
```

これらの設定を直ちに有効にするには、次のコマンドを使ってデータベースに マージします。

```
xrdb -merge ~/.Xresources
```

`xrdb(1x)` をご覧下さい。

9.4.11 X における keymap とポインタボタンマッピング

`xmodmap` プログラムは X クライアントアプリケーション上で イベントのキーコードをキーシムに変換するために用いるキーボード modifier マップと keymap テーブルを編集したり表示するために用いられます。

```
$ xmodmap -pm
 ... 現在の modifier マップを表示
$ xmodmap -pk | pager
 ... 現在の keymap テーブルを表示
$ xmodmap -e "pointer = 3 2 1" # 左利き用マウスに設定
$ xmodmap ~/.xmodmaprc # キーボードのマッピングを ~/.xmodmaprc で設定
```

これは通常ユーザのセッション起動スクリプトである `~/.xsession` から起動されます。

`keycode` を取得するには、X 上で `xev` を起動し、 keycode を取得したいキーを押します。`keysym` の意味を知るには、 `/usr/include/X11/keysymdef.h` ファイルにある MACRO 定義を 調べます。このファイルにある全ての `#define` 文は `XK_` として指定されており、それぞれが `keysym` 名と対応しています。

`xmodmap(1x)` をご覧下さい。

9.4.12 X で root 権限を取得する

GUI プログラムを root 権限で起動する必要がある場合、次の手順に従ってプログラムの出力をユーザの X サーバに表示してください。 セキュリティの問題を可能な限り避けるため、**決して X サーバを直接 root から起動しようとしてはいけません**。

X サーバを通常のユーザとして起動し、`xterm` 端末を開きます。 そして次を実行します。

```
$ XAUTHORITY=$HOME/.Xauthority
$ export XAUTHORITY
$ su root
Password:*****
# printtool &
```

このトリックを `su` で非 root ユーザに変更するために使う場合、 `~/.Xauthority` がその非 root ユーザが属するグループが読み込み可能であることを確認してください。

この作業を自動化するには、ユーザアカウントの `~/.xsession` に次の行を追加します。

```
# root アカウント に su した時に X が動くようにする。
if [ -z "$XAUTHORITY" ]; then
        XAUTHORITY=$HOME/.Xauthority
        export XAUTHORITY
fi
unset XSTARTUP
# 特定のウィンドウ/セッションが必要な場合、次の行のコメントを外して
# 必要に応じて編集する。
#XSTARTUP=/usr/bin/blackbox
# x-window/session-manager プログラムを起動
if [ -z "$XSTARTUP" ]; then
  if [ -x /usr/bin/x-session-manager ]; then
    XSTARTUP=x-session-manager
  elif [ -x /usr/bin/x-window-manager ]; then
    XSTARTUP=x-window-manager
  elif [ -x /usr/bin/x-terminal-emulator ]; then
    XSTARTUP=x-terminal-emulator
  fi
fi
# X window/session manager を自動実行
exec $XSTARTUP
```

そして (`su -` ではなく) `su` を ユーザの `xterm` ウィンドウ上で起動します。するとこの `xterm` から起動する GUI プログラムは root 権限で動きますが出力をユーザの X window 上に表示できます。 このトリックは標準の `/etc/X11/Xsession` が実行される限り動きます。ユーザが `~/.xinitrc` や `~/.xsession` をカスタマイズした場合、上で言及した環境変数 `XAUTHORITY` がこれらのスクリプトと同じように設定される必要があります。

代わりに、`sudo` を上記の手順を自動化するために 使用できます。

```
$ sudo xterm
... 又は
$ sudo -H -s
```

ここで、`/root/.bashrc` に次の行が含まれる必要があります。

```
if [ $SUDO_USER ]; then
    sudo -H -u $SUDO_USER xauth extract - $DISPLAY | xauth merge -
fi
```

これはユーザのホームディレクトリが NFS マウントされている場合でさえもうまく動きます。それは root が `.Xauthority` ファイルを読まないためです。

この目的のために用いられる専門プログラムとしては、kdesu, gksu, gksudo, gnome-sudo, そして xsu などが存在します。同様の結果を得るために、`/root/.Xauthority` のシンボリックリンクをユーザのホームディレクトリに作成することや、スクリプト sux (`http://fgouget.free.fr/sux/sux-readme.shtml`) を使う、root の初期化スクリプトに "`xauth merge ~`*`USER_RUNNING_X`*`/.Xauthority`" を追加するなどの他にいくつかの方法が使えます。

さらに知りたい場合は debian-devel mailing list (`http://lists.debian.org/debian-devel/2002/debian-devel-200207/msg00259.html`) をご覧下さい。

9.4.13 X での TrueType フォント

XFree86-4 での標準的な `xfs` は TrueType をうまく処理します。XFree86-3 を使っている場合は、`xfs-xtt` のようなサードパーティのフォントサーバをインストールする必要があります。

TrueType フォントを使いたいアプリケーションが libXft にリンクされているか、それとも libfreetype にリンクされているかを確認することだけが必要です。(前もってコンパイルされた .deb を使っている場合はこれさえも心配する必要はたぶん無いでしょう。)

まずフォントのサポートインフラを設定します。

- `x-ttcidfont-conf` および `defoma` パッケージをインストールする。これは `fonts.scale` と `fonts.dir` ファイルの自動生成を行います。

  ```
  # apt-get install x-ttcidfont-conf
  ```

- `/etc/X11/XF86Config-4` の `Section "Files"` を次のように編集します。

  ```
  Section "Files"
      FontPath  "/var/lib/defoma/x-ttcidfont-conf.d/dirs/TrueType"
      FontPath  "/usr/share/fonts/truetype"
      FontPath  "/usr/lib/X11/fonts/CID"
      FontPath  "/usr/lib/X11/fonts/Speedo"
      FontPath  "/usr/lib/X11/fonts/misc"
      FontPath  "/usr/lib/X11/fonts/cyrillic"
      FontPath  "/usr/lib/X11/fonts/100dpi:unscaled"
  ```

```
         FontPath  "/usr/lib/X11/fonts/75dpi:unscaled"
         FontPath  "/usr/lib/X11/fonts/Type1"
     EndSection
```

最初の行は、XFree86 が Debian パッケージからインストールした TrueType フォントを使えるように設定します。XFree86 が Type1 フォントのレンダリングをうまくできないので、Type1 フォントのエントリは格下げされています。ビットマップフォントに対する `:unscaled` のトリックは XF4 ではもはや必要ありませんが、残しています。`/etc/X11/XF86Config-4` の手動での変更を保存するには、'X サーバを手動で設定する' on page 144 にある指示に従ってください。

次に、DFSG フォントパッケージをインストールします。

- 欧米TrueType フォント:
  - `ttf-bitstream-vera`: Bitstream, Inc により作成された高品質な TrueType フォントセット。[2]
  - `ttf-freefont`: UCS 文字セットをカバーする TrueType フォント。
  - `ttf-thryomanes`: Latin, Greek, Cyrillic, および IPA をカバーする TrueType Unicode フォント。
- アジア圏のフォント:
  - `tfm-arphic-bsmi00lp`: Tex フォトメトリックデータ用の 中国 Arphic "AR PL Mingti2L Big5" TrueType フォント。
  - `tfm-arphic-bkai00mp`: Tex フォトメトリックデータ用の 中国 Arphic "AR PL KaitiM GB" TrueType フォント。
  - `tfm-arphic-gbsn00lp`: Tex フォトメトリックデータ用の 中国 Arphic "AR PL SungtiL GB" TrueType フォント。
  - `tfm-arphic-gkai00mp`: Tex フォトメトリックデータ用の 中国 Arphic "AR PL KaitiM GB" TrueType フォント。
  - `ttf-baekmuk`: 韓国の Baekmuk シリーズ TrueType フォント
  - `hbf-jfs56`: CJK 用の 中国 Jianti Fangsong 56x56 ビットマップフォント (GB2312)
  - `hbf-cns40-b5`: CJK 用の中国 Fanti Song 40x40 ビットマップフォント (Big5)
  - `hbf-kanji48`: CJK 用の日本語漢字 48x49 ビットマップ フォント

フリーな フォントは時々制限があるので、Debian ユーザ にとって商用の TrueType フォントをインストール又は共有することは 選択肢のひとつです。 この手順を容易にするためにいくつか便利なパッケージが作成されています。

- `ttf-commercial`
- `msttcorefonts (>1.1.0)` [3]

フリーな システムを非フリーなフォントで汚染することを 犠牲にするならば、本当に良い TrueType フォントの選択を行ってください。

Debian におけるこれらのフォントパッケージは努力無しで動き、 通常の "コア" フォントシステムを使用する全ての X プログラムに対して 使用できるようになるべきです。X プログラムには Xterm、Emacs や 他のほとんどの 非 KDE 、非 GNOME アプリケーションが含まれます。

ついに、`xfontsel` を起動して fndry メニュで TrueType フォントを選べば、"fmly" メニュ

---

[2] これは Woody では得られませんが、Sarge からインストールできます。

[3] Woody ではこのパッケージは Microsoft の website の変更により、2002年 8月の時点で動きません。Woody をインストールしていても Sarge のパッケージを使いましょう。

にたくさんの黒色の選択可能なエントリが見付けられるはずです。

(GTK 1.2 がアンチエイリアスフォントレンダリングを行えるようにハックされた libgdkxft0 を使っている) KDE 2.2 と GNOME 1.4 では、Xft1 を設定する必要があります。その上、Xft1 はかなり時代遅れになっており、基本的に GNOME 1.4 と KDE 2.2 でしか使われていません。 `/etc/X11/XftConfig` を編集し、

```
dir "/var/lib/defoma/x-ttcidfont-conf.d/dirs/TrueType"
```

のような行を他の dir 行の前に追加してください。

[4]

GNOME2 と KDE3 (Sarge リリース後) では、フォントを見付けるのに Xft2 を使っている `fontconfig` を設定する必要があります。 [5] `fontconfig` に依存するパッケージは既に (間接的に) このパッケージに Depends しているので、設定を行うのに何か特別なパッケージをインストールする必要はありません。

まず、`/etc/fonts/fonts.conf` を見ます。下に示すような行があるはずです。無い場合は、`/etc/fonts/local.conf` を開き、下の内容を <fontconfig> の行のすぐ後に追加します。

```
<dir>/var/lib/defoma/x-ttcidfont-conf.d/dirs/TrueType</dir>
```

`fontconfig` はこの変更に直ちに対応し、“fc-list” により新しいフォントが追加されるはずです。 `fontconfig` の他の巧妙な機能によって、フォントを `~/.fonts/` に入れるだけで fontconfig を使用するプログラムは直ちにこれらのフォントを使用可能になります。

Debian パッケージを使わずに X 上で新しい TrueType フォントセットを手動でインストールした場合、次を実行します。

```
# xset fp rehash
```

これにより XFree86 がこのディレクトリの内容を再び見て新しいフォントを取得するようになります。

9.4.14 X でのウェブブラウザ

Sarge リリース時点でグラフィカルに表示可能なウェブブラウザパッケージがいくつか存在します。
- `mozilla` The Mozilla ブラウザ
- `mozilla-firefox` Mozilla ブラウザの変種 (スタンドアロン)

---

[4] xft1 を他には使っていないので、この変更が効力を及ぼす前に X を再起動する必要があるかどうか分かりません。“xftcache” が Xft1 キャッシュを更新するはずだと記憶していますが、だれかが確認して私に知らせていただくとありがたい。

[5] `Fontconfig` は Woody には存在しません。

- `epiphany-browser` Mozilla ブラウザの変種 (Gnome)
- `konqueror` KDE ブラウザ
- `amaya` W3C リファレンスブラウザ
- などなど。

`testing` 又は `unstable` では、共有ライブラリのバージョンを 合わせる必要があるため、`mozilla` 変種のブラウザ間での バージョン不一致問題に直面するかもしれません。

`mozilla` のようなブラウザ用のプラグインは、“`*.so`” を手動でプラグインディレクトリにインストールし、ブラウザを 再起動することにより有効にできます。

プラグイン一覧
- Java プラグイン: `http://java.sun.com` からバイナリの “J2SE” を インストール。
- Flash プラグイン: `http://www.macromedia.com/software/flashplayer/` からバイナリの “Macromedia Flash Player 5” をインストール。
- `freewrl`: VRML ブラウザと Netscape プラグイン。
- など。

9.4.15 X 用メール クライアント (MUA)

Sarge リリース時点で、グラフィカルなディスプレ イ表示が可能なクライアントパッケージが数種類存在します:
- `mozilla-thunderbird` スタンドアロン メール クライアント
- `kmail` KDE メール クライアント
- `evolution` Novell によるグループウェアスイート
- など。

9.5 SSH

SSH (Secure SHell) はインターネットに接続するための安全な方法です。 OpenSSH と呼ばれるフリー版の SSH は Debian では `ssh` パッケージとして得られます。

9.5.1 SSH の基礎

まず OpenSSH サーバとクライアントをインストールします。

```
# apt-get update && apt-get install ssh
```

OpenSSH サーバを起動したい場合は `/etc/ssh/sshd_not_to_be_run` が存在する必要があります。

SSH には二つの認証プロトコルが存在します。

- SSH プロトコル version 1:

  - Potato はこのプロトコルしかサポートしていません。
  - 使用可能な認証方法
    - RSAAuthentication: RSA 認証キーベースのユーザ認証
    - RhostsAuthentication: .rhosts ベースのホスト認証 (危険であり無効にされている)
    - RhostsRSAAuthentication: RSA ホストキーと .rhosts 認証の組合せ (無効)
    - ChallengeResponseAuthentication: RSA challenge-response 認証
    - PasswordAuthentication: パスワードベースの認証

- SSH プロトコル version 2:
  - Woody 以降のバージョンはこれを主要なプロトコルとして使っている。
  - 使用可能な認証方法
    - PubkeyAuthentication: 公開鍵ベースのユーザ認証
    - HostbasedAuthentication: `.rhosts` 又は `/etc/hosts.equiv` による認証と公開鍵クライアントホスト認証の 組合せ (無効)
    - ChallengeResponseAuthentication: challenge-response 認証
    - PasswordAuthentication: パスワードベースの認証

Woody や他の非 Debian システムに移行しようする場合、これらのプロトコルの違い に注意してください。

詳細は `/usr/share/doc/ssh/README.Debian.gz` や `ssh(1)`, `sshd(8)`, `ssh-agent(1)`,並びに `ssh-keygen(1)` をご覧下さい。

主要な設定ファイルを次に挙げます。

- `/etc/ssh/ssh_config`: SSH クライアント標準 `ssh(1)` をご覧下さい。 注目に値すべきエントリ は次のとおり。
  - `Host`: キーワードの後にあたえたパターンの一つにマッチしたホスト のみに対して (次のホストキーワードまで) 宣言により制限する。
  - `Protocol`: SSH プロトコルバージョンを指定する。 標準は “2.1”。
  - `PreferredAuthentications`: SSH2 クライアント認証方法を指定する。 標準は “hostbased,publickey,keyboard-interactive,password”。
  - `PasswordAuthentication`: パスワードを使ってログインしたい場合、 これが `no` にセットされていないことを確認してください。
  - `ForwardX11`: 標準では無効。これはコマンドラインオプション `-X` により上書き可能です。

- `/etc/ssh/sshd_config`: SSH サーバの標準。 `sshd(8)` をご覧下さい。注目に値すべきエントリは 次のとおり。
  - `ListenAddress`: `sshd` が listen すべきローカル アドレスを指定。複数のオプションが許される。
  - `AllowTcpForwarding`: 標準では無効。
  - `X11Forwarding`: 標準では無効。

- `$HOME/.ssh/authorized_keys`: クライアントがこのホストで 本アカウントに接続するために使う標準の公開鍵のリスト。 `ssh-keygen(1)` をご覧下さい。

- `$HOME/.ssh/identity`: `ssh-add(1)` と`ssh-agent(1)` をご覧下さい。

次のコマンドは `ssh` 接続をクライアントから開始します。

```
$ ssh username@hostname.domain.ext
$ ssh -1 username@hostname.domain.ext # Force SSH version 1
$ ssh -1 -o RSAAuthentication=no -l username foo.host
    # SSH1 でパスワードを強制
$ ssh -o PreferredAuthentications=password -l username foo.host
    # SSH2 でパスワードを強制。
```

ユーザにとって、`ssh` は (^] を用いて爆撃しない) 賢くより安全な `telnet` として動きます。

9.5.2 SMTP/POP3 トンネル用のポートフォワード

`ssh` を通して localhost の port 4025 から *remote-server* の port 25 並びに localhost の port 4110 から *remote-server* の port 100 に接続 するパイプを構築するには、ローカルマシンで次を実行します。

```
# ssh -q -L 4025:remote-server:25 4110:remote-server:110 \
           username@remote-server
```

これはインターネットを通して SMPT/POP3 サーバに接続するための安全な手段です。リモートホストの `/etc/ssh/sshd_config` の `AllowTcpForwarding` エントリを `yes` にセットしてください。

9.5.3 パスワードの入力が少ない接続 – RSA

RSAAuthentication (SSH プロトコル) 又は PubkeyAuthentication (SSH2 プロトコル) を用いてリモートシステムごとのパスワードを覚えるのを避けられます。

リモートホストで、`/etc/ssh/sshd_config` の各エントリを “RSAAuthentication yes” 又は “PubkeyAuthentication yes” にセットします。

そしてローカルで認証キーを生成し、リモートシステムに公開鍵をインストールします。

```
$ ssh-keygen          # RSAAuthentication: SSH1 用の RSA 鍵
$ cat .ssh/identity.pub | ssh user1@remote \
        "cat - >>.ssh/authorized_keys"
...
$ ssh-keygen -t rsa   # PubkeyAuthentication: SSH2 用の RSA 鍵
$ cat .ssh/id_rsa.pub | ssh user1@remote \
        "cat - >>.ssh/authorized_keys"
```

```
...
$ ssh-keygen -t dsa   # PubkeyAuthentication: SSH2 用の DSA 鍵
$ cat .ssh/id_dsa.pub | ssh user1@remote \
        "cat - >>.ssh/authorized_keys"
```

“`ssh-keygen -p`” を用いて後にパスフレーズを変更できます。 接続できるかテストして必ず設定を検証してください。 問題がある場合は、“`ssh -v`” を使ってください。

ホスト制限や特定のコマンドを起動するために `authorized_keys` に あるエントリにオプションを追加できます。詳細は `sshd(8)` をご覧下さい。

SSH2 には `HostbasedAuthentication` 機能があることに注意してください。 これを動かすには、サーバマシンの `/etc/ssh/sshd_config` と クライアントマシンの `/etc/ssh/ssh_config` 又は `$HOME/.ssh/config` の両方を yes にセットする 必要があります。

9.5.4 他の SSH クライアントの扱い

非 Unix ライクプラットフォームのためのフリーな SSH クライアントは あまり存在しません。

Windows puTTY (`http://www.chiark.greenend.org.uk/~sgtatham/putty/`) (GPL)

Windows (cygwin) cygwin (`http://www.cygwin.com/`) (GPL) にある SSH

Macintosh Classic macSSH (`http://www.macssh.com/`) (GPL) [Mac OS X には OpenSSH が含まれていることに注意してください。Terminal アプリケーション上で `ssh` を使ってください。]

SourceForge.net, site documentation (`http://www.sourceforge.net/docman/?group_id=1`) の “6. CVS Instructions” もご覧下さい。

9.5.5 `ssh-agent` のセットアップ

パスフレーズで SSH 認証鍵を保護する方がより安全です。パスフレーズが セットされていなかった場合、`ssh-keygen -p` を使ってセットしてください。

リモートホストの `~/.ssh/authorized_keys` に パスワードを用いてリモートホストに接続して ‘パスワードの入力が少ない接続 – RSA’ on page 159 に記述されているように公開鍵 (すなわち、`~/.ssh/id_rsa.pub`) を置いてください。

```
$ ssh-agent bash # 又は代わりに zsh/tcsh/pdksh プログラムを起動する。
$ ssh-add ~/.ssh/id_rsa
Enter passphrase for /home/osamu/.ssh/id_rsa:
Identity added: /home/osamu/.ssh/id_rsa (/home/osamu/.ssh/id_rsa)
```

```
$ scp foo user@remote.host:foo
 ... ここからはパスフレーズが必要 :-)
$^D
 ... ssh-agent セッションを終了。
```

X サーバの場合、通常 Debian の起動スクリプトは `ssh-agent` を親プロセスとして実行します。ですので `ssh-add` を一度だけしか実行する必要がありません。

より詳しくは、`ssh-agent(1)` と `ssh-add(1)` をご覧下さい。

9.5.6 SSH のトラブルシュート

問題が発生した場合、設定ファイルのアクセス権をチェックし、`ssh` を “`-v`” オプションつきで実行してください。

root であってファイアーウォールのトラブルが発生した場合、“`-P`” オプションを使ってください。これはサーバのポート 1 から 1023 の使用を避けます。

リモートのサイトへの ssh 接続が突然動かなくなった場合、それはシステム管理者のやっつけ仕事のせいかもしれませんし、最もありえるのはシステムメンテナンス中に `host_key` が変更されたためです。このような原因で接続が停止し、だれもハックしてリモートホストを偽造しようとしていないことを確認してから、`host_key` エントリをローカルマシンの `$HOME/.ssh/known_hosts` から削除して接続を回復できます。

9.6 メール

メールの設定は三つのカテゴリに分割されます。

- mail transfer agent (MTA): `exim4`, `postfix`, `sendmail`, `qmail`, `ssmtp`, `nullmailer` など
- mail utilities: `procmail`, `fetchmail`, `mailx`, `crm114`, など
- mail user agent (MUA): `mutt`, `emacs+gnus` など

9.6.1 Mail transport agents (MTAs)

フル機能の MTA としては、Woody では `exim`を、Sarge では `exim4` を使っています。[6] リファレンス:

- `exim` 用の `exim-doc` と `exim-doc-html` パッケージ。
- `exim4` 用の `exim4-doc-info` と `exim-doc-html` パッケージ。
- `http://www.exim.org/`

---

[6] 次の章では `exim` を例として使っています。Sarge の場合は必要に応じて `exim4` に置き換えてください。

セキュリティを心配している場合の唯一の妥当な代替 MTA は、`postfix` です。`sendmail` と `qmail` も Debian パッケージとして得られますが、お薦めできません。

ノート PC などのサテライトシステムであって MTA のリレー機能が必要ない場合、次に挙げる軽量なパッケージのどれかの使用を考えても良いかもしれません。

- `ssmtp`: SMTP 接続とエイリアス機能が必要な場合に良い。又は
- `nullmailer`: スプール可能だがエイリアス機能は無い。

現時点では、私はノート PC をパーソナルワークステーションとして使う場合でも、`exim` が最適と考えています。

これらの衝突するパッケージをインストールする場合、`exim` を削除する必要があるかもしれません。

```
# dpkg -P --force-depends exim
# apt-get install nullmailer         # 又は ssmtp
```

Smarthost

消費者レベルのサービスの接続をしているホスト上で `exim4` 又は `exim` を起動している場合、ISP や他の手段で提供されている smarthost を通じてメールを送るようにしてください。[7] いくつかの理由があります。

- 通常 ISP の smarthost はより信頼できる接続環境を持っているので SMTP リトライを確実にできます。
- 多分ダイアルアップの spam リストによりブロックされるであろう **ダイナミック IP アドレス** からメールを直接送るのを避けられます。
- 複数の送信者にメールを送るためのローカルのバンド幅を節約できます。

想像できる唯一の例外としては次があげられます。

- ISP の SMTP サービスのトラブルの際の緊急手段。
- 教育目的の実験台として。
- 営業目的のホストサーバにするため。

Exim の基本的な設定

`exim4` 又は `exim` を MTA として使うには、次の設定を行います。

```
   /etc/exim/exim.conf      "eximconfig" を実行し、編集 (exim)
   /etc/exim4/*             "dpkg-reconfigure exim4-config" を実行し、編
集 (exim4)
   /etc/inetd.conf          exim をデーモンとして起動するために smtp をコメ
ントアウトする
   /etc/email-addresses     外に出すメールのためのドメインリストを追加
```

---

[7] ブロードバンドルータ経由でダイアルアップ, DSL, ケーブルサービス又は LAN 接続を行っているホストに対してこのルールに従う必要があります。ホストが固定 IP を持っている場合でも、このルールに従うのは依然良い考えです。ほとんどのワークステーションやホームサーバはこのカテゴリに属します。

`exim4` 又は `exim` を `-brw, -bf, -bF,` などの オプション付きで用いてフィルタをチェックしてください。

Exim の元で存在しないメールアドレスのための物置を設定する

`/etc/exim/exim.conf` (Woody 以降) の DIRECTORS パートにおいて、 末尾 (localuser: director の後) にその前の director が解決できない 全てのアドレスにマッチするごみ箱 director を追加してください。 (Miquel van Smoorenburg による)

```
catchall:
  driver = smartuser
    new_address = webmaster@mydomain.com
```

各仮想ドメインなどに対するより詳しいレシピを設定したい場合、 `/etc/exim/exim.conf` の末尾に次の内容を追加してください。 (あまりテストしていません)

```
*@yourdomain.com ${lookup{$1}lsearch*{/etc/email-addresses} \
        {$value}fail} T
```

そして、`/etc/email-addresses` に "*" エントリを追加してください。

Exim の元で送信メールに対してアドレス書き換えを設定する

適切な "From:" ヘッダを生成するために送信メールのアドレス書き換えを行う には、`exim` を用いて `/etc/exim/exim.conf` の末尾 近くに次のような設定を行うことにより行えます。

```
*@host1.something.dyndns.org \
  "${if eq {${lookup{$1}lsearch{/etc/passwd}{1}{0}}} {1}  \
   {$0}{$1@something.dyndns.org}}"  frFs
```

これは `*@host1.something.dyndns.org` にマッチする全アドレスを 書き換えます。

1. local part ($1) がローカルユーザかどうかを `/etc/password` を通じて検索する。
2. ローカルユーザである場合、アドレスを最初の場所 ($0) にあったものに 書き換えます。
3. ローカルユーザでない場合は、ドメイン部分を書き換えます。

Exim の元で SMTP 認証を設定する。

yahoo.com のような SMTP サーバは SMTP 認証を要求します。 `/etc/exim/exim.conf` を次のように設定します。

```
remote_smtp:
  driver = smtp
  authenticate_hosts = smtp.mail.yahoo.com
...

smarthost:
  driver = domainlist
  transport = remote_smtp
  route_list = "* smtp.mail.yahoo.com bydns_a"
...

plain:
  driver = plaintext
  public_name = PLAIN
  client_send = "^cmatheson3^this_is_my_password"
```

最後の行をダブルクォートするのを忘れないでください。

9.6.2 メールを取得する – Fetchmail

`fetchmail` はデーモンモードで起動され、メールを ISP の POP3 アカウントから取得し、ローカルのメールシステムに転送します。 設定ファイルは次のとおり。

`/etc/init.d/fetchmail`	
`/etc/rc?.d/???fetchmail`	標準では優先度 30 で `update-rc.d fetchmail` を起動する
`/etc/fetchmailrc`	設定ファイル (chmod600 のアクセス権限で `fetchmail` により所有される)

Potato で `init.d` スクリプトから `fetchmail` をデーモンモードで起動する方法に関する情報は混乱しています。 (Woody では既に修正されています) example scripts (`http://www.debian.org/doc/manuals/debian-reference/examples/`) にある `/etc/init.d/fetchmail` と `/etc/fetchmailrc` のサンプルをご覧下さい。

メールのヘッダが ISP のメーラのせいで ^M で汚染されている場合、 `$HOME/.fetchmailrc` に “stripcr” をオプションとして次のように追加してください。

```
options fetchall no keep stripcr
```

9.6.3 `crm114` でのスパム処理

`crm114` パッケージは CRM114 で書かれた `/usr/share/crm114/mailfilter.crm` スクリプト を提供します。このスクリプトはスパム(不要メール)とハム(有用メール) を流し込むと非常に効率的に訓練できるスパムフィルターを提供します。

CRM114 はフィルターを書くために設計された小さな言語です。 スーパーパワーを与えた grep のような物と考えて下さい。 `crm(1)` を参照してください。

9.6.4 メール処理 – Procmail

`procmail` はローカルのメール配信およびフィルタプログラムです。 本プログラムを使う際にはアカウント毎に `$HOME/.procmailrc` を作成する必要があります。例は _procmailrc (`http://www.debian.org/doc/manuals/debian-reference/examples/`) にあります。

9.6.5 メールを読む – Mutt

mail user agent (MUA) として `mutt` を `vim` と 組み合わせて使ってください。 `~/.muttrc` でカスタマイズできます。例えば、

```
# use visual mode and "gq" to reformat quotes
set editor="vim -c ’set tw=72 et ft=mail’"
#
# header weeding taken from the manual (Sven’s Draconian header weeding)
#
ignore *
unignore from: date subject to cc
unignore user-agent x-mailer
hdr_order from subject to cc date user-agent x-mailer
auto_view application/msword
....
```

HTML メールと MS Word 添付ファイルをインラインで表示するため、次を `/etc/mailcap` 又は `$HOME/.mailcap` に追加してください。

```
text/html; lynx -force_html %s; needsterminal;
application/msword; /usr/bin/antiword ’%s’; copiousoutput;
description="Microsoft Word Text"; nametemplate=%s.doc
```

9.7 ローカライゼーション (l10n)

Debian は国際化されており、言語や地域差に関するサポートが充実しつつあります。 次のサブセクションは Debian が現在サポートする多様性の形態の一部をリストし、 それに続く

サブセクションにおいて地域に従って異なる日付表示、数字や金銭の フォーマットなどの円滑な入出力が可能となるように使用環境を カスタマイズするプロセスである **ローカライゼーション** について議論します。

9.7.1 ローカライゼーションの基本

ローカライゼーションと各国語サポートにはいくつかの観点があります。

キーボードのローカライゼーション

Debian は24程のキーマップが配布されています。 Woody でのキーボードを再設定するには、次を実行します。

- `dpkg-reconfigure --priority=low console-data # 端末`
- `dpkg-reconfigure --priority=low xserver-xfree86 # XF4`
- `dpkg-reconfigure --priority=low xserver-common-v3 # XF3`

データファイルのローカライゼーション

膨大な Debian ソフトウェアパッケージの大部分は glibc に存在する **ロケール** 技術 により提供される LC_CTYPE 環境変数 を通して非 US-ASCII 文字データの操作をサポートしています。

- 8-bit クリーン: 実際には全プログラムがサポート
- 他の Latin 文字セット (例えば ISO-8859-1 や ISO-8859-2): ほとんどの プログラムがサポート
- 中国語、日本語、韓国語のようなマルチバイト言語: 多くの新しめのアプリケーションがサポート

表示のローカライゼーション

X は UTF-8 を含むあらゆるコーディングを表示できますし、全てのフォントを サポートします。全ての 8 ビットフォントだけでなく、中国語、日本語や韓国語 のような 16 ビットフォントも含まれます。マルチバイト文字の入力メソッドは 'X での代替入力メソッド' on page 174 メカニズムによりサポートされています。'多言語 X window システムの例' on page 171 と 'X 端末エミュレータの UTF-8 サポート' on page 175 をご覧下さい。

日本語 EUC コードは `kon2` パッケージを通して (S)VGA グラフィックコンソールにも表示できます。`jfbterm` という新しい代わりになる日本語ディスプレ イがありますが、これはフレームバッファ 端末も使っています。これらの端末環境では、日本語の入力メソッドが アプリケーションによりサポートされている必要があります。 Emacs で日本語を入力するには、`egg` パッケージを、 Vim 環境で日本語を入力するには、日本語化された `jvim` パッケージを使ってください。

非 Unicode フォントを X にインストールすると、X においてあらゆる エンコーディングのドキュメントを表示するのに役立ちます。ですから、 フォントのエンコーディングについてあまり心配する必要はありません。

メッセージやドキュメンテーションのローカライゼーション

エラーメッセージ、標準的なプログラムの出力、メニュ、そしてマニュアルページ などの Debian システムで表示されるテキストメッセージやドキュメントの多くは 既に翻訳されています。現在、ドイツ語、スペイン語、フィン語、フランス語、 ハンガリー語、イタリア語、日本語、韓国語、ポーランド語、ポルトガル語、 中国語、そしてロシア語のマニュアルページが `manpages-`*LANG* パッケージ (ここで *LANG* は 2 文字のコンマで分けられた ISO country code のリストです。取得可能な Unix マニュアルページのリストを得る には `apt-cache search manpages-|less` を使ってください。) を通じて供給されています。

NLS マニュアルページにアクセスするには、ユーザは環境変数 LC_MESSAGES を 適切な文字列にセットする必要があります。例えば、イタリア語のマニュアルページ の場合は、LC_MESSAGES を `it` にセットする必要があります。 すると `man` プログラムは `/usr/share/man/it/` の下にある イタリア語のマニュアルページを検索します。

9.7.2 ロケール

Debian は **ロケール** 技術をサポートしています。ロケールは プログラムが文字セット、時刻や時間のフォーマット、通貨文字などのような ローカルのしきたりに従って適切な出力や機能を供給することを可能とするための 機構です。例えば、米国英語とドイツのロケールの両方をシステムにインストール したと仮定すると、多くのプログラムのエラーメッセージは二ヵ国語にできます。

```
$ LANG="en_US" cat foo
cat: foo: No such file or directory
$ LANG="de_DE" cat foo
cat: foo: Datei oder Verzeichnis nicht gefunden
```

Glibc はこの機能のプログラムへのサポートをライブラリとして供給します。 `locale(7)` をご覧下さい。

9.7.3 ロケール入門

完全なロケールの説明は3つの部分、すなわち `xx_YY.ZZZZ` から構成されます。

- **`xx`**: ISO 639 言語コード (小文字)
- **`YY`**: ISO 3166 国コード (大文字)
- **`ZZZZ`**: コードセット、すなわち文字セットや エンコードの識別子。

言語コードと国コードについては、`info gettext` にある適切な説明 をご覧下さい。

このコードセット部分はクロスプラットフォーム互換性を達成するために 全ての - を削除し、全ての文字を小文字に変換することにより 内部で正規化されることがあります。典型的なコードセットは次のとおりです。

- **UTF-8**: 全ての領域のための Unicode、ほとんどは 1-3 Octed(新しいデファクトスタンダード。)

- **ISO-8859-1**: 西ヨーロッパ (古いデファクトスタンダード)
- **ISO-8859-2**: 東ヨーロッパ (ボスニア、クロアチア、チェコ、ハンガリー、ポーランド、ルーマニア、セルバニア、スロバキア、スロベニア)
- **ISO-8859-3**: マルタ語
- **ISO-8859-5**: マケドニア語、セルビア語
- **ISO-8859-6**: アラビア語
- **ISO-8859-7**: ギリシャ語
- **ISO-8859-8**: ヘブライ語
- **ISO-8859-9**: トルコ語
- **ISO-8859-11**: タイ語 (=TIS-620)
- **ISO-8859-13**: ラトビア語、リトアニア語、マオリ語
- **ISO-8859-14**: ウェールズ語
- **ISO-8859-15**: ユーロを用いた西ヨーロッパ
- **KOI8-R**: ロシア語
- **KOI8-U**: ウクライナ語
- **CP1250**: チェコ語、ハンガリー語、ポーランド語 (MS Windows が起源)
- **CP1251**: ブルガリア語、ベラルーシ語 (MS Windows が起源)
- **eucJP**: Unix スタイルの日本語 (=ujis)
- **eucKR**: Unix スタイルの韓国語
- **GB2312**: zh_CN 用の Unix スタイルの簡体字 (=GB, =eucCN)
- **Big5**: zh_TW 用の繁体字
- **sjis**: Microsoft スタイルの日本語 (シフト JIS)

基本的なエンコーディングシステムの jargon の意味については次のとおり。

- **ASCII**: 7 ビット (0-0x7f)
- **ISO-8859-?**: 8 ビット (0-0xff)
- **ISO-10646-1**: 国際文字集合 (UCS) (31 ビット,0-0x7fffffff)
- **UCS-2**: 2 個のオクテッドを用いた 16 ビットの UCS (Unicode: 0-0xffff)
- **UCS-4**: 4 個のオクテッドを使った UCS (UCS: 0-0x7fffffff)
- **UTF-8**: 1-6 オクテッドでエンコードされた UCS (ほとんどは 3 オクテッド)
- **ISO-2022**: エスケープシーケンス付きの 7 ビット(0-0xff) ISO-2022-JP は日本語のメールで最もポピュラーなエンコーディングです。
- **EUC**: 8 ビット + 16 ビットの組合せ (0-0xff) で Unix スタイル。
- **Shift-JIS**: 8 ビット + 16 ビットの組合せ (0-0xff)、 Microsoft スタイル。

ISO-8859-?, EUC, ISO-10646-1, UCS-2, UCS-4, そして UTF-8 は ASCII コードを 7 ビット文字のために共有しています。EUC と シフト JIS は 最上位ビットが立った (0x80-0xff) 文字をその部分のエンコードが 16 ビット であることを示すために使います。UTF-8 も最上位ビットが立った文字 (0x80-0xff) をこれが非 7 ビット文字シーケンスバイトであり、これが非 ASCII 文字を操作 するのに最適なエンコーディングシステムであることを示すために使います。

Unicode の実装でバイトオーダーが異なるものがあることに注意してください。

- **標準の UCS2, UCS4**: ビッグエンディアン
- **Microsoft UCS-2, UCS-4**: x86 用のリトルエンディアン (マシン依存)

各種文字セット間の変換に関しては '`recode` によりテキストファイルを変換する' on page 116 を御覧下さい。 より詳しくは Introduction to i18n (`http://www.debian.org/doc/manuals/intro-i18n/`) をご覧下さい。

9.7.4 ロケールサポートを有効にする

Debian は全ての使用可能なロケールを前もってコンパイルして**いません**。(標準の “C” ロケールを除いて) システムにコンパイルされているロケールの種類を知るには、`/usr/lib/locale` をチェックしてください。必要なロケールがコンパイルされていない場合には、2つのオプションがあります。

- `/etc/locale.gen` を編集して望みのロケールを追加しましょう。そして root で `locale-gen` を実行し、ロケールを コンパイルします。`locale-gen(8)` と このマニュアルページの “SEE ALSO” セクションにリストされた マニュアルページをご覧下さい。
- `locales` パッケージの再設定のために `dpkg-reconfigure locales` を起動しましょう。又は、このパッケージが まだインストールされていない場合は、`locales` を インストールして debconf インターフェースを起動して必要なロケールを選択し、データベースをコンパイルします。

9.7.5 特定のロケールを有効にする

次に示す環境変数はこの順番で評価され、特定のロケール値をプログラムに提供します。

1. LANGUAGE: この環境変数はコロンで分割され、優先順位の高い順に並んだ ロケール名のリストです。POSIX ロケールが “C” 以外の値にセットされている場合 のみ使われます。[Woody での説明。Potato では常に POSIX ロケールより高い 優先度を持つ] (GNU 拡張)
2. LC_ALL: これが null ではない場合、全てのロケール カテゴリに対して 使用されます。(POSIX.1) 通常 “” (null) です。
3. LC_*: これが null ではない場合、この値は対応するカテゴリに対して 使用されます。(POSIX.1) 通常 “C” です。

   LC_* variables には次の値が存在します。
   - LC_CTYPE: 文字の種類とエンコーディング。
   - LC_COLLATE: ソートの順番。
   - LC_TIME: 日付と時間の表示フォーマット。
   - LC_NUMERIC: 数字のフォーマット。
   - LC_MONETARY: 通貨のフォーマット。
   - LC_MESSAGES: 情報およびダイアログメッセージと対話式な応答のフォーマット。
   - LC_PAPER: 紙サイズ。
   - LC_NAME: 名前のフォーマット。
   - LC_ADDRESS: アドレスのフォーマットと位置情報。
   - LC_TELEPHONE: 電話番号フォーマット。
   - LC_MEASUREMENT: 測定単位。 (メートル又はその他の単位)
   - LC_IDENTIFICATION: ロケール情報に関するメタデータ。
4. LANG: これが null でなく、LC_ALL が未定義の場合、この値は 未定義の値の全ての LC_* ロケールカテゴリに対して使われます。(POSIX.1) 通常 “C” です。

(Netscape4 などの) いくつかのアプリケーションは LC_* の設定を無視する ことに注意してください。

`locale` プログラムは有効なロケールの設定と使用可能な ロケールを表示できます。`locale(1)` をご覧下さい。(`locale -a` はシステムが存在を知っている全てのロケールを表示しますが、これは全てがコンパイル済みであることを意味しない ことに注意下さい。'ロケールサポートを有効にする' on page 169 をご覧下さい。)

9.7.6 ISO 8601 日付表示ロケール

国際的なデータ標準である `yyyy-mm-dd` 形式 (ISO 8601 データフォーマット) の日付のロケールのサポートは `en_DK` すなわち "English in Denmark" と呼ばれるロケールによりサポートされています。これはちょっとしたジョークです。これは端末上で `ls` のためにしか動かないようです。

9.7.7 米国 (ISO-8859-1) の例

次の行を `~/.bash_profile` に追加してください。

```
LC_CTYPE=en_US.ISO-8859-1
export LC_CTYPE
```

9.7.8 ユーロを用いたフランス語の例 (ISO-8859-15)

次の行を `~/.bash_profile` に追加してください。

```
LANG=fr_FR@euro
export LANG
LC_CTYPE=fr_FR@euro
export LC_CTYPE
```

フランスの "AZERTY" キーボードを 'キーボードのローカライゼーション' on page 166 に記述しているように 設定し、`manpages-fr` をインストールしてフランス語の マニュアルページを追加してください。米国での 右 Alt キーはヨーロッパでは 右 Gr と呼ばれます。これを他のキーと同時に押すと数多くのアクセント付き 文字や特別な文字の入力ができます。例えば、Alt-Gr+E はユーロ記号になります。

ほとんどの西ヨーロッパ言語は同様に設定可能です。

新しいユーロ貨幣のサポートを追加するには、 Debian Euro HOWTO (`http://www.debian.org/doc/manuals/debian-euro-support/`) を、 フランス語についてより詳しく知るには、 Utiliser et configurer Debian pour le français (`http://www.debian.org/doc/manuals/fr/debian-fr-howto/`) をご覧下さい。

9.7.9 多言語 X window システムの例

コンソール毎に、日本語と英語とドイツ語とフランス語を、それぞれ EUC と UTF-8 と ISO-8859-1 のエンコーディングで同時にサポートする多言語 X window システム を設定しましょう。

まず Debian menu システムを用いたカスタマイズをお見せします。 Debian menu システムの詳細は `/usr/share/doc/menu/html/index.html` をご覧下さい。 また、この例では `mozilla` web ブラウザへのショートカットを 作成します。[8]

- ‘ローカライゼーション (l10n)’ on page 165 で記述されている方法を用いて日本語の ja_JP.eucJP ロケールや他の必要なロケールのサポートを追加します。(全員)
- かな漢字変換システムとその辞書をインストールします。(日本語用)
  - `canna` – ローカルサーバ (“フリービール” ライセンス) 又は
  - `freewnn-jserver` – ネットワーク拡張性の高い サーバ (パブリックドメイン) (Public Domain)
- 日本語入力メソッドシステムをインストールします。(日本語用)
  - `kinput2-canna` – X 用 又は
  - `kinput2-canna-wnn` – X 用
  - `egg` – 端末でも Emacsen 環境で直接動く (任意)
- vt 互換端末をインストールします。(全員)
  - `xterm` – X (ISO-8859-1 と UTF-8 用)
  - `kterm` – X (日本語 EUC 用)
  - `mlterm` – X (多言語用)
- 必要な全てのフォントパッケージをインストールします。(全員)
- ユーザ指定の X 環境をセットする `~/.xsession` ファイルを ‘X セッションのカスタマイズ’ on page 147 の記述に従い作成します。(全員)

```
#!/bin/sh
# root に su した時も X が動くようにする。
if [ -z "$XAUTHORITY" ]; then
        XAUTHORITY=$HOME/.Xauthority
        export XAUTHORITY
fi

# debian menu システムを通じて特別な環境をセットする。
# ロケールのリセット
unset LC_CTYPE LC_NUMERIC LC_TIME LC_COLLATE LC_MONETARY LC_MESSAGES
unset LC_PAPER LC_NAME LC_ADDRESS LC_TELEPHONE LC_MEASUREMENT
```

---

[8] In this example, 2 bug work arounds are deployed for the version of `blackbox` in 2003. I use `sh -c` in command. Also `~/.menu/*` entry is not used but root requiring `/etc/menu/*` was used instead.

```
unset LC_IDENTIFICATION LC_ALL LANG LANGUAGE PAGER
# X での標準のロケールをセット
LANG=C
# ロケールをエクスポート
export LC_CTYPE LC_NUMERIC LC_TIME LC_COLLATE LC_MONETARY LC_MESSAGES
export LC_PAPER LC_NAME LC_ADDRESS LC_TELEPHONE LC_MEASUREMENT
export LC_IDENTIFICATION LC_ALL LANG LANGUAGE PAGER
###
# kinput2 による日本語入力メソッドを有効にする
kinput2 &
XMODIFIERS="@im=kinput2"
export XMODIFIERS
# blackbox ウィンドウマネージャはどうですか? (軽いですよ)
exec blackbox
#exec xfwm
#exec wmaker
```

- Linux 端末用に `~/.bash_profile` にロケールを設定する。(全員)
- 存在するならば`~/.bashrc` からロケールの設定を削除する。(全員)
- `/etc/menu/` にいくつかファイルを追加する。(全員)
  - `/etc/menu/xterm-local`: (menu に新しいエントリを追加) [9]

    ```
    ?package(xterm):\
     needs=x11\
     section=XShells\
     longtitle="XTerm: terminal emulator (en_US.ISO-8859-1)"\
     title="XTerm (en_US.ISO-8859-1)"\
     command="sh -c 'LC_ALL=en_US.ISO-8859-1 xterm'"
    ?package(xterm):\
     needs=x11\
     section=XShells\
     longtitle="XTerm: terminal emulator (de_DE.ISO-8859-1)"\
     title="XTerm (de_DE.ISO-8859-1)"\
     command="sh -c 'LC_ALL=de_DE.ISO-8859-1 xterm -T xterm-de'"
    ?package(xterm):\
     needs=x11\
     section=XShells\
     longtitle="XTerm: terminal emulator for X with Unicode support (Jap
     title="UXTerm (ja_JP.UTF-8)"\
     command="sh -c 'LC_ALL=ja_JP.UTF-8 uxterm'"
    ```

  - `/etc/menu/kterm`: (システムの標準を上書き) [10]

    ```
    ?package(kterm):\
     needs="x11"\
    ```

---

[9] パッケージ名と重複しない名前を使いましょう。

[10] パッケージ名と重複しないファイル名を使いましょう。

```
 section="XShells"\
 command="sh -c 'LC_ALL=ja_JP.eucJP PAGER=w3m /usr/X11R6/bin/kterm -
 title="Kanji Terminal"
?package(kterm):\
 needs="x11"\
 section="XShells"\
 command="sh -c 'LANG=ja_JP.eucJP \
           LC_MESSAGES=en_US.ISO-8859-1 PAGER=w3m /usr/X11R6/bin/kter
 title="Kanji Terminal (bilingal)"
```

  - `/etc/menu/mozilla-local`: (新しいショートカットを追加) [11]

```
?package(mozilla-browser):needs="x11" section="/" \
        title=" Mozilla Navigator" command="mozilla-1.5" hints="Web
        icon=/usr/share/pixmaps/mozilla.xpm
```

  - root アカウントから `update-menus` を起動します。

- `~/.muttrc` に次の行を追加します。(日本語)

```
# UTF-8 support is not popular in popular Japanese EMACS environment
# 7-bit encoding of iso-2022-jp is easier for everyone.
# default encoding order = us-ascii --> iso-8859-1 --> iso-2022-jp
set send_charset="us-ascii:iso-8859-1:iso-2022-jp"
set allow_8bit=no
```

- X アプリケーションのために XIM `kinput2` を有効にします。(日本語用)
  - X リソースファイルに `*inputMethod: kinput2` と `KTerm*VT100*OpenIm:true` を追加します。(Debian はこれをどういうわけか自動で対応しているようです。)
  - (`mlterm` のような) いくつかのアプリケーションは `*inputMethod:` を設定すると他の情報は動的に起動時に設定する ことを許可します。(`mlterm` では **Ctrl-MouseButton-3** を押します。)

- `startx` をタイプするか、ディスプレ イマネージャ (`xdm`, `gdm`, `kdm`, `wdm` など) のどれかを用いてログインすることにより X を起動します。(全員)

- Vim 6, (x)emacs21, mc-4.5, mutt-1.4 のような日本語に対応した アプリケーションを `kterm` 上で起動します。(日本語用) (Emacs は最もポピュラーなプラットフォームのようですが、私は使っていません。)

- **Shift+Space** を押して日本語入力モードの ON/OFF をトグルします。(日本語用)

- ローカライズされた端末上でコマンドを起動して翻訳済みの マニュアルページを読みます。(全員)

他の CJK 言語のサポートについては、次の章に続いて SuSE pages for CJK (`http://www.suse.de/~mfabian/suse-cjk/suse-cjk.html`) をご覧下さい。

---

[11] `section="/"` のスラッシュは最初の menu へのエントリを可能にし、`title=" Mozilla Navigator"` にある先頭のスペースはリストの 先頭へのエントリを可能にします。

9.7.10 X での代替入力メソッド

パッケージとして使用可能な X 入力メソッドがいくつか存在します。

言語	LC_CTYPE	XIM サーバ	XMODIFIERS	開始キー
日本語	`ja_JP*`	`kinput2`	`"@im=kinput2"`	`Shift-Space`
韓国語	`ko_KR*`	`ami`	`"@im=Ami"`	`Shift-Space`
中国語(繁体字)	`zh_TW.Big5`	`xcin`	`"@im=xcin-zh_TW.big5"`	`Ctrl-Space`
中国語(簡体字)	`zh_CN.GB2312`	`xcin`	`"@im=xcin-zh_CN.GB2312"`	`Ctrl-Space`

日本語入力メソッドである `kinput2` は `kinput2-canna-wnn`, `kinput2-canna`, `kinput2-wnn` のようなパッケージにより供給されています。日本語入力には他に `canna` や `freewnn-jserver` のような辞書サーバが実際には必要になります。

9.7.11 X ターミナルエミュレータ

適切なフォントがインストールされた時に 8 ビットエンコーディングをサポートする X 端末は数多く存在します。

- `xterm` – X 付属のターミナルエミュレータ
- `gnome-terminal` – Gnome 用の `xterm`
- `konsole` – KDE 用の `xterm`
- `rxvt` – VT102 ターミナル (軽量)
- `aterm` – Afterstep WM 用の VT102
- `eterm` – Enlightment WM 用の VT102
- `wterm` – WindowMaker WM 用の VT102

X 端末のマルチバイトエンコーディングのサポートは UTF-8 エンコーディング ('X 端末エミュレータの UTF-8 サポート' on page 175)を通じて `xterm` により供給されています。他の伝統的なエンコーディングのサポートは (2003 年時点で) 推進中です。次のパッケージは伝統的なエンコーディングのサポートを提供します。

- `aterm-ml` – 多言語
- `kterm` – 多言語 (日本語など)
- `rxvt-ml` – 多言語
- `wterm-ml` – 多言語
- `cxterm-big5` – 中国語 (繁体字、Big5)
- `cxterm-gb` – 中国語 (簡体字、GB)
- `cxterm-ks` – 中国語 (KS)
- `cxterm-jis` – 日本語
- `hanterm-classic` – 韓国語 (ハングル)
- `hanterm-xf` – 韓国語 (ハングル)
- `hztty` – 中国語 (GB, Big5, zW/HZ)

`kterm` では、(多分他の端末でも) Ctrl-中クリックマウス操作をした後にメニュから XIM を有効にしたいかもしれません。

9.7.12 X 端末エミュレータの UTF-8 サポート

X ターミナルエミュレータの UTF-8 サポートは、XFree86 4.x 用の `xterm` パッケージにある `uxterm` プログラムにより供給されています。これは全言語のサポートを可能にします。これは "UXTerm" X リソースクラスセットを用いた後者のプログラムを起動する `xterm(1)` プログラムのラッパーです。

例えば、英語、ロシア語、日本語、中国語、韓国語の文字をきれいに大きく表示することを可能にするには、関連する全フォントをインストールした後に `~/.Xresources` に次を追加してください。

```
! set large font
UXTerm*font: -misc-fixed-medium-r-normal-*-18-120-100-100-c-90-iso10646-1
! Use XIM for Japanese
*inputMethod: kinput2
```

そして X リソースを更新するため、'X リソース' on page 151 に記述されているように `xrdb -merge ~/.Xresources` を起動します。

`vim`, `mutt` や `emacs` のような 有名なコンソールプログラムのほとんどは最近 (Woody から Sarge にかけて) UTF-8 と互換性がありますが、`mc` のようなプログラムは まだ UTF-8 互換ではなく、シンプルな 8 ビット文字を表示できるだけです。 未知あるいは混在したエンコーディングのファイルの中の 7 ビット ASCII の 部分を編集する場合、ロケールを気にしない 8 ビットクリーンなエディタを 使う方が安全です。

The Unicode HOWTO (`http://www.tldp.org/HOWTO/Unicode-HOWTO.html`) をご覧下さい。

9.7.13 フレームバッファコンソールで UTF-8 を使う例

フレームバッファコンソール上での UTF-8 のサポートは `debian-installer` 上で使用される `bterm` により供給されています。

9.7.14 ロケールを越えて

まず最初に **母国語環境** の設定を行う際には、 言語環境に対応するタスクを選ぶことにより選び抜かれるパッケージを調べるために `tasksel` や `aptitude` を使うことを考えてください。 このパッケージの選択は多言語環境のセットアップのためでも役に立ちます。 システムを注意深く設定するために、 インストール中に依存性の衝突に出くわした 場合、既存のシステムと衝突するソフトウェアのインストールを避けてください。 新たにインストールしたパッケージは既存のパッケージよりも高い優先度を 持っているかもしれないので、いくつかのコマンドでは元の状態に戻すために `update-alternative` を使う必要があるかもしれません。

新しめの主要なプログラムは glibc 2.2 を使っており、ほぼ国際化されています。 それゆえ、Vim に対する `jvim` のような特別に国際化されたバージョンは `vim` version 6.0 により X

上でほぼ同様の機能が提供されているために必要では無いかもしれません。現実には、まだいくらかおおざっぱです。`jvim` は端末上でも直接日本語入力メソッド (`canna`) のサポート付きでコンパイルされており、さらに多くの他の日本語特有の問題に分別深く対応しているので、まだこれが欲しいかもしれません。:-)

快適な作業環境を可能にするには、ロケール設定を越えた設定が必要になるかもしれません。`language-env` パッケージやその `set-language-env` コマンドはこの設定作業をかなり軽減してくれます。

又、国際化に関する文書である Introduction to i18n (`http://www.debian.org/doc/manuals/intro-i18n/`) もご覧下さい。これは開発者向けですが、システム管理者に対しても有用です。

9.8 マルチリンガリゼーション (m17n)

`language-env` 類似パッケージによって実現される 'ローカライゼーション (l10n)' on page 165 は単一言語によるローカリゼーションを目指しています。これらのパッケージはテキストのエンコーディングとして伝統的エンコーディングを使います。このような環境下では、フランス語と日本語はそれぞれ ISO-8859-1 と EUC-JP という相容れないエンコーディングを使うのでテキストを混ぜることが出来ません。

多言語化された UTF-8 デスクトップは、Gnome や KDE のプログラムを、存在する UTF-8 ロケールのうちのいずれかにロケールを設定した上で起動することにより得られます。(Sarge) そのような環境下では、UTF-8 対応のソフトの下で、英語、中国語、ロシア語、日本語の文字を混ぜて使えます。

そのような環境では、新しい `scim` を用いる多言語化された入力メソッド (IM) が好まれています。`scim` により提供される入力メソッドは、Ctrl と Space を同時に押すことにより on/off できます。入力変換エンジンは、小さな SCIM パネルをクリックすることにより、切替え可能です。

`vim` は、UTF-8 端末、例えば `gnome-terminal` などで起動された時に、多言語化された環境を提供し、UTF-8 および伝統的なエンコードのファイル (EUC-JP、ISO-8859-1 など) の両方を扱うことが可能です。[Esc] キーを押して、`:help mbyte.txt` とタイプすれば、vim のヘルプメッセージを表示できます。

章 10

ネットワークの設定

この章では Debian でのネットワーク管理に焦点を当てています。 GNU/Linux のネットワークに関する一般的な紹介は Net-HOWTO (`http://www.tldp.org/HOWTO/Net-HOWTO/index.html`) をごらんください。

Debian ホストのインターネットへのアクセスを可能にするには、 ネットワークインターフェイスが適切に設定される必要があります。

まず最初に必要なことは、ネットワークインターフェイスデバイスのカーネルサポートです。 ネットワークデバイスの例としては、以下が挙げられます: イーサネットカード、無線 LAN カードやモデム。 ネットワークインターフェイスデバイスを kernel がサポートすることです。 これらのサポートを有効にするには、'Debian での Linux kernel' on page 99 で記述されているように kernel を再構築してモジュールを追加する必要があるかもしれません。

ネットワークデバイスの設定は以降で説明します。 本章の情報は Sarge 向けに更新されました。 本章の内容のほとんどは以前のリリースには適用できません。

10.1 IP ネットワークの基礎

Debian ホストはそれぞれが異なった Internet Protocol (IP) アドレスを持った 複数のインターフェイスを持つことができます。 インターフェイスは次に挙げるいくつかの異なるタイプかもしれません。

- ループバック: `lo`
- イーサネット: `eth0`, `eth1`, ...
- 無線LAN: `wlan0`, `wlan1`,`wifi0` ... [1]
- トークンリング: `tr0`, `tr1`, ...
- PPP: `ppp0`, `ppp1`, ...

SLIP, PLIP (srrial line IP と pararell line IP), 特定のデバイスで トラフィックを制御するための "shaper" なデバイス, フレームリレー, Ax.25, X.25, ARCnet そして LocalTalk のような他のさまざまな種類の ネットワークデバイスが使用可能です。

---

[1] 無線 LAN インターフェイスの一部は本当はイーサネット インターフェイスの別名であり、無線 LAN 独特の設定パラメータへのヒントを与えてくれます。 これらのパラメータは `iwconfig` プログラムを用いて 制御可能です。

インターネット (又は IP ベースのネットワーク) に直接接続された全ての ネットワークインターフェイスはユニークな 32 ビットの IP アドレスにより 識別されます。[2] IP アドレスネットワークを示す部分とホストを示す部分に分割できます。IP アドレスのネットワークアドレスの部分を 1 に、ホストアドレスの 部分を 0 にすると、ネットワークのいわゆるネットマスクが得られます。

伝統的に、IP ネットワークはそれぞれ 8 ビット、16 ビット、24 ビットの長さの ワークアドレス部分を持つクラスに分類されています。これらのシステムは柔軟性に欠けており、多数の IP アドレスを無駄にしていました。それゆえ、今日では IPv4 ネットワークは可変長のネットワークアドレス部分を持つ ように割り当てられています。

	IP アドレス	ネットマスク	長さ
クラス A	1.0.0.0 - 126.255.255.255	255.0.0.0	= /8
クラス B	128.0.0.0 - 191.255.255.255	255.255.0.0	= /16
クラス C	192.0.0.0 - 223.255.255.255	255.255.255.0	= /24

以上の長さを持たない IP アドレスは特別な目的に用いられます。

各クラスのためにローカルエリアネットワーク (LAN) で使用するために予約された アドレスの範囲が存在します。これらのアドレスはインターネット上のあらゆるアドレスと衝突しないことが 保証されています。(同じトークンにより、これらのアドレスの 1つがホストに割り当てられた場合、そのホストはインターネットに直接アクセスしてはいけませんが、個別のサービスのためのプロキシとして機能するゲートウェイを通して アクセスするか、ネットワークアドレス変換を通じてアクセスする 必要があります。) これらのアドレスの範囲はそれぞれのクラスにあるアドレスの数に応じて、次のテーブルに示すように与えられています。

	ネットワークアドレス	長さ	数
クラス A	10.x.x.x	/8	1
クラス B	172.16.x.x - 172.31.x.x	/16	16
クラス C	192.168.0.x - 192.168.255.x	/24	256

IP ネットワークの最初のアドレスはネットワーク自身のアドレスです。最後のアドレスはネットワークのブロードキャストアドレスです。[3] 他の全てのアドレスはネットワーク上のホストに割り当てられます。これらのうち、最初又は最後のアドレスは通常ネットワークのインターネットゲートウェイに割り当てられます。

ルーティングテーブルは kernel の目的地への IP パケットの送信方法 についての情報を含んでいます。ここに IP アドレス 192.168.50.x/24 を持つローカルエリアネットワーク (LAN) 上の Debian ホストのためのサンプルのルーティングテーブルを示します。ホスト 192.168.50.x (LAN 上にもある) は企業ネットワーク 172.20.x.x/16 のためのルータであり、192.168.50.254 (LAN 上にもある) はインターネット 用のルータです。

---

[2] これは IPv4 が使用されている場合に正しい。IPv6 ではアドレスは 128 ビット。`http://www.ipv6.org/` 参照。

[3] ネットワークのアドレスはネットワークアドレスとネットワークのネットマスクの和を取ることにより得られます。ブロードキャストアドレスはネットワークアドレスと ネットマスクの一の補数の積を取ることにより得られます。

```
# route
Kernel IP routing table
Destination   Gateway         Genmask         Flags Metric Ref Use Iface
127.0.0.0     *               255.0.0.0       U     0      0     2 lo
192.168.50.0  *               255.255.255.0   U     0      0   137 eth0
172.20.0.0    192.168.50.1    255.255.0.0     UG    1      0     7 eth0
default       192.168.50.254  0.0.0.0         UG    1      0    36 eth0
```

- ヘッダの後の最初の行は `127.x.x.x` 向けのトラフィックは `lo` すなわちループバックインターフェイスを通ることを示しています。
- 2 行目は LAN 上のホスト向けのトラフィックは `eth0` を通ることを 示しています。
- 3 行目は企業ネットワーク向けのトラフィックは `eth0` を通って ゲートウェイ `192.168.50.1`に向けられることを示しています。
- 4 行目はインターネット向けのトラフィックは`eth0` を通ってゲートウェイ `192.168.50.254` に向けられることを示しています。

テーブル上の IP アドレスは `/etc/networks` を調べるか C ライブラリのリゾルバを用いて得られる名前としても表せます。

ルーティングに加え、kernel はネットワークアドレス変換、トラフィック整理 やフィルタリングも行えます。

より詳しい背景情報については Net-HOWTO (`http://www.tldp.org/HOWTO/Net-HOWTO/index.html`) と other networking HOWTOs (`http://www.tldp.org/HOWTO/Networking-Overview-HOWTO.html`) をごらんください。

10.2 低レベルネットワークの設定

GNU/Linux システム上での伝統的な低レベルネットワーク設定ツールとしては `net-tools` パッケージに収録されている `ifconfig` や `route` プログラムがあります。 これらのツールは公式に `iproute` パッケージに収録 されている `ip` により取って代われています。 `ip` プログラムは Linux 2.2 以上で動き、旧式のツールよりも より機能的です。しかしながら、旧式のツールもまだ動き、多くのユーザにより 親しまれています。

10.2.1 低レベルネットワーク設定ツール – **`ifconfig`** と **`route`**

ここにインターフェイス `eth0` の IP アドレスを `192.168.0.3` から `192.168.0.111` に変更し、`192.168.0.1` 経由で `10.0.0.0` ネットワークに通すように `eth0` を設定 する方法を示します。 現在の全てのネットワークインターフェイスとルーティングの状況を表示するため、 まずはインターフェイス引数無しで `ifconfig` と `route` を起動することから始めましょう。

```
# ifconfig
eth0 Link encap:Ethernet  HWaddr 08:00:46:7A:02:B0
     inet addr:192.168.0.3  Bcast:192.168.255.255  Mask:255.255.0.0
```

```
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:23363 errors:0 dropped:0 overruns:0 frame:0
          TX packets:21798 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:13479541 (12.8 MiB)  TX bytes:20262643 (19.3 MiB)
          Interrupt:9

     lo   Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:230172 errors:0 dropped:0 overruns:0 frame:0
          TX packets:230172 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:22685256 (21.6 MiB)  TX bytes:22685256 (21.6 MiB)
     # route
     Kernel IP routing table
     Destination  Gateway      Genmask          Flags Metric Ref Use Iface
     192.168.0.0  *            255.255.0.0      U     0      0     0 eth0
     default      192.168.0.1  255.255.255.255  UG    0      0     0 eth0
```

まずインターフェイスをダウンさせます。

```
     # ifconfig eth0 inet down
     # ifconfig
     lo   Link encap:Local Loopback
       ... (eth0 エントリはもはや存在しない)
     # route
       ... (ルーティングテーブルのエントリももはや存在しない)
```

次に eth0 を新しい IP アドレスと新しいルーティングを用いて立ち上げます。

```
     # ifconfig eth0 inet up 192.168.0.111 \
               netmask 255.255.255.0 broadcast 192.168.0.255
     # route add -net 10.0.0.0 netmask 255.0.0.0 gw 192.168.0.1 dev eth0
```

結果は次のようになります。

```
     # ifconfig
     eth0 Link encap:Ethernet  HWaddr 08:00:46:7A:02:B0
          inet addr:192.168.0.111  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          ...

     lo   Link encap:Local Loopback
```

```
      inet addr:127.0.0.1  Mask:255.0.0.0
      ...
# route
Kernel IP routing table
Destination  Gateway      Genmask       Flags Metric Ref Use Iface
192.168.0.0  *            255.255.255.0  U      0      0     0 eth0
10.0.0.0     192.168.0.1  255.0.0.0     UG    0      0     0 eth0
```

より詳しい情報は、`ifconfig(8)` と `route(8)` をごらんください。

10.2.2 低レベルネットワーク設定ツール – ip

前記の `ifconfig` や `route` コマンドと等価な `ip` コマンドは次のようになります。

- `ip link show`
- `ip route list`
- `ip link set eth0 down`
- `ip addr del dev eth0 local 192.168.0.3`
- `ip addr add dev eth0 local 192.168.0.111/24 broadcast 192.168.0.255`
- `ip link set eth0 up`
- `ip route add dev eth0 to 10.0.0.0/8 src 192.168.0.111 via 192.168.0.1`

`ip` プログラムは引数 `help` 付きで起動するとその コマンド文法を表示します。例えば、`ip link help` は次を表示します。

```
Usage: ip link set DEVICE { up | down | arp { on | off } |
                     dynamic { on | off } |
                     multicast { on | off } | txqueuelen PACKETS |
                     name NEWNAME |
                     address LLADDR | broadcast LLADDR |
                     mtu MTU }
       ip link show [ DEVICE ]
```

`ip(8)` をごらんください。

10.2.3 無線 LAN インターフェイスの設定

無線 LAN インターフェイスの設定には、`wireless-tools` パッケージに収録されている `iwconfig` プログラムが `ifconfig` と `ip` の両方に加えて使われます。

`iwconfig(8)` をごらんください。

10.2.4 PPP インターフェイスの設定

インターネットにダイアルアップ電話線を接続したモデム経由でアクセスする場合、Point-to-Point Protocol (PPP) を用いて接続をネゴシエイトします。そのような接続はネットワークインターフェイス `ppp0`, `ppp1`, などとしてアクセスされます。

PPP インターフェイスは `ppp` パッケージに収録されている PPP デーモン `pppd` により管理されます。それゆえ、ユーザにとっては、PPP インターフェイスを設定するのは `pppd` を設定することを意味します。

pppd を手動で設定する

ネットワークリンクを確立するには、通信ポート (通常シリアルポート) を開き、コマンドを通信デバイス (通常はモデム) に送信し、電話番号をダイアルし、外部の PPP デーモンへの認証を行い、PPP インターフェイスがカーネルにより作成され、トラフィックがリンクを通じて送られるようにルーティングテーブルが修正される必要があります。`pppd` はこれらの仕事の全てをこなし、その結果として非常に長い操作オプションのリストを持ちます。これらのオプションは `pppd(8)` に記述されています。

Debian システムでは、汎用のオプションは `/etc/ppp/options` にセットされています。ユーザが指定するオプションは `~/.ppprc` にセットします。使用する通信ポートに依存するオプションは `/etc/ppp/options.`*`portname`* に保存されます。例えば、二つのモデム — 組込みの /dev/LT-modem を通じてアクセスされる Lucent LT モデムと外付けの /dev/ttyS0 を通じてアクセスされるモデムを持っていると仮定します。次の 2 つのオプションファイルを作成します。

```
# cat > /etc/ppp/options.LT-modem <<EOF
115200
init "/usr/sbin/chat -f /etc/chatscripts/setup-LT-modem"
EOF
# cat > /etc/ppp/options.ttyS0 <<EOF
115200
init "/usr/sbin/chat -f /etc/chatscripts/setup-ttyS0"
EOF
```

これらは次のチャットスクリプトを参照しています。まず、`/etc/chatscripts/setup-LT-modem` です。

```
ABORT ERROR
'' ATZ
OK 'ATW2X2 S7=70 S11=55'
OK AT
```

次に、`/etc/chatscripts/setup-ttyS0` です。

```
ABORT ERROR
'' ATZ
OK 'ATL1M1Q0V1W2X4&C1&D2 S6=4 S7=70 S11=55 S95=63 S109=1 +FCLASS=0'
OK AT
```

これらのファイルの内容はもちろんハードウェアに依存している必要があります。

オプションは `pppd` に引数としても与えられます。

Debian では、`pppd` は普通 `pon` コマンドを 用いて起動されます。`pon` が起動されると、その第一引数は `pppd` によっても読まれる `/etc/ppp/peers/` に あるオプションファイルと同じ名前です。[4] こうして特定の peer—例えば、特定のインターネットサービスプロバイダ (ISP) を指定するオプションを設定できます。

例えば、アムステルダムとハーグの間を通信したいとします。 それぞれの町には2つの ISP サービス—Planet と KPN を持っています。 まず各 ISP 用の基本的なオプションファイルを作ります。

```
# cat > /etc/ppp/peers/KPN <<EOF
remotename KPN
noauth
user kpn
noipdefault
ipparam KPN
EOF
# cat > /etc/ppp/peers/Planet <<EOF
remotename Planet
auth
user user3579@planet.nl
noipdefault
mru 1000
mtu 1000
ipparam Planet
EOF
```

これらのファイルは2つの ISP 間で異なるオプションを設定します。 両方の ISP に共通のオプションは `/etc/ppp/options` 又は適切なインターフェイス独自のオプションファイルのどちらかに 置きます。

次にそれぞれの町のための各 ISP 用のオプションファイルを作成します。 この例では片方の場所の ISP へ接続するのと他の場所の ISP に 接続する時の違いのみを必要なチャットスクリプトにします。 (ローカルのアクセス電話番号が異なるので、チャットスクリプトは異なります。)

```
# cat > /etc/ppp/peers/KPN-Amsterdam <<EOF
```

---

[4] このオプションファイルは `call` オプションを用いて含まれます。

```
connect "/usr/sbin/chat -v -f /etc/chatscripts/KPN-Amsterdam"
file /etc/ppp/peers/KPN
EOF
# cat > /etc/ppp/peers/KPN-DenHaag <<EOF
connect "/usr/sbin/chat -v -f /etc/chatscripts/KPN-DenHaag"
file /etc/ppp/peers/KPN
EOF
# cat > /etc/ppp/peers/Planet-Amsterdam <<EOF
connect "/usr/sbin/chat -v -f /etc/chatscripts/Planet-Amsterdam"
file /etc/ppp/peers/Planet
EOF
# cat > /etc/ppp/peers/Planet-DenHaag <<EOF
connect "/usr/sbin/chat -v -f /etc/chatscripts/Planet-DenHaag"
file /etc/ppp/peers/Planet
EOF
```

それぞれの `file` 指示は早く示されるオプションファイルの一つ の一つを含みます。`connect` 指示は接続を確立するために pppd が使う コマンドを指示します。 通常この目的には `chat` プログラムを使い、 チャットスクリプトを ISP に適応させます。 アムステルダムは電話番号を除いて同じかもしれませんし、ISP が 他の会社を通じてサービスを提供しているならば、異なるかもしれません。

```
# cat > /etc/chatscripts/KPN-DenHaag <<EOF
ABORT BUSY
ABORT 'NO CARRIER'
ABORT VOICE
ABORT 'NO DIALTONE'
ABORT 'NO DIAL TONE'
ABORT 'NO ANSWER'
ABORT ERROR
OK-AT-OK ATDT 0676012321
CONNECT \d\c
EOF
# cat > /etc/chatscripts/Planet-DenHaag <<EOF
ABORT BUSY
ABORT 'NO CARRIER'
ABORT VOICE
ABORT 'NO DIALTONE'
ABORT 'NO DIAL TONE'
ABORT 'NO ANSWER'
ABORT ERROR
OK-AT-OK ATDT 0676002505
CONNECT \d\c
EOF
```

これらの ISP と接続を確立できるようにするには、pppd が peer に要求に応じて供給するク

ライアント名とパスワードが必要です。 これらの情報は `/etc/ppp/pap-secrets` (PAP プロトコル使用時) 又は `/etc/ppp/chap-secrets` (CHAP プロトコル使用時) に保存されます。 CHAP はより安全ですが、PAP は依然広く使われています。 これらのファイルには秘密が含まれているので、グループと他人が読み書きできない ファイルアクセス権限を設定しなくてはなりません。 これらのファイルのフォーマットは `pppd(8)` で 説明されています。 “secret” (3 番目のフィールド) はファイル中のクライアント名 (最初のフィールド) および/又は サーバ名 (2 番目のフィールド) を検索することにより調べられます。 ISP に接続している場合、一般にサーバ名は知りませんので、クライアント名を 代わりに供給します。これは上に示した `peers/KPN` と `peers/Planet`にある `user` 行で行われます。

```
# client name          server name   secret
kpn                    *             kpn
user3579@planet.nl     *             myfavoritepet
```

詳細は `/usr/share/doc/ppp/README.Debian.gz` をごらんください。

`pppconfig` を用いて **`pppd`** を設定する

`pppd` を設定する手軽な方法は同名のパッケージにより 供給される `pppconfig` を使うことです。 `pppconfig` はメニュインターフェイスを通じてユーザが質問 に答えた後に上に示したようなファイルを設定します。

`wvdial` を用いて PPP インターフェイスを設定する

`pppd` を用いる異なるアプローチとしては、 `wvdial` パッケージに収録されている `wvdial` から `pppd` を起動することです。 `pppd` の代わりに、`chat` を起動してダイアルし、 接続のネゴシエイトを行います。`wvdial` はダイアルと最初のネゴ シエイトを行い、そして残りの作業を行うために `pppd`を起動します。 電話番号、ユーザ名、そしてパスワードを与えるだけで `wvdial` は ほとんどの場合接続を確立できます。

10.3 コンピュータの名前付け

10.3.1 ホスト名

カーネルはシステムの **ホスト名**を維持します。 initscript `/etc/init.d/hostname.sh` はシステムのホスト名を ブート時に (`hostname` を用いて) `/etc/hostname` に保存されている名前に設定します。 このファイルには完全修飾ドメイン名 (FQDN) ではなく、 システムのホスト名 **のみ** が含まれていなくてはなりません。

現在のホスト名を表示するには、引数無しで `hostname` を起動してください。

10.3.2 Mailname

ホストの **mailname** はホストを識別するためにメール関連の プログラムが使用する名前です。 `/etc/mailname` ファイルには名前に続いて改行を持つ行が 含まれます。 mailname はホストの IP アドレスの一つを解決する完全修飾ドメイン名です。 `mailname(5)` をごらんください。

Debian ホストにより送信されるメールの `From:` ヘッダにある 受取人が何かは Mail User Agent (MUA) や Mail Transfer Agent (MTA) がどのように設定されているかに依存します。 ローカルユーザ *`foo`* がメールを mailname *`myhost.dom`* から送ると仮定します。 すると、送信されるメールの `From:` ヘッダは次のようになります。

- MUA が `From:` ヘッダセットを持たない場合 "`From:` *`foo@myhost.dom`*"
- MUA が "`From:` *`bar`*" セットを持つ場合 "`From:` *`bar@myhost.dom`*"
- MUA が "`From:` *`bar@bogus.dom`*" セットを持つ場合 "`From:` *`bar@bogus.dom`*"

MUA が `From:` ヘッダをセットしたとしても、MTA が "`Sender:`*`foo@herman.dom`*" ヘッダを 本当の発信源を示すために追加するかもしれません。

もちろん 'Exim の元で存在しないメールアドレスのための物置を設定する' on page 163 や 'Exim の元で送信メールに対してアドレス書き換えを設定する' on page 163 で議論 したように、受信者が見るメールアドレスは他のアドレスに変更可能です。

10.4 ドメインネームサービス (DNS)

ホストは IP アドレスに加え、ドメイン名によっても参照されます。 DNS はネームリゾルバがドメイン名を IP アドレスや他のホストの属性と 関連付けるためにネームサーバを勤めるクライアントサーバシステムです。 GNU C ライブラリ `resolver(3)` もファイルにある IP アドレスの検索を行えますし、Network Information Services (NIS) の役目も勤めます。

ソフトウェアには (例えば GNOME のように) システムのホスト名が標準的な FQDN を持つ IP アドレスに解決可能であることを期待しています。 システムのホスト名とドメイン名はまったく異なるものであるため、これは 本当に不適切です。しかし、これらのソフトを使わざるを得ません。 これらのソフトをサポートするには、システムのホスト名が解決可能 であることを保証する必要があります。 ほとんどの場合、これは `/etc/hosts` に IP アドレスと システムのホスト名を含む行を追加することにより行なわれます。 システムに継続的な IP アドレスが割り当てられている場合、この手段は 利用できますが、そうでは無い場合、アドレス 127.0.0.1 を使います。

```
127.0.0.1 localhost
127.0.0.1 uranus
```

システムのホスト名が FQDN を持つ IP アドレスに解決可能であるかを 知るには、`hostname --fqdn` コマンドを使用ください。

10.4.1 リゾルバ

特定のドメイン名に関連づけられたIPアドレスを捜し出す仕事は、リゾルバの仕事です。最も良く使われているリゾルバはGNU Cライブラリにある リゾルバ(`resolver(3)`)により実現される 関数群です。他のリゾルバとしては、`libfiredns` パッケージに収録されている FireDNS リゾルバがあります。 他のパッケージも存在します。

どのように GNU LIBC リゾルバが名前を解決するかは `/etc/nsswitch.conf` 設定ファイル中の `hosts` 行により制御されています。 この行は名前を解決するために使われるサービス、例えば `dns,files,nis,nisplus` をリストしています。 `nsswitch.conf(5)` をごらんください。 `files` サービスが使用されている範囲については、リゾルバの挙動は `/etc/hosts` 設定ファイルによっても制御されています。 `hosts(5)` をごらんください。

上の全てのファイルは静的であり、お好みのエディタで編集できます。

`dns` サービスが使用される範囲においては、リゾルバの挙動は `/etc/resolv.conf` 設定ファイルによっても制御されています。 `resolv.conf(5)` をごらんください。 `resolv.conf` の重要な機能のひとつは、名前を解決するために 接続されるネームサーバのIPアドレスをリストすることです。 このリストはたびたびネットワーク環境に依存し、ネットワーク環境はコンピュータ を起動する間時にふれて変化します。 `pppd` や `dhclient` のようなプログラムは `resolv.conf` 内の行を追加したり削除する操作を行えますが、 これらの機能は常に適切に動くとは限らず、お互いに衝突しかねません。 `resolvconf` パッケージはこのファイルを更新するための 標準的なフレームワークを供給することにより、よりうまくこの問題を解決します。'ネームサーバ情報の管理 – `resolvconf`' on page 187 をごらんください。

10.4.2 ネームサーバ情報の管理 – `resolvconf`

`resolvconf` パッケージは利用可能なネームサーバ に関する情報の動的な管理のためのフレームワークを供給します。 リゾルバや DNS キャッシュが使うためのネームサーバのリストを動的に管理 する方法についての長年の問題を解決します。 resolvconf は自身をネットワークインターフェイスを制御するプログラムと ネームサーバ情報を供給するプログラム、そしてネームサーバ情報を必要とする アプリケーションの間の仲介人として設定します。

`resolvconf` は手動の設定を必要とせずに動作するように 設計されています。しかしながら、パッケージはまだ非常に新しく、 適切に動作させるためには、まだ手動の介入をいくらか必要とするかもしれません。 あなたが `/etc/resolv.conf` を更新するようにパッケージを カスタマイズしたならば、これは確実に当てはまります。ですので、 カスタマイズを無効にする必要があります。 詳細は `/usr/share/doc/resolvconf/README.gz` をごらんください。

10.4.3 検索した名前をキャッシュする – `nscd`, `dnsmasq`, `pdnsd`, `bind9`

もしネームサーバの反応が遅い場合は `libc6` リゾルバ を用いて検索される結果を `nscd` を用いてキャッシュしたくなる でしょう。

もしあなたのローカルネットの他のホストのために検索結果をキャッシュしようとするなら、`dnsmasq` や `pdnsd` のようなキャッシュフォワーディングネームサーバを使うのがいいでしょう。

`bind9` の `named` もキャッシュフォワーディングネームサーバとして使えない事はありません。ただ重いプログラムなので、その先進的機能を使わないなら、先程述べた他のプログラムを使う方が賢明です。`resolvconf` と連係しうまく動きます。

10.4.4 ドメインネームサービスを供給する – `bind`

あるドメインのための信頼できるネームサービスを供給する必要がある場合、`bind9` パッケージによる `named` のような完全に一人前のネームサーバが必要です。新たにインストールする場合、`bind9` がお薦めです。

`bind9` をインストールした際には、`dnsutils` もインストールしましょう。次に、`bind9-host`, `dns-browse`, `dnscvsutil`, `nslint` のようなユーティリティパッケージもインストールしたいかもしれません。さらに、`bind9-doc` のようなドキュメントパッケージもインストールしたいかもしれません。そして、`libbind-dev` や `libnet-dns-perl` のような開発用パッケージもインストールしたいかもしれません。DHCP を用いたインターフェイスを設定する場合、`dhcp-dns` パッケージを見つけるかもしれません。

基本的な設定を行うには、`bind9` をインストールするか、`dpkg-reconfigure` を実行しましょう。設定は `named.conf` を編集することです。Debian では、このファイルは `/etc/bind/` にあり、主に基本的な DNS ゾーンの定義のために用いられます。このファイルには `named.conf.local` というローカルゾーンを定義するファイルと、`named.conf.options` という設定オプションのために用いられるファイルという二つの別ファイルが含まれます。(後者は `/var/run/bind/named.options` ファイルを生成するために `resolvconf` によって処理されます。この生成ファイルは `forwarders` 指定が現在利用できるローカルではないネームサーバのリストであるという点を除いてオリジナルと同じです。これを利用するには、`named.conf` の `include` 行が `/var/run/bind/named.options` を含むように変更してください。'ネームサーバ情報の管理 – `resolvconf`' on page 187 をごらんください。)

完全なパス名無しで `named.conf*` という名前のデータベースファイルが `/var/cache/bind/` に保存されます。これは `named` により生成されるファイルを保存する正しい場所です。例えば、デーモンが動いているセカンダリゾーン用のデータベースファイルです。`/etc/bind/` にある静的なデータベースファイルは、フルパス名により `named.conf` にあるものと参照されなくてはなりません。詳細は `/usr/share/doc/bind9/README.Debian.gz` をごらんください。

10.5 DHCP を用いてネットワークインターフェイスを設定する

低レベルのネットワークインターフェイスの設定は Dynamic Host Configuration Protocol (DHCP) を用いて自動化できます。ファイアーウォールやルータボックス、そしてあなたが

使っているブロードバンド ISP は IP アドレスや他のパラメータをこの方法で提供しているかもしれません。

DHCP を動かすには、次のパッケージのうち一つをインストールする必要があります。

- `dhcp3-client` (version 3, Internet Software Consortium による)
- `dhcpcd` (Yoichi Hariguchi と Sergei Viznyuk による)
- `pump` (Red Hat による)

`pump` はシンプルで広く使われています。 `dhcp3-client` は複雑ですがより広範な設定が可能です。[5]

10.6 Debian での高レベルなネットワーク設定

10.6.1 `ifupdown`を用いた高レベルなネットワーク設定

ネットワーク設定をより容易にするため、Debian は `ifup` と `ifdown` プログラムと `/etc/network/interfaces` ファイル から構成される標準の高レベルネットワーク設定ツールを供給します。[6] ネットワークの設定に `ifupdown` を使おうとした場合、 通常低レベルのコマンドを同時に使うべき **ではありません**。 これは、低レベルな設定ツールを呼び出す `whereami,divine,intuitively` などのような他の 高レベルな設定ツールを使うべきではないということも意味しています。 `ifupdown` プログラムはこのプログラム単独でネットワーク インターフェイスの設定や設定解除に利用されるという意図を持って書かれました。

インターフェイスの設定を更新するには、次のようにします。

```
# ifdown eth0
# editor /etc/network/interfaces  # お好きなようにいじる
# ifup eth0
```

より詳しい情報は `interfaces(5)`, `/usr/share/doc/ifupdown/examples/network-interfaces.gz`,と `ifup(8)` をごらんください。

スタティック IP アドレスを用いたインターフェイスの設定

`192.168.0.111` の固定 IP アドレスを持った Ethernet インタフェース の設定を行いたいと仮定します。このアドレスは `192.168.0` から始まるので、LAN 上にある必要があります。さらに、LAN のインターネットへのゲートウェイの アドレスを `192.168.0.1` と仮定します。 次のような行を含むように `/etc/network/interfaces` を編集しましょう。

---

[5] 2004年4月時点では、`dhcp-client` パッケージも利用可能です。 これには ISC DHCP クライアントの version2 が含まれています。 これは現在 `dhcp3-client` に収録されている version 3 により 置き換えられます。 メンテナは `dhcp3-client` を Sarge 以降に `dhcp-client` に改名しようと計画しています。 experimentalバージョンの `dhcp-client` がインストールされてない事を確認して下さい。`ifupdown` がうまく動きません。

[6] 最新のバージョンの `/etc/network/interfaces` の ファイルフォーマットは Potato 以前のフォーマットと少し互換性がありません。 `ifupdown` のインストール後スクリプトが必要な場合に 自動的にこのファイルを更新します。

```
iface eth0 inet static
       address 192.168.0.111
       netmask 255.255.255.0
        gateway 192.168.0.1
```

“up” や “down” 行の適切なコマンドを指定することにより、インターフェイスの他の面についての設定ができますし、インターフェイスを立ち上げた後又は落とした後に他のアクションを取ることもできます。

```
iface eth0 inet static
   address 192.168.0.111
        netmask 255.255.255.0
        gateway 192.168.0.1
        up route add -net 10.0.0.0 netmask 255.0.0.0 gw 192.168.0.2 dev $IFACE
   down route del -net 10.0.0.0 netmask 255.0.0.0 gw 192.168.0.2 dev $IFACE
        up echo Interface $IFACE going up | /usr/bin/logger -t ifup
        down echo Interface $IFACE Going down | /usr/bin/logger -t ifdown
```

その代わりに、`/etc/network/if-up.d` や `/etc/network/if-down.d` ディレクトリにスクリプトを挿入できます。そのようなスクリプトは拡張オプションも実装できます。詳細は `interfaces(5)` をごらんください。例えば、`resolvconf` パッケージには インターフェイスを立ち上げるときに `/etc/resolv.conf` に含まれるような DNS 情報を指定するオプションを追加できます。

```
iface eth0 inet static
        address 192.168.0.111
        netmask 255.255.255.0
        gateway 192.168.0.1
        dns-search somedomain.org
        dns-nameservers 195.238.2.21 195.238.2.22
```

`dns-search` オプションの `somedomain.org` 引数は `resolv.conf(5)` にある `search` オプションの引数に対応します。`dns-nameservers` オプションの `195.238.2.21` および `195.238.2.22` 引数は `nameserver` オプションの引数に対応します。他の利用可能なオプションとしては、`dns-domain` と `dns-sortlist` があります。‘ネームサーバ情報の管理 – `resolvconf`’ on page 187 をごらんください。

DHCP を用いてインターフェイスを設定する

DHCP を用いてインターフェイスを設定するには、次のような行を含めるように `/etc/network/interfaces` を編集してください。

```
iface eth0 inet dhcp
```

これを機能させるには、‘DHCP を用いてネットワークインターフェイスを設定する’ on page 188 で言及した DHCP クライアント のうちいずれかをインストールする必要があります。

無線 LAN インターフェイスの設定を行う

`wireless-tools` パッケージには、インターフェイスを 立ち上げる前に無線 LAN (802.11a/b/g) ハードウェアの設定を可能とする フックスクリプト `/etc/network/if-pre-up.d/wireless-tools` が含まれます。 設定は `iwconfig` プログラムを通じて行われます。 `iwconfig(8)` をごらんください。 `iwconfig` に与えることが可能なコマンドパラメータそれぞれ に対して、“wireless-” 接頭辞を付けたパラメータのような名前で `/etc/network/interfaces` 内にオプションとして含めることが できます。 例えば、DHCP を用いて `eth0` を立ち上げる前に `eth0` の ESSID を `myessid` に、暗号鍵を `123456789e` に設定するには、次のような行を含むように `/etc/network/interfaces` を編集しましょう。

```
iface eth0 inet dhcp
        wireless-essid myessid
        wireless-key 123456789e
```

このインターフェイスに対して `waproamd` を走らせている場合、 ESSID と暗号鍵を設定するこの方法を用いるべきではありません。 `ifup` が起動された時、`waproamd` は既に ESSID と暗号鍵を設定済みです。‘ネットワーク設定のトリガを設定する – `waproamd`’ on page 199 をごらんください。

PPP インターフェイスの設定を行う

`ifup` と `ifdown` プログラムは PPP インターフェイスの追加と削除のために `pon` と `poff` を使います。ですから、まずは ‘PPP インターフェイスの設定’ on page 182 を読みましょう。

peer `myisp` に対して動くように PPP を設定する必要があると仮定します。 次のような行を含むように `/etc/network/interfaces` を 編集しましょう。

```
iface ppp0 inet ppp
        provider myisp
```

この一区切りが正しい位置にあれば、`ifup ppp0` は次を実行します。

```
pon myisp
```

不運なことに、`/etc/network/interfaces` にある `ppp` 節に `pppd` オプションをさらに供給することはいまのところ できません。[7]

---

[7] bug #196877 (`http://bugs.debian.org/196877`) 参照。

現在 PPP インターフェイスの補助的な設定を行うために `ifupdown` を使うことはできません。`pon` は `pppd` が接続を 確立する前に存在するため、`ifup` は インターフェイスの 準備ができる前に up スクリプトを起動します。 このバグ[8] が修正されるまでは、`/etc/ppp/ip-up` 又は `/etc/ppp/ip-up.d/` にある補助的な設定を実行する必要性が 残ります。

PPPoE インターフェイスの設定を行う

たとえ、カスタマのマシンが Ethernet や ATM ネットワーク経由でつないでいた としても、多くのブロードバンドインターネットサービスプロバイダ (ISP) は PPP を接続のネゴシエイトを行うために使用します。 これは PPP over Ethernet (PPPoE) により実現されます。この技術は PPP ストリームを Ethernet フレーム内にカプセル化する技術です。 ISP が *`myisp`* と呼ばれていると仮定します。 まず peer *`myisp`* のための PPP と PPPOE を設定します。 最も簡単な方法は `pppoeconf` パッケージをインストールし、 コンソールから `pppoeconf` を起動することです。 次に、下に示す行を含むように `/etc/network/interfaces` を編集します。

```
iface eth0 inet ppp
        provider myisp
```

時々 PPPoE over DSL を用いた時の Maximum Transmit Unit (MTU) についての問題が発生します。 詳細は DSL-HOWTO (`http://www.tldp.org/HOWTO/DSL-HOWTO/`) をごらんください。

ブロードバンドモデムがルータ機能を持っている場合、モデム/ルータは それ自身で PPPoE 接続を操作し、LAN 側では単なる インターネットへの Ethernet ゲートウェイにしか見えないことに注意してください。

ゲートウェイ用に複数の Ethernet インターフェイスを設定する

`eth0` が DHCP により設定した IP アドレスを用いてインターネット に接続し、`eth1` が固定 IP アドレス `192.168.1.1` を用いて LAN に接続していると仮定します。 次に示すような行を含むように `/etc/network/interfaces` を編集しましょう。

```
iface eth0 inet dhcp

iface eth1 inet static
        address 192.168.1.1
        netmask 255.255.255.0
```

'ゲートウェイルータを構築する' on page 203 に記述しているようにホストの NAT を有効にしている 場合、LAN 上の全てのホストでインターネット接続を共有できます。

---

[8] bug #127786 (`http://bugs.debian.org/127786`) 参照

仮想インターフェイスを設定する

仮想インターフェイスを用いると、1つの Ethernet カードを複数の IP サブネットへのインターフェイスとして設定できます。例えば、ホストが 192.168.0.x/24 の LAN ネットワーク上にあると仮定します。ホストを既存の Ethernet カードを用いて DHCP を用いて供給されるパブリックな IP アドレスを用いてインターネットに接続したい場合、次に示すような行を含むように `/etc/network/interfaces` を編集しましょう。

```
iface eth0 inet static
        address 192.168.0.1
        netmask 255.255.255.0
        network 192.168.0.0
        broadcast 192.168.0.255

iface eth0:0 inet dhcp
```

インターフェイス `eth0:0` は仮想インターフェイスです。これが立ち上がると、親の `eth0` も立ち上がります。

10.6.2 `ifupdown`論理インターフェイス定義を用いた高レベルなネットワーク設定

次の章では、読者は **物理** インターフェイスと **論理** インターフェイスの違いを理解することが重要になります。[9] **物理**インターフェイスは私達が通常“インターフェイス” と呼んでいる物でカーネルによって `eth0`,`ppp1` などと名付けられるものです。**論理**インターフェイスは物理インターフェイスの変数パラメータに割り当てられる値のセットのことです。この説明が分かりにくいなら、“論理インターフェイス *X* として設定する” という表現を “インターフェイスプロファイル *X* を用いて設定する” に置き換えます。

`/etc/network/interfaces` にある `iface` 定義は実際は物理インターフェイスの定義ではなく、論理インターフェイスの定義です。[10] インターフェイスを絶対再設定したくない場合、物理インターフェイス *X* は標準では論理インターフェイス *X* として設定されるので無視できます。

しかしながら、あなたのコンピュータが仕事場と家の間を持ち運ぶノート PC であると仮定しましょう。コンピュータを職場のネットワークに接続した時と家庭内 LAN に接続したときでは、接続する場所に従い `eth0` を再設定する必要があります。

まず (以前行ったように `eth0` ではなく) 二つの論理インターフェイス `home` と `work` を定義し、インターフェイスが家庭内 LAN と職場ネットワークのために設定されるかをそれぞれ記述します。

---

[9] この技術は `ifupdown` のドキュメントで使用されています。

[10] `auto` 行にあるインターフェイス名は物理インターフェイスで無ければいけません。論理インターフェイスではありません。

```
iface home inet static
        address 192.168.0.123
        netmask 255.255.255.0
        gateway 192.168.0.1

iface work inet static
        address 81.201.3.123
        netmask 255.255.0.0
        gateway 81.201.1.1
```

そして、物理インターフェイス`eth0`をコマンドラインで、次のように適切な設定ファイルを指定して立ち上げます。

```
# ifup eth0=home
```

職場ネットワークのために`eth0`を再設定するには、次のコマンドを実行します。

```
# ifdown eth0
# ifup eth0=work
```

上記で書かれた`interfaces`ファイルを用いると、もはや`ifup eth0`を用いるだけで`eth0`を立ち上げることができることに注意してください。この理由としては、`ifup`は物理インターフェイスの名前を標準の論理インターフェイスの名前として使用しており、私達の例では`eth0`という論理インターフェイスを定義していないためです。

10.6.3 **`ifupdown`**を用いた魔法のようなネットワーク設定

インターフェイス名は`ifup`が起動する時に他の名前に“マップ”できます。どのような名前にマップされるかは状況に依存して変更できます。それゆえ、`ifup`は前もって定義された候補の中から適切な論理インターフェイスとして与えられた物理インターフェイスを立ち上げるように設定できます。

論理インターフェイスの名前のマッピングは次のように発生します。

- 論理インターフェイス名が`ifup`コマンドラインに与えられていない場合、物理インターフェイス名は論理インターフェイス名の初期値として使用される。
- 論理インターフェイス名が`mapping`節の glob パターンにマッチする場合、新しい論理インターフェイス名の生成にマッピングが適用される。これは各マッピング節に対して順々に行われる。
- 最後の論理インターフェイス名が`/etc/network/interfaces`に定義されている論理インターフェイスの定義のラベルである場合、物理インターフェイスが論理インターフェイスとして立ち上げられる。上記に当てはまらない場合は、`ifup`が“Ignoring unknown interface”というメッセージを表示して終了する。

`mapping`行の文法は次のとおりです。

```
mapping glob-pattern
         script script-name
        [map script input]
```

`mapping` 行で示されたスクリプトは **物理** インターフェイス名を引数にし、“map” 行の “map” に続く (“map”自身を除く) 全ての内容を標準入力に送り込み 毎回起動されます。 スクリプトは終了前に標準出力にマッピングの結果を表示します。

例えば、次の `mapping` 行は `eth0` を home 論理インターフェイスとして `ifup` を立ち上げさせます。

```
mapping eth0
        script /usr/local/sbin/echo-home
```

ここで `/usr/local/sbin/echo-home` の内容は次のようなものです。

```
#!/bin/sh
echo home
```

マッピングはスクリプトを用いて行われるので、ある種のテストに基づいて 論理インターフェイスを選択することが可能です。 この例は ‘`guessnet`を使った論理インターフェイス選択’ on page 195 を参照ください。

`guessnet`を使った論理インターフェイス選択

`guessnet` をインストールして `/etc/network/interfaces` に次のような行を 追加しましょう。

```
mapping eth0
        script guessnet-ifupdown
        map home
        map work
```

`ifup eth0` すると、`guessnet` は `eth0` が home と work のどちらで立ち上げられたかをチェックします。これには、論理インターフェイスの定義内に保存された情報を 用います。

`laptop-net`を使った自動ネットワーク設定

`laptop-net` パッケージはネットワークの再設定に 異なる手法を取っています。laptop-net は `ifupdown` の 論理インターフェイスを利用していませんが、その代わり、固有の 設定 “スキーム” とシステム “プロファイル” というシステムを持っています。 しかしながら、物理インターフェイスを設定するために、laptop-net は依然 `ifup` と `ifdown` を使用しています。 詳細な情報は、`laptop-net-doc` にある良質な情報に 当たってください。

10.6.4 `network-manager` を用いた 自動ネットワーク設定

`network-manager` プログラムは現在 Fedora 開発者により 開発中であり、Ubuntu 用にパッケージされています。いつか Debian に表れて ifupdown や類似品を時代遅れにするかもしれません。

10.7 カーネルによる一貫性のないインターフェイスの命名を扱う

`eth0`,`eth1` などの名前は kernel がインターフェイスを作成する 順番に従い kernel により割り当てられます。 一方、ブート時に検知されるアダプタは通常毎回同じ順番で検知され、それゆえ 毎回同じ名前が割り当てられます。 ホットプラグされたアダプタにはこれは当てはまりません。ホットプラグされた アダプタはあらゆる順序で検知され、異なる状況下では kernel により結局異なる 名前が割り当てられる結果となります。

現行の`udev`パッケージではコンフィギュレーション によりこの問題を解決できます。

10.8 ネットワーク設定のトリガを設定する

インターフェイスがどのように設定や再設定されるのかを見てきました。 これらは適切な時期に行われる必要があります。

伝統的にネットワークは `/etc/rcS.d/S40networking` initscript 経由でブートシーケンスの間に設定され、滅多に再設定されませんでした。 ネットワークに依存するサービスはブートシーケンスで後に起動されました。 シャットダウン又はリブート時に initscript は逆の順番で起動されました。

しかしながら、現在 GNU と Linux において動的に変化するハードウェアと状況 のサポートがトレンドです。前者のサポートはホットスワップ可能な PCMCIA カードのために追加されました。より最近では、`hotplug` 機構が 追加され、多くの周辺機器がコンピュータの起動中に抜き差しできるように なりました。 これにはネットワーク関連のハードウェアが含まれます。 ハードウェアに依存するホットスワップされるハードウェアはハードウェアが挿入された後に起動され、ハードウェアが外された時に停止する必要がある ことに注意してください。 これはこの種のサービスが System V init システムの制御から外され、 その代わりに `ifupdown` の制御下に置かれる必要がある ことを意味しています。

例えば、initscript `/etc/init.d/foo` により制御される サービス `foo` が動的に再設定されるネットワークインターフェイス `eth0` に依存するとします。

- まず init システムの制御から `foo` を取り除きます。 `sysv-rc` init システムを利用している場合、 次を行います。 [11]

```
# rm /etc/rc[2345].d/S??f00
```

- 次に foo initscript を呼び出す `up` と `down` オプションを `/etc/network/interfaces` にある `eth0` 行に追加することにより `foo` を `ifupdown` の制御の元に置きます。

---

[11] 詳細は 'ランレベル' on page 21 を参照下さい。

```
        iface eth0 inet dhcp
                up /etc/init.d/foo start
                down /etc/init.d/foo stop
```

10.8.1 ブート時にネットワーク設定のトリガーを設定する

ブート時に `/etc/rcS.d/S40networking` init スクリプトは `ifup -a` コマンドを起動します。 これは `/etc/network/interfaces` の `auto` 節にリスト されている全ての物理インターフェイスを立ち上げます。

最近動的な方法を用いてネットワーク設定を操作する方がより良いことがたびたびあります。 いったん動的に変化するハードウェアのサポートメカニズムが機能すると、 まるで動的であったかのように静的なハードウェアを扱うことがシンプルになります。 ブートはもう 1つの hotplug イベントとして扱えます。 ('ネットワーク設定のトリガを設定する – `hotplug`' on page 197 参照)

しかしながら、ほとんど全ての場合、少なくともループバックデバイス `lo` はブート時に立ち上げておきたいでしょう。 それゆえ、`/etc/network/interfaces` に次の行が含まれている ことを確認してください。

```
auto lo

iface lo inet loopback
```

ブート時に他の物理インターフェイスも立ち上げたい場合、 `auto` 行にさらに物理インターフェイスを追加できます。 `auto` 行に 決して PCMCIA インターフェイスを リストしてはなりません。PCMCIA `cardmgr` はブートシーケンス において `/etc/rcS.d/S40networking` が起動する時よりも後に 起動されます。

10.8.2 ネットワーク設定のトリガを設定する – **`hotplug`**

hot-plug のサポートを追加するには、`hotplug` パッケージ をインストールしてください。

ネットワークハードウェアはブート時や、カード (例えば PCMCIA カード) を マシンに挿入した時、 `discover` が起動され、必要なドライバ モジュールをロードした時にもホットプラグできます。

kernel が新しいハードウェアを検知すると、ハードウェアのドライバを初期化し、 設定を行うために `hotplug` プログラムを起動します。 その後、ハードウェアが削除されると、kernel は異なる環境変数設定を 用いて `hotplug` を再び起動します。 Debian では、`hotplug` が呼ばれると、`/etc/hotplug/` と `/etc/hotplug.d/` にあるスクリプトを起動します。 詳細は `hotplug(8)` をごらんください。

新たに挿入されたネットワークハードウェアはスクリプト `/etc/hotplug/net.agent` により設定されます。[12] PCMCIA ネットワークカードが挿入され、`eth0` が利用可能になる と

---

[12] `/etc/hotplug.d/net/` にインストールされたあらゆるフックスクリプトによっても設定されるかもしれません。 例えば `ifplugd` や `waproamd` パッケージはフックスクリプトをそこにインストールします。

仮定します[13]。

```
ifup eth0=hotplug
```

`/etc/network/interfaces` に論理インターフェイスの定義を追加するか、`hotplug` という名のマッピングをしなければ、このコマンドは何もしません。コマンドが `eth0` を設定するようにするには、次の行を `/etc/network/interfaces` に追加しましょう。

```
mapping hotplug
        script echo
```

'`ifupdown`論理インターフェイス定義を用いた高レベルなネットワーク設定' on page 193 で説明したように、これは次と等価になるように上に示したコマンドをマップします。

```
ifup eth0=eth0
```

もしインターフェイスをコントロールするために `hotplug` によってスタートした `ifplugd` や `waproamd` のインスタンスを用いている場合、このようなマッピング行を追加**しないでください**。

`eth0` のみを使いたく、他のインターフェイスをホットプラグ時に立ち上げたくない場合、次のように `echo` の代わりに `grep` を使いましょう。

```
mapping hotplug
        script grep
        map eth0
```

さらに小技を学ぶには、'`ifupdown` を用いた魔法のようなネットワーク設定' on page 194 と `/usr/share/doc/hotplug/README.Debian` をごらんください。

10.8.3 ネットワーク設定のトリガを設定する – `ifplugd`

`ifplugd` デーモンはハードウェアがネットワークに接続されているかどうかによりインターフェイスの立ち上げや切断を行います。このプログラムは Ethernet インターフェイスに接続されている通信中のケーブルや無線 LAN インターフェイスとつながったアクセスポイントを検知します (無線 LAN インターフェイスの場合は実際には `waproamd` を使うのでしょうが)。`ifplugd` がリンクの状態の変化を検知すると、標準ではインターフェイスに対して `ifup` や `ifdown` を呼び出します。

---

[13] `hotplug` はバージョン0.0.20040329-4あたりではここに記載されたのと異なった挙動をするモードに設定できる。そのようなモードの一つはいわゆる "all" モードで、`hotplug` は全てのホットプラグ可能なインターフェイスを起動します。そうのようなもう一つのモードはいわゆる "auto" モードで `hotplug` は `/etc/network/interfaces` の中に `auto` 行にリストされている場合にのみインターフェイスを起動します。このような特殊なモードでは `ifup` は `=hotplug` というサフィックス無しで起動されます。

10.8.4 ネットワーク設定のトリガを設定する – `waproamd`

`waproamd` デーモンは `ifplugd` と無線 LAN カードと一緒に使うように設計されている以外は同じです。無線 LAN ハードウエアがつながるアクセスポイントをアクティブにスキャンします。つながった場合には、`waproamd` は `ifup` を起動します。

もし `waproamd` を使っている場合には、無線 LAN カードは `waproamd` を通じて設定されます。つまり、`/etc/network/interfaces` の `wireless-*` オプションは使いません。

10.8.5 ネットワーク設定と PCMCIA

PCMCIA ネットワークインターフェイス (2.4 と 2.6 カーネル) を設定するための可能なアプローチはいくつか存在します。

- 32 ビット PCI (CardBus) PCMCIA ネットワークカード用
  - `ifupdown` が `hotplug` によりコントロールされる。
    - Woody と Sarge 'ネットワーク設定のトリガを設定する – `hotplug`' on page 197 に記述されているように、`/etc/network/interfaces` にマッピング行を追加することにより、`ifupdown` への `hotplug` の制御をローカルで有効にする必要があります。
- 16 ビット PCMCIA ネットワークカード用
  - モジュールのロードを制限された `pcmcia-cs` を用いて `hotplug` により制御された `ifupdown`
    - **推奨**
    - Woody と Sarge `/etc/pcmcia/network` の先頭に `exit 0` を加えることにより、ローカルで無効にする必要があります。また、'ネットワーク設定のトリガを設定する – `hotplug`' on page 197 に記述されているように、`/etc/network/interfaces` にマッピング行を追加することにより、`ifupdown` への `hotplug` の制御をローカルで有効にする必要があります。
  - 標準の `/etc/pcmcia/network` 経由で `pcmcia-cs` により制御される `ifupdown`
    - **時代遅れ** だが Woody と Sarge では依然標準
  - `/etc/pcmcia/network.opts` にある特別なコードを経由して `pcmcia-cs` により制御される低レベルツール
    - **時代遅れ**
    - Woody および Sarge `/etc/pcmcia/network.opts` を編集することにより特別なコードが有効にされます。

16 ビットカードに関する推奨アプローチは Linux 2.4 のホットプラグサブシステムが PCMCIA もサポートするようになった事を利用しています。[14] PCMCIA ネットワーク

---

[14] 過去の Debian リリースでは、PCMCIA ネットワークカードの標準的な設定方法は `cardmgr` を通じた `/etc/pcmcia/network` と `/etc/pcmcia/network.opts` フックスクリプトでした。これらのフックスクリプトはより一般的な目的の hotplug 能力が得られた Linux の前の時代に開発されました。標準状態では、インターフェイスが追加されると `ifup` を、インターフェイスが削除されると `igdown` を単に呼び出すだけのこれらのスクリプトをまだ使っている人がいます。上に述べたように、今はインターフェイスの追加削除の作業を行うには、`hotplug` を使うことを推奨します。`/etc/pcmcia/network.opts` にある特定の変数が `y` にセットされると有効にされる低レベルなネットワーク設定コマンドを呼ぶ Debian Woody の特別なシステムを利用する人もまだいます。このシステムには複数の問題があります。問題としては、競合条件による不具合や、16 ビッ

カードはホットプラグ可能です。従って、PCMCIA カードを通じてネットワークを要求するあらゆるサービスは カードの挿入時にサービスを開始し、カードを抜いた時にサービスを停止 するように設定されなければなりません。これは通常サービスが `ifup` で起動され、`ifdown` で停止される ように調整することにより実現されます。しかしながら、PCMCIA ネットワークカードをコールドプラグするように 設定するのを選択する人も中にはいます。システムをブートする前にカードを挿入し、ブートシーケンス中で カードを通じたネットワークを要求するサービスを起動するのです。サービスの開始前にカードが完全に設定されるのを保証するため、次に示す設定を行う必要があります。

- `cardmgr` がフォアグラウンドで走るように `/etc/default/pcmcia` 中で `CARDMGR_OPTS="-f"` と設定します。
- `/etc/rc?.d/S20pcmcia`を `/etc/rc?.d/S12pcmcia` などとリネームします。

16 ビットの PCMCIA カードを使っている場合は`pcmcia-cs` がまだ必要なことに注意して下さい。このパッケージに含まれる `cardmgr` デーモンは ソケットの管理やドライバーモジュールの起動に必要です。ネットワークの設定を `/etc/pcmcia/network` を通して行いたく無いだけです。

`cardmgr`が正常に動作するには、16 ビット PCMCIA カードにリソースを割り当てるよう、`/etc/pcmcia/config.opts` を編集する必要があるかもしれません。より詳しい情報は Linux PCMCIA HOWTO (`http://www.tldp.org/HOWTO/PCMCIA-HOWTO.html`) をごらんください。

10.9 多段マッピング

あなたのネットワークカードがホットプラグされ‘ネットワーク設定のトリガを設定する – `hotplug`’ on page 197 に書かれている 自動設定が有効になっていると仮定しましょう。さらに論理インターフェイスを、(‘カーネルによる一貫性のないインターフェイスの命名を扱う’ on page 196 に書かれたように)アダプタ固有の 物理インターフェイスの特徴に従かうか、(例えば ‘`guessnet`を使った論理インターフェイス選択’ on page 195 に書かれたように) インターフェイスがつながれているネットワークの特徴に従って「物理」インターフェイスにマップする必要が有ると仮定しましょう。

第一段階のマッピングは `hotplug` のグループ名が 与えられると、インターフェイスがホットプラグされると kernel が割り当てたしたインターフェイス名を出力します。第二段階のマッピングは kernel が割り当てたインターフェイス名が 与えられると、アダプタの名前を出力します。第三段階のマッピングはアダプタの名前が与えられると、 ネットワーク環境に基づく論理インターフェイス名を出力します。

```
# hotplug にインターフェイスを起動させる
mapping hotplug
        script echo

# インターフェイスが有線か無線 LAN かを決定
```

ト PCMCIA カードしか動かないこと、`ifupdown` が行うために残した方が良いことをやってしまうことがあります。結果としてこれは時代遅れです。

```
mapping eth?
        script /usr/local/sbin/get-mac-address.sh
        map 02:23:45:3C:45:3C wired
        map 00:A3:03:63:26:93 wifi

# どの有線ネットワークが使えるかを決める
mapping wired
        script guessnet-ifupdown
        map work-wired
        map home

# どの無線 LAN ネットワークが使えるかを決める
mapping wifi
        script ifscout
        map starbucks
        map work-wireless

iface work-wired inet static
        ...
```

10.10 ネットワークサービスの設定

デスクトップやホームサーバ環境における典型的なネットワークサービスの設定には 次が挙げられます。

- インターネット スーパーサーバ と TCP/IP デーモンラッパ。'サービスへのアクセスを制限する' on page 136 参照。
  - `/etc/inetd.conf`
- `ssh`: OpenSSH secure shell。'SSH' on page 157 参照。
  - `/etc/ssh/ssh_config`
  - `/etc/ssh/sshd_config`
- `exim`: メール転送エージェント。 'Mailname' on page 186 および 'Mail transport agents (MTAs)' on page 161 参照。
  - `/etc/exim/exim.conf`
  - `/etc/mailname`
  - `/etc/aliases`
  - `/etc/email-addresses`
- `fetchmail`: POP3 アカウントからメールを取得。'メールを取得する – Fetchmail' on page 164 参照。
  - `/etc/fetchmailrc`
- `procmail`: ローカルのメール配達およびフィルタプログラム。'メール処理 – Procmail' on page 165 参照。
  - `~/.procmailrc`
- ホスト名と DNS (プロキシ、キャッシュなど)。'ホスト名' on page 185 と'ドメインネームサービス (DNS)' on page 186 参照。

  - `/etc/host.conf`
  - `/etc/hostname`
  - `/etc/hosts`
  - `/etc/hosts.allow`
  - `/etc/hosts.deny`
  - `/etc/resolv.conf`
  - `/etc/bind/named.conf` (編集)
  - `/etc/bind/db.lan` (LAN ホスト用に追加)
  - `/etc/bind/db.`*`192.168.0`* (LAN の逆引き用に追加)
- DHCP。'DHCP を用いてネットワークインターフェイスを設定する' on page 188 参照。
  - `/etc/dhcp3/dhclient.conf` (DHCP クライアント)
  - `/etc/default/dhcp3-server` (DHCP サーバ)
  - `/etc/dhcp3/dhcpd.conf` (DHCP サーバ)
- `cvs`: バージョン管理システム。'Concurrent Versions System (CVS)' on page 213 参照。
  - `/etc/cvs-cron.conf`
  - `/etc/cvs-pserver.conf`
- `nfs-kernel-server`: ネットワークファイルシステム。 'NFS 設定' on page 41 参照。(unix ライクシステム用)
  - `/etc/exports`
- `samba`: Windows 用のネットワークファイルシステムおよびプリンタ共有。'Samba の設定' on page 41 および 'Samba' on page 129 参照。
  - `/etc/samba/smb.conf`
- プリンタデーモンシステム。'プリンタの設定' on page 42 参照。
  - `/etc/printcap` (lpr 用)
- `apache` と `apache2` ウェブサーバ。
  - `/etc/apache/*`
  - `/etc/apache2/*`
- `squid`: ウェブプロキシキャッシュサーバ。
  - `/etc/squid/*`

10.11 ネットワークのトラブルシュート

ネットワークに関する問題に直面したら、次の出力を最初の真偽の確認として チェックしてください。

```
# ifconfig
# cat /proc/pci
# cat /proc/interrupts
# dmesg | more
```

又、'ネットワークテストの基礎' on page 125 をごらんください。

特定のウェブサイトを見れない問題に直面した場合は、'いくつかのウェブサイトにアクセスした時の奇妙な挙動' on page 45 をごらんください。

10.12 ゲートウェイルータを構築する

Debian ホストはネットワークアドレス変換 (NAT、マスカレードとしても知られる) 、メール転送、DHCP、DNS キャッシング、HTTP プロキシキャッシング、CVS サービス、NFS サービス、そして Samba サービスを行う多目的の ゲートウェイマシンにできます。これらの設定の例は 'LAN を使用するためのホストと IP' on page 31 をごらんください。

10.12.1 Netfilter 設定

netfilter/iptables プロジェクトは Linux 2.4 以降のための ファイアーウォールサブシステムです。 Netfilter (`http://www.netfilter.org/`) をごらんください。 この文書では多くのネットワーク設定例が説明されています。

netfilter の基礎

Netfilter は五つの組込みチェイン、すなわち PREROUTING, INPUT, FORWARD, OUTPUT そして POSTROUTING を使うパケットを処理します。

```
                ルーティング
                方向
IN ------> PRE ---> ------> FORWARD -----> ----> POST -----> OUT
interface   ROUTING \       filter       /       ROUTING     interface
              DNAT      |     tracking    ^       SNAT
            REDIRECT  |                   |        MASQUERADE
                              v                    |
                          INPUT                OUTPUT
                             | filter               ^ filter,DNAT
                             v                      |
                           \--> ローカルのプロセス --/
                                ユーザスペースのプログラム
```

Netfilter テーブル

パケットはそれぞれの組込みチェインにて次のテーブルを用いて処理されます。

- フィルタ (パケットフィルタ、標準)
  - INPUT (ルータ自体に入り込むパケット用)
  - FORWARD (ルータを通ってルーティングされているパケット用)
  - OUTPUT (ローカルで生成されたパケット用)
- nat (ネットワークアドレス変換)
  - PREROUTING (入り込んですぐにパケットを変換するため)
  - OUTPUT (ルーティング前にローカルで生成されたパケットを変換するため)
  - POSTROUTING (出て行こうとするパケットを変換するため)
- mangle (ネットワークアドレス変換、kernel 2.4.18 以降のみでうまく動く)
  - 全ての組込チェインが利用可能

Netfilter のターゲット

ファイアーウォールのルールにはいくつかのターゲットがあります。

- 四つの基本的なターゲット
  - ACCEPT はパケットの通過を許すことを意味する。
  - DROP はパケットを落とすことを意味する。
  - QUEUE はパケットをユーザスペースに通すことを意味する。(kernel がサポートしている場合)
  - RETURN はこのチェインの検討を止め、以前の (呼び出し元) チェイン内の 次のルールから検討を始めることを意味する。
- 拡張されたターゲット。
  - LOG は kernel ログに記録する。
  - REJECT はエラーパケットを送信し、そのパケットを落とす。
  - SNAT パケットの送信元アドレスを変換し、POSTROUTING チェインでのみ使用する (nat テーブルのみ)

    `--to-source ipaddr[-ipaddr][:port-port]`
  - MASQUERADE は SNAT と同じだが動的に割り当てられる IP (ダイアルアップ) 接続用 (nat テーブルのみ)

    `--to-ports port[-port]`
  - DNAT はパケットの送信先アドレスを変換し、PREROUTING と OUTPUT チェイン とこれらのチェインからのみ呼ばれるユーザが定義したチェインで使用される。 (nat テーブルのみ)

    `--to-destination ipaddr[-ipaddr][:port-port]`
  - REDIRECT はマシン自身にパケットを送るためにパケットの送信先 IP アドレスをマシン自身の IP アドレスに変換する。

    `--to-ports port[-port]`

Netfilter コマンド

`iptables` の基本的なコマンドは次のようになります。

```
   iptables -N chain                    #chain を作成

   iptables -A chain \                  # chain にルールを追加
            -t table \                  # table を                    使
用 (filter, nat, mangle)
            -p protocol \               # tcp, udp, icmp, 又はこれら全て
            -s source-address[/mask] \
            --sport port[:port] \       # -p が tcp 又は udp の場合は送信
元ポート
            -d destination-address[/mask] \
            --dport port[:port] \       # -p が tcp 又は udp の場合は送信
先ポート
            -j target \                 # マッチした場合の行動
            -i in-interface-name \      # INPUT,  FORWARD, PREROUTING 用
            -o out-interface-name       # FORWARD, OUTPUT, POSTROUTING 用
```

ネットワークアドレス変換

LAN 上のマシンは LAN 上の IP アドレスをインターネット上で使える IP アドレスに変換するゲートウェイを経由してインターネットのリソースにアクセスできます。

```
# apt-get install ipmasq
```

`ipmasq` の防護を強化するためにサンプルルールを追加してください。

又、ネットワークに PCMCIA NIC 経由でアクセスしている場合、`ipmasq` が `/etc/pcmcia/network.opts` (`/usr/share/doc/ipmasq/ipmasq.txt.gz` 参照) 又は `/etc/network/interfaces` ('ネットワーク設定と PCMCIA' on page 199 と 'ネットワーク設定のトリガを設定する' on page 196 参照) から起動される必要があります。

SMTP 接続をリダイレクトする (2.4 用)

他の LAN 環境で利用するように設定しているノート PC を持っており、再設定せずにノート PC 上でメーラを使いたいとします。

ゲートウェイマシンに `iptables` を通じて次のルールを追加し、SMTP 接続をゲートウェイマシンにリダイレクトします。

```
# iptables -t nat -A PREROUTING -s 192.168.1.0/24 -j REDIRECT \
           -p tcp --dport smtp --to-port 25 # smtp=25, INPUT is open
```

より徹底的なリダイレクトルールセットが必要な場合は、`ipmasq` パッケージをインストールし、`/etc/ipmasq/rules/` ディレクトリに `M30redirect.def`(`http://www.debian.org/doc/manuals/debian-reference/examples/`) を追加することを考えてください。

10.12.2 複数のネット接続を管理する

[FIXME] Policy routing (by Phil Brutsche `<pbrutsch@tux.creighton.edu>`): 詳細は iproute manual (`http://lartc.org/`) をごらんください。トラフィックコントロール (tc) にも興味が湧くかもしれません。

次のような環境を考えます。

```
eth0: 192.168.1.2/24; gateway 192.168.1.1
eth1: 10.0.0.2/24; gateway 10.0.0.1
このマシン上でマスカレードは行いません。
```

特別な魔法を施します。

1 ip rule add from 192.168.1.2 lookup 1

```
ip rule add from 10.0.0.2 lookup 2
ip route add to default via 10.0.0.1 metric 0
ip route add to default via 192.168.1.1 metric 1
ip route add table 1 to 192.168.1.0/24 via eth0
ip route add table 1 to 10.0.0.2/24 via eth1
ip route add table 1 to default via 192.168.1.1
ip route add table 2 to 192.168.1.0/24 via eth0
ip route add table 2 to 10.0.0.2/24 via eth1
ip route add table 2 to default via 10.0.0.2
```

[FIXME] これを試したことはありません。オートダイアル機能を使わないで高速な接続のバックアップとしてダイアルアップをどうやって設定しますか? パッチを送ってください。:)

章 11

エディタ

11.1 有名なエディタ

Linux はコンソールテキストエディタの多くの代替品を提供します。それらは:

- `vim`: 力強く軽量な BSD の遺産であるエディタ。VI iMproved.
- `emacs`: 究極かつ巨大な GNU の遺産であるエディタ。 RMS (Richard M. Stallman) オリジナル。
- `xemacs`: 元々は Lucid による Emacs: The Next Generation。
- `mcedit`: 新参の GNU エディタ。元々は `mc` の 内部組込みエディタ。'MC 内のエディタ' on page 52 をご覧ください。
- `ae`: Potato のデフォルトのエディタ。やめときましょう。
- `nano`: Woody のデフォルトのエディタ。`pico` をエミュレート。
- `joe`: WordStar または TurboPascal の古豪。
- `jed`: 高速かつフル機能を持った、Emacs のキーバインディングを持つ メニュ指向のエディタ。
- `jove`: Emacs キーバインディングを持つ非常に小さなエディタ。
- `nvi`: 新 vi。オリジナルの vi と同じバグを持つ。

デフォルトのエディタを変更するには、`update-alternatives --config editor`を 使用してください。又、使用するエディタを決定するために環境変数 `EDITOR` 並びに `VISUAL` を用いることには多くの問題があります。 'MC 内のエディタ' on page 52 をご覧ください。

また、数種の X ベースのテキストエディタも言及する価値があります:

- `gvim`: GUI 付きの Vim (`vim` および `vim-gtk` パッケージ)
- `emacs`: 唯一の真の Emacs (X を自動検知)。
- `xemacs`: 次世代の Emacs (X を自動検知)。

これらの xclient コマンドは `-fn` *`a24`* のような標準的な オプションを引数に取り、私のような古株にも優しく使えます :) 'X クライアント' on page 146 をご覧ください。

11.2 緊急用エディタ

`/bin/` にはいくつかのエディタが存在します。`/usr/` に アクセスできない場合にファイルの編集を容易にするため、これらのエディタのうち 一つをインストールすべきです。

- `elvis-tiny`: 最小限の vi エディタ (`vi` で起動)
- `nano-tiny`: 最小限の 非 vi エディタ (`nano-tiny` で起動)
- `nano`: 最小限の 非 vi エディタ (`nano` で起動) (Sarge)
- `ed`: 最小限のエディタ (常に存在するが使用しがたい)

11.3 Emacs と Vim

11.3.1 Vim の使用ヒント

Vim 起動中に **<F1>** を押すことにより、“VIM - main help file” ドキュメント をご覧ください。

`<F1>`	ヘルプ
`<esc>`	ノーマルモードに戻る
`V`	ビジュアルモード
`i`	挿入モード
`:`	コマンドライン
`:set tw=72`	テキスト幅を 72 に設定
`<F11>`	挿入(ペースト)モード
`:r! date -R`	RFC-822 日付を挿入
`q`*`a`*	レジスタ *a* にキーストロークを記録
`q`	キーストロークの記録を停止
`@`*`a`*	レジスタ *a* からキーストロークを実行
`:edit` *`foo.txt`*	*`foo.txt`* をロードし、ファイルを編集
`:wnext`	現在のファイルを保存し次のファイルを編集

`q` および `@` はシンプルなマクロ記録および再生に使用できます。 例えば、カーソル位置の単語に HTML イタリックタグを挿入するマクロを作成するには、`qii<i>^[ea</i>^[q` と入力してください。(ここで `^[` は ESC キー入力) そして単語の始まりで `@i` を入力すると、**<i>** および **</i>** タグが挿入されます。

‘Vim で GnuPG を使う’ on page 245 もご覧ください。

11.3.2 Emacs の使用ヒント

`<F1>`	ヘルプ
`<F10>`	メニュ
`C-u M-! date -R`	RFC-822 日付を挿入

11.3.3 エディタを起動

エディタを起動:	emacs filename	vim filename
vi 互換モードで起動:		vim -C
vi 非互換モードで起動:		vim -N
コンパイル時のデフォルトで起動:	emacs -q	vim -N -u NONE

11.3.4 エディタのコマンドのサマリ (Emacs, Vim)

終了:	C-x C-c	:qa /:wq /:xa /:q!
中断/コマンドモード:	C-g	<esc>
1字戻る(左):	C-b	h
1字進む(右):	C-f	l
1行進む(下):	C-n	j
1行戻る(上):	C-p	k
行の先頭に飛ぶ(^):	C-a	0
行の最後に飛ぶ($):	C-e	$
複数回コマンドを実行:	C-u nnn cmd	nnn cmd
複数のコマンド:	M-digitkey cmd	
ファイルの保存:	C-x C-s	:w file
バッファの先頭に飛ぶ:	M-<	1G
バッファの最後に飛ぶ:	M->	G
1スクリーン先にスクロール:	C-v	^F
半スクリーン先にスクロール:		^D
1行先にスクロール:		^E
1行後ろにスクロール:	M-v	^B
半スクリーン後ろにスクロール:		^U
1スクリーン後ろにスクロール:		^Y
他のウィンドウにスクロール:	M-C-v	
カーソルの下を削除:	C-d	x
カーソルから行末まで削除:	C-k	D
順方向インクリメンタル探索:	C-s	
逆方向インクリメンタル探索:	C-r	
順方向一括探索:	C-s enter	/
逆方向一括探索:	C-r enter	?
順方向正規表現探索:	M-C-s	
逆方向正規表現探索:	M-C-r	
順方向正規表現一括探索:	M-C-s enter	/
逆方向正規表現一括探索:	M-C-r enter	?
ヘルプ:	C-h C-h	:help
Apropos:	C-h a	
キーバインドヘルプ:	C-h b	:help [key]
Info ヘルプ:	C-h i	
メジャーモードヘルプ:	C-h m	
ヘルプチュートリアル:	C-h t	:help howto

Undo:	C-_	u
Redo:	C-f	^R
カーソル位置をマーク:	C-@	m{a-zA-Z}
マークした位置に移動、再入力で元に戻る:	C-x C-x	
現在のファイルのマークに飛ぶ:		'{a-z}
あらゆるファイルにおいてマークに飛ぶ:		'{A-Z}
領域コピー:	M-w	{visual}y
領域削除:	C-w	{visual}d
Yank:	C-y	
以前削除したバッファをYank:	M-y	p
領域をを大文字に変換:	C-x C-u	{visual}U
領域を小文字に変換:	C-x C-l	{visual}u
スペシャルキャラクタを挿入:	C-q octalnum/keystroke	^V decimal/keystroke
置換:	M-x replace-string	:%s/aaa/bbb/g
正規表現を用いた置換:	M-x replace-regexp	:%s/aaa/bbb/g
置換問合せ:	M-%	:%s/aaa/bbb/gc
置換問合せ:	M-x query-replace	
正規表現を用いた置換問合せ:	M-x query-replace-regexp	
ファイルを開く:	C-x C-f	:r file
ファイルをセーブ:	C-x C-s	:w
全バッファのセーブ:	C-x s	:wa
別ファイルにセーブ:	C-x C-w file	:w file
バッファのプロンプト:	C-x b	
バッファ表示:	C-x C-b	:buffers
読み込み専用にトグルする:	C-x C-q	:set ro
プロンプトを出し、バッファを削除する:	C-x k	
縦に2分割する:	C-x 2	:split
横に2分割する:	C-x 3	:vsplit (ver. 6)
他のウィンドウに移動する:	C-x o	^Wp
現在のウィンドウを削除する:	C-x 0	:q
他のウィンドウを削除する:	C-x 1	^Wo
シェルをバックグラウンド起動する:	M-x compile	
バックグラウンド起動したシェルを殺す:	M-x kill-compilation	
makeを起動:		:make Makefile
エラーメッセージを調べる:	C-x`	:echo errmsg
シェルを起動し、記録する:	M-x shell	:!script -a tmp
...clean BS, ...		:!col -b <tmp >record
シェルの記録を保存/再呼び出し:	C-x C-w record	:r record
シェルを起動する:	M-! sh	:sh
コマンドを起動する:	M-! cmd	:!cmd
コマンドを起動し、挿入する:	C-u M-! cmd	:r!cmd
フィルタを起動する:	M-| file	{visual}:w file
フィルタを起動し、挿入する:	C-u M-| filter	{visual}:!filter
オプションを表示する		:se[t] {option}?
オプションをデフォルトにリセットする		:se[t] {option}&

ブーリアンオプションをリセットする	`:se[t] no{option}`
ブーリアンオプションをトグルする	`:se[t] inv{option}`
`72`列でテキストをラップする	`:se tw=72`
ラップしない	`:se tw=0`
自動インデント	`:se ai`
タブを広げる	`:se et`
特別なコマンド `(mail)`	`:se comments=n:>,n:\\|`

`GDB` を起動	`M-x gdb`
`describe GDB mode`	`C-h m`
`1`行進める	`M-s`
次の行	`M-n`
`1`命令進める `(stepi)`	`M-i`
現在のスタックフレームを終了	`C-c C-f`
続行する	`M-c`
`up arg frames`	`M-u`
`down arg frames`	`M-d`
`copy number from point, insert at the end`	`C-x &`
ブレークポイントを設定	`C-x SPC`

11.3.5　Vim の設定

Vim の全機能と文法強調を使用するには、`~/.vimrc` または `/etc/vimrc` が次の行を含むようにしてください。

```
set nocompatible
set nopaste
set pastetoggle=<f11>
syn on
```

ペーストモードは端末でのカットアンドペースト操作にと自動インデントとの 干渉を避けることができます。これは単なる “:set noai” 以上のことを行います。

Vim への GnuPG の統合については ‘Vim で GnuPG を使う’ on page 245 をご覧ください。

11.3.6　Ctags

`apt-get install exuberant-ctags` を行い、ソースファイルにおいて `ctags` を起動します。*function_name* が始まる行に ジャンプするため、Vim 上で`:tag` *`function_name`* とタイプします。Tag は C、C++、Java、Python 並びに他の多くの言語に対して 働きます。

Emacs でも同様の Ctags を利用できます。

11.3.7 構文強調されたスクリーンからHTMLソースに変換

Vimのコマンドモードから`so \$VIMRUNTIME/syntax/2html.vim`を実行すると構文強調されたテキストからHTMLテキストへの変換が行えます。`:w file.html`および`:q`でセーブします。Cコードなどで役立ちます。

11.3.8 `vim`によりスクリーンを分割する

`vim`は複数に分割したスクリーン環境において複数のファイルを編集可能です。詳細は`:help usr_08.txt`とタイプしてみてください。

異なったファイルに対してスクリーンを分割するには、viコマンドプロンプトで次のようにタイプします。

```
:split another-file
:vsplit another-file
```

又はシェルプロンプトにおいて

```
$ vi -o file1.txt file2.txt   # 水平分割
$ vi -O file1.txt file2.txt   # 垂直分割
```

を実行することによりマルチウィンドウのviを実現できます。

```
$ vimdiff file.txt~ file.txt          # file.txt の最近の変更点を調べる
$ vimdiff file.en.sgml file.fr.sgml  # 翻訳版との相異点を調べる
$ gvimdiff file.txt~ file.txt         # X における上記コマンド
```

以上のコマンドによりオリジナルとバックアップファイル間の相異点をうまく表示できます。SGMLではタグとマッチするので、SGMLモードにおいて翻訳を比較するとうまく動きます。

Ctrl-Wコマンドを用いた特別なカーソル操作コマンドは次の通り:

```
Ctrl-W +      ウィンドウサイズを広げる
Ctrl-W -      ウィンドウサイズを狭める
Ctrl-W h      左のウィンドウに移動
Ctrl-W j      下のウィンドウに移動
Ctrl-W k      上のウィンドウに移動
Ctrl-W l      右のウィンドウに移動
...
```

スクリーンのスクロール制御には次のコマンドを使用してください。

```
:set scrollbind
:set noscrollbind
```

章 12

バージョン管理システム

12.1 Concurrent Versions System (CVS)

詳細な情報は、`lynx` を使って `/usr/share/doc/cvs/html-cvsclient`, `/usr/share/doc/cvs/html-info`, `/usr/share/doc/cvsbook` を見るか、 `info cvs` や `man cvs` を実行して下さい。

12.1.1 CVS サーバのインストール

次に示すセットアップにより、“src” グループのメンバのみがCVS リポジトリにコミットでき、“staff” グループのメンバだけが CVS の管理を行うように設定できます。これにより自爆する危険を低減できます。

```
# cd /var/lib; umask 002; mkdir cvs # [Woody] FSH
# apt-get install cvs cvs-doc cvsbook
# export CVSROOT=/var/lib/cvs
# cd $CVSROOT
# chown root:src .  # プロジェクトの開始をさらに制限するには "staff" にする
# chmod 3775 .             # 上記で "staff" を使っている場合は 2775 を使う
# cvs -d /var/lib/cvs init # ここでは明示的に -d を指定する方が安全
# cd CVSROOT
# chown -R root:staff .
# chmod 2775 .
# touch val-tags
# chmod 664 history val-tags
# chown root:src history val-tags
```

12.1.2 CVS セッションの例

次は CVS リポジトリへのアクセスのためのシェル環境をセットアップします。

Anonymous CVS (ダウンロードのみ)

読み込みのみのリモートアクセスのためには次を実行します。

```
$ export CVSROOT=:pserver:anonymous@cvs.sf.net:/cvsroot/qref
$ cvs login
$ cvs -z3 co qref
```

ローカル CVS サーバを使う

サーバと同じマシン上でシェルからローカルのアクセスを行うには次を実行します。

```
$ export CVSROOT=/var/lib/cvs
```

リモート CVS サーバを使う

SSH を用いずに (cvs の RSH プロトコル機能を利用して) リモートアクセスを行うには次を実行します。

```
$ export CVSROOT=:pserver:account@cvs.foobar.com:/var/lib/cvs
$ cvs login
```

これは盗聴される危険があります。

`ssh` を通じてリモート CVS にアクセスする

SSH を用いてリモートアクセスするには次を実行します。

```
$ export CVSROOT=:ext:account@cvs.foobar.com:/var/lib/cvs
```

又は SourceForge の場合では次を実行します。

```
$ export CVSROOT=:ext:account@cvs.sf.net:/cvsroot/qref
```

パスワードプロンプトを除外する RSA 認証 (‘パスワードの入力が少ない接続 – RSA’ on page 159) も利用できます。

新規 CVS アーカイブの作成

次に示すプロジェクトのためにアーカイブを作成するとして、

アイテム	値	意味
ソースツリー	*~/project-x*	全ソースコード
プロジェクト名	*project-x*	作成するプロジェクト名
ベンダータグ	*Main-branch*	ブランチ全体のタグ
リリースタグ	*Release-initial*	指定したリリースのためのタグ

次の手順によりアーカイブを作成できます。

```
$ cd ~/project-x                  # ソースディレクトリに飛び込む
 ... ソースツリーを作成する ...
$ cvs import -m "Start project-x" project-x Main-branch Release-initial
$ cd ..; rm -R ~/project-x
```

CVS を用いた作業

ローカルの CVS リポジトリを用いて *project-x* の作業を行うには、次のようにします。

```
$ cd                              # 作業エリアに移動
$ cvs co project-x                # CVS からソースを取得してローカルにコピー
$ cd project-x
 ... 内容をいろいろ変更する ...
$ cvs diff -u                     # diff -u repository/ local/ と同様の出力
$ cvs up -C modified_file         # ファイルへの変更を undo する
$ cvs ci -m "Describe change"     # ローカルのソースを CVS に保存
$ vi newfile_added
$ cvs add newfile_added
$ cvs ci -m "Added newfile_added"
$ cvs up                          # CVS から最新版をマージする
 ... CVS から新規に作成したサブディレクトリを作成するには、
 ... 代わりに "cvs up -d -P" を使います。
 ... "C filename" で始まる行を監視してください。
 ... 未修正のコードは `.#filename.version' に移動します。
 ... filename にある "<<<<<<<" と
 ... ">>>>>>>" を探してください。
$ cvs tag Release-1               # リリースタグを追加
 ... さらに編集を続ける ...
$ cvs tag -d Release-1            # リリースタグを削除
$ cvs ci -m "more comments"
```

```
   $ cvs tag Release-1             # リリースタグを再び追加
   $ cd                            # 作業エリアに戻る
   $ cvs co -r Release-initial -d old project-x
    ... オリジナルのバージョンを old ディレクトリに置いておく
   $ cd old
   $ cvs tag -b Release-initial-bugfixes # ブランチ (-b) タグを作成
    ... 旧バージョンで作業できるようになりました (Tag=sticky)
   $ cvs update -d -P              # 空のディレクトリは作成不可
    ... ソースツリーは今 "Release-initial-bugfixes" sticky タグを持つ
    ... このブランチで作業
   $ cvs up -d -P # このブランチの他者により修正されたファイルを同期
   $ cvs ci -m "check into this branch"
   $ cvs update -kk -A -d -P
    ... sticky タグを削除し、内容を忘れる
    ... キーワード拡張無しで main trunk から更新
   $ cvs update -kk -d -P -j Release-initial-bugfixes
    ... Release-initial-bugfixes ブランチから main にマージ
    ... キーワード拡張無しの trunk。エディタで衝突を修正
   $ cvs ci -m "merge Release-initial-bugfixes"
   $ cd
   $ tar -cvzf old-project-x.tar.gz old  # アーカイブを作成。bz2 にするに
は -z の代わりに -j とする
   $ cvs release -d old            # ローカルのソースを削除 (任意)
```

覚えておくべき役立つオプションを次に挙げます。(cvs の第一引数として使います)

`-n`	効果を及ぼさずに起動
`-t`	cvs の活動の各ステップごとに示されるメッセージを表示

CVS からのエクスポート

CVS から最新版を取得するには、“tomorrow” を使います。

```
   $ cvs ex -D tomorrow module_name
```

CVS 管理者

(ローカルサーバの) プロジェクトへのエイリアスを追加します。

```
   $ su - admin            # staff のメンバ
   $ export CVSROOT=/var/lib/cvs
   $ cvs co CVSROOT/modules
   $ cd CVSROOT
```

```
$ echo "px -a project-x" >>modules
$ cvs ci -m "Now px is an alias for project-x"
$ cvs release -d .
$ exit                  # 又は su から戻るために Ctrl-D を実行
$ cvs co -d project px
 ... CVS から project ディレクトリに project-x
(エイリアス:px) をチェックアウトする
$ cd project
 ... 内容を変更する
```

12.1.3 CVS のトラブルシュート

リポジトリのファイルアクセス権限

CVS はカレントのリポジトリファイルを上書きしませんが、他のファイルで置き換えることはあります。それゆえ、リポジトリディレクトリに書き込み権限を設定することは致命的です。全ての新しいリポジトリの作成には、必要ならばこの条件を保証するために次の手順を行ってください。

```
# cd /var/lib/cvs
# chown -R root:src repository
# chmod -R ug+rwX   repository
# chmod    2775     repository  # 必要ならば、これをサブディレクトリに対
しても行う
```

実行権ビット

ファイルの実行権ビットはチェックアウト時にも保持されます。チェックアウトしたファイルの実行権が設定されているのを見掛けたときはいつでも次のコマンドで CVS リポジトリにあるファイル権限を変更してください。

```
# chmod ugo-x filename
```

12.1.4 CVS コマンド

ここに CVS のコマンドとそのショートカットを示します。

```
{add|ad|new} [-k kflag] [-m 'message'] files...
{admin|adm|rcs} [rcs-options] files...
{annotate|ann} [options] [files...]
{checkout|co|get} [options] modules...
{commit|ci|com}  [-lnR]  [-m  'log_message'  |  -f  file] \
```

```
         [-r revision] [files...]
{diff|di|dif} [-kl] [rcsdiff_options] [[-r rev1 | -D date1] \
         [-r rev2 |  -D date2]] [files...]
{export|ex|exp} [-flNn] -r rev|-D date [-d dir] [-k kflag] module...
{history|hi|his} [-report] [-flags] [-options args] [files...]
{import|im|imp} [-options] repository vendortag releasetag...
{login|logon|lgn}
{log|lo|rlog} [-l] rlog-options [files...]
{rdiff|patch|pa} [-flags] [-V vn] [-r t|-D d [-r t2|-D d2]] modules...
{release|re|rel} [-d] directories...
{remove|rm|delete} [-lR] [files...]
{rtag|rt|rfreeze} [-falnR]  [-b]  [-d]  [-r  tag  |  -D  date] \
          symbolic_tag modules...
{status|st|stat} [-lR] [-v] [files...]
{tag|ta|freeze} [-lR] [-F] [-b] [-d] [-r tag | -D date]  [-f] \
          symbolic_tag [files...]
{update|up|upd} [-AdflPpR] [-d] [-r tag|-D date] files...
```

12.2 Subversion

Subversion は CVS の置換を意図した次世代のバージョン管理システムです。Subversion の開発者は現在 “α 版” と考えていますが、ほとんどの人が使うには十分安定しています。本文書の執筆時では、Subversion は Debian `unstable` でのみ得られます。

12.2.1 Subversion サーバのインストール

`subversion` メタパッケージはサーバを設定するために必要なパッケージ (`libapache2-dav-svn` と `subversion-tools`) に依存します。

リポジトリの設定

現在、`subversion` パッケージはリポジトリを設定しませんので、手動で設定する必要があります。リポジトリの場所としては `/var/local/repos` が考えられます。

ディレクトリを作成します。

```
# mkdir -p /var/local/repos
```

リポジトリデータベースを作成します。

```
# svnadmin create /var/local/repos
```

リポジトリに WWW サーバが書き込み可能にします。

```
# chown -R www-data:www-data /var/local/repos
```

Apache2 の設定

ユーザ認証を通じてリポジトリへのアクセスを許可するには、`/etc/apache2/mods-available/dav_svn.conf` に次の行を追加 (又はコメントアウト) しましょう。

```
<Location /repos>
  DAV svn
  SVNPath /var/local/repos
  AuthType Basic
  AuthName "Subversion repository"
  AuthUserFile /etc/subversion/passwd
  <LimitExcept GET PROPFIND OPTIONS REPORT>
    Require valid-user
  </LimitExcept>
</Location>
```

そして、次のコマンドでユーザ認証ファイルを作成します。

```
htpasswd2 -c /etc/subversion/passwd some-username
```

Apache2 を再起動すると、新しい Subversion リポジトリは URL http://*hostname*/repos によりアクセス可能になります。

12.2.2 CVS リポジトリから Subversion への移行

12.2.3 Subversion の使用例

次の章は Subversion での異なるコマンドの使いかたについて教えます。

新規 Subversion アーカイブの作成

新規 Subversion アーカイブを作成するには、次をタイプします。

```
$ cd ~/your-project          # ソースディレクトリに移動
$ svn import http://localhost/repos your-project \
  project-name -m "initial project import"
```

これはプロジェクトファイルを含む Subversion リポジトリに *project-name* という名前のディレクトリを作成します。アーカイブが存在するかどうかを調べるには http://localhost/repos/ を見てください。

Subversion を用いた作業

Subversion を用いて *project-y* の作業を行います。

```
$ cd                            # 作業エリアに移動
$ svn co http://localhost/repos/project-y  # ソースをチェックアウト
$ cd project-y
 ... 作業を行う ...
$ svn diff                      # diff -u repository/ local/ と同様の出力
$ svn revert modified_file      # ファイルへの変更を undo する
$ svn ci -m "Describe changes"  # 変更をリポジトリにチェックインする
$ vi newfile_added
$ svn add newfile_added
$ svn add new_dir               # new_dir に全ファイルを再帰的に追加
$ svn add -N new_dir2           # 非再帰的にディレクトリを追加
$ svn ci -m "Added newfile_added, new_dir, new_dir2"
$ svn up                        # リポジトリから最新版をマージ
$ svn log                       # コミットした全変更を表示
$ svn copy http://localhost/repos/project-y \
      http://localhost/repos/project-y-branch \
      -m "creating my branch of project-y"  # project-y ブランチを作成
$ svn copy http://localhost/repos/project-y \
      http://localhost/repos/proj-y_release1.0 \
      -m "project-y 1.0 release"    # リリースタグを作成
 ... ブランチ作成とタグづけは同じであることに注意。唯一の違いは
 ... ブランチはコミットされるがタグはそうではないことです。

 ... ブランチへ変更を施す

# ブランチに分けたコピーをメインのコピーにバックマージする
$ svn merge http://localhost/repos/project-y \
   http://localhost/repos/project-y-branch
$ svn co -r 4 http://localhost/repos/project-y # バージョン 4 を取得
```

章 13

プログラミング

実行用テストファイルの名前として “test” を使うのは止めましょう。`test` というコマンドがシェル組込みコマンドにあります。

13.1 始め方

参考文献をいくつか挙げます。

- `/usr/share/doc/`*`package`* にある文書とサンプル。
- Unix / Programming Information (`http://arioch.unomaha.edu/~jclark/#info`)
- *Linux Programming Bible* (John Goerzen/IDG books)

多量の info 文書が書籍として GNU (`http://www.gnu.org/`) から得られます。

次の 4 サブセクションには、`newusers` プログラムのような バッチプロセッサを用いて `/etc/passwd` にアカウント情報を 追加するための数種類の言語によるサンプルスクリプトが含まれます。スクリプトは `first_name last_name password` の形式の数行の入力ファイルを必要と します。(実際ユーザのホームディレクトリはこれらのスクリプトによって 作成されません。)

13.2 シェル

Unix ライクなシステムの動きを理解する 最良の方法は、シェルスクリプトを読むことです。ここで、シェルプログラムに関する いくつかのポイントと注意を与えます。間違いから学ぶには、Shell Mistakes (`http://www.greenend.org.uk/rjk/2001/04/shell.html`) を読んでください。

13.2.1 Bash – GNU 標準の対話シェル

Bash の参考文献をいくつか挙げます。

- 学び始めの際の情報としては、これらが良いでしょう
- `bash(1)`
- `info bash`
- LDP BASH Programming - Introduction HOWTO (`http://www.tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html`)
- `mc /usr/share/doc/bash/examples/ /usr/share/doc/bash/` (サンプルファイルを見るには、`bash-doc` をインストール してください。)
- *Learning the bash Shell*, 2nd edition (O'Reilly)

次に短いプログラム例を示します。(標準入力から `newusers` 用の アカウントエントリを作成)

```
#!/bin/bash
# (C) Osamu Aoki Sun Aug 26 16:53:55 UTC 2001 Public Domain
pid=1000;
while read n1 n2 n3 ; do
if [ ${n1:0:1} != "#" ]; then
let pid=$pid+1
echo ${n1}_${n2}:password:${pid}:${pid}:,,,/home/${n1}_${n2}:/bin/bash
fi
done
```

13.2.2 POSIX シェル

Debian では、数種類のパッケージが POSIX シェルを供給します。

- `dash` (Sarge)
  - Priority: optional
  - Installed-Size: 176
  - 最小で最軽量。最初のブート時に最適。
- `ash` (Woody)
  - Priority: optional
  - Installed-Size: 180
  - 小さく軽量。最初のブート時に適している。
- `bash`
  - Essential: yes
  - Priority: required
  - Installed-Size: 580
  - 大きく多機能。多くの拡張が実装されている。
- `pdksh`
  - Priority: optional
  - Installed-Size: 408
  - 完全な AT&T ksh 互換

移植性を考慮してシェルスクリプトを作成している場合、POSIX シェルスクリプト としてスクリプトを書くのが最良です。シェルスクリプトの POSIX 互換性を テストするには、`ash`

(又は `dash`) にリンクされた `/bin/sh` を使用してください。“bashism” 又は “zshism” を用いたスクリプトを作るのは避けましょう。例えば、次のような表現は避けましょう。

- `if [ foo == bar ] ; then ...`
- `diff -u file.c{.orig,}`
- `mkdir /foo{bar,baz}`

本文書におけるシェルの記述方法は POSIX タイプのシェルのみを用いており、それゆえ、`tcsh` を含む `csh` タイプのシェルは使用していません。

13.2.3 シェルパラメータ

いくつかの**特別なパラメータ**を覚える必要があります。

`$0` = シェル又はシェルスクリプトの名前
`$1` = シェルの第 1 引数
...
`$9` = シェルの第 9 引数
`$#` = ポジショナルパラメータの数
`"$*"` = `"$1 $2 $3 $4 ... $n"`
`"$@"` = `"$1" "$2" "$3" "$4" ... "$n"`
`$?` = 最後のコマンドの終了ステータス
`$$` = シェルスクリプトの PID
`$!` = 最後に起動されたバックグラウンドジョブの PID

基本的な**パラメータ拡張**を覚える必要があります:

形式	*var* がセット	*var* がセットされていない
`${var:-string}`	`$var`	*string*
`${var:+string}`	*string*	`null`
`${var:=string}`	`$var`	*string* (そして *var=string* を起動)
`${var:?string}`	`$var`	(`echo` *string* して終了する)

ここで、これらの演算子全てにおいて、コロン ‘:’ は実際は無くても構いません。

- ‘:’ があると演算子は “exist” と “not null” をテストします。
- ‘:’ が無いと演算子は “exist” のみテストします。

覚えておくべき基本的な**パラメータ置換**には次があります。

形式	結果
`${var%suffix}`	最小の *suffix* パターンを削除
`${var%%suffix}`	最大の *suffix* パターンを削除
`${var#prefix}`	最小の *prefix* パターンを削除
`${var##prefix}`	最大の *prefix* パターンを削除

13.2.4 シェルリダイレクト

覚えておくべき基本的な**リダイレクト**を次に挙げます。(ここで [*n*] はファイルディスクリプタを指定するための任意の番号)

`[`*n*`]>` *`file`*	`stdout` (又は*n*) を *`file`* にリダイレクト。
`[`*n*`]>>` *`file`*	`stdout` (又は*n*) を *`file`* に追加
`[`*n*`]<` *`file`*	*`file`* から `stdin` (又は*n*) にリダイレクト
`[`*n1*`]>&`*n2*	`stdout` (又は *n1*) を *n2* にリダイレクト
`2>` *`file`* `>&2`	`stdout` と `stderr` を *`file`* にリダイレクト
`>` *`file`* `2>&1`	`stdout` と `stderr` を *`file`* にリダイレクト
`|` *`command`*	`stdout` を *`command`* にパイプする
`2>&1 |` *`command`*	`stderr` と `stdout` を *`command`* にパイプする

ここで、
- stdin: 標準入力 (ファイルディスクリプタ = 0)
- stdout: 標準出力 (ファイルディスクリプタ = 1)
- stderr: 標準エラー出力 (ファイルディスクリプタ = 2)

です。

シェルは任意のファイルディスクリプタにより `exec` 組込みコマンドを用いてファイルを開くことができます。

```
$ echo Hello >foo
$ exec 3<foo 4>bar  # ファイルを開く
$ cat <&3 >&4       # 標準入力を 3, 標準出力を 4 にリダイレクト
$ exec 3<&- 4>&-    # ファイルを閉じる
$ cat bar
Hello
```

ここで *n*`<&-` と *n*`>&-` はファイルディスクリプタ *n* を閉じることを意味します。

13.2.5 シェルの条件文

コマンドは条件文の表現に使われる**終了ステータス**を返します。
- 成功: 0 (真)
- エラー: 1–255 (偽)

ここでは 0 の値を “真” の意味で使っていますが、これは他の計算分野で使われている慣習とは異なることに注意してください。又、‘[’ は `test` コマンドと等価であり、条件文表現として ‘]’ まで引数を評価します。

覚えておくべき基本的な**条件イディオム**を次に挙げます。

```
command && if_success_run_this_command_too || true
command || if_not_success_run_this_command_instead

if [ conditional_expression ]; then
 if_success_run_this_command
else
 if_not_success_run_this_command
fi
```

ここで `||` は、シェルが `-e` フラグを呼び出した時にこのシェルスクリプトが誤ってこの行で終了しないことを保証するために必要でした。

条件文表現内での **ファイル** 比較演算を次に挙げます。

`-e` *file*	*file* が存在する場合に真
`-d` *file*	*file* というディレクトリが存在する場合に真
`-f` *file*	*file* という普通のファイルが存在する場合に真
`-w` *file*	*file* が存在し、書き込み可である場合に真
`-x` *file*	*file* が存在し実行可能出ある場合に真
file1 `-nt` *file2*	*file1* が *file2*より (更新時間が)新しい場合に真
file1 `-ot` *file2*	*file1* が *file2* よりも (更新時間が) 古い場合に真
file1 `-ef` *file2*	両者が同じデバイスであり、同じ `inode` 番号を持つ場合に真

条件文表現内での **文字列** 比較演算を次に挙げます。

`-z` *str*	*str* の長さが 0 の場合に真
`-n` *str*	*str* の長さが 0 ではない場合に真
str1 `==` *str2*	二つの文字列が等しい場合に真
str1 `=` *str2*	二つの文字列が等しい場合に真 ("=" は厳密な POSIX 互換なシェルの場合、 "==" の代わりに使用するべき)
str1 `!=` *str2*	二つの文字列が等しくない場合に真
str1 `<` *str2*	*str1* が50音順で *str2* の前に来る場合に真 (ロケール依存)
str1 `>` *str2*	*str1* が50音順で *str2* の後に来る場合に真 (ロケール依存)

条件文表現内での **算術** 整数比較演算には、`-eq`, `-ne`, `-lt`, `-le`, `-gt` そして `-ge` があります。

13.2.6 コマンドライン処理

シェルは次のようにスクリプトを処理します。

- スペース,タブ,改行, ;, (,), <, >, |, & などのメタ文字により **トークン** に分割する。
- “...” や ’...’ 内では無い場合、**keyword** をチェックする。(ループ)
- “...” や ’...’ 内では無い場合、**alias** を展開する。(ループ)
- “...” や ’...’ 内では無い場合、**brace** を展開する (例: `a{1,2}` -> `a1 a2`)
- “...” や ’...’ 内では無い場合、**チルダ** を展開する (例: *~user* -> *user* のホームディレクトリ)
- ’...’ 内で無い場合、**parameter** を $*PARAMETER* に展開する。
- ’...’ 内で無い場合、**コマンド置換** を $(*command*)に展開する。
- “...” や ’...’ 内では無い場合、$IFS により **単語** に分割する。
- “...” や ’...’ 内では無い場合、**pathname** 内の *?[] を展開する。
- **command** を捜す。
  - 関数
  - 組込み
  - $PATH 内のファイル
- ループ

“” 内の ” には何も効果はありません。

シェル内で `set -x` を実行するか、`-x` オプション付きでシェルを起動すると、シェルにコマンドの実行結果を全て表示させます。これはデバッグにはとても便利です。

13.3 Awk

Awk の参考文献をいくつか挙げます。

- *Effective awk Programming*, 3rd edition (O'Reilly)
- *Sed & awk*, 2nd edition (O'Reilly)
- `mawk(1)` and `gawk(1)`
- `info gawk`

(`newusers` のコマンドエントリを作成する) 短いプログラム例 を示します。

```
#!/usr/bin/awk -f
# Script to create a file suitable for use in the 'newusers' command,
# from a file consisting of user IDs and passwords in the form:
# first_name last_name password
# Copyright (c) KMSelf Sat Aug 25 20:47:38 PDT 2001
# Distributed under GNU GPL v 2, or at your option, any later version.
# This program is distributed WITHOUT ANY WARRANTY.

BEGIN {
    # Assign starting UID, GID
    if ( ARGC > 2 ) {
        startuid = ARGV[1]
        delete ARGV[1]
    }
    else {
        printf( "Usage:  newusers startUID file\n" \
          "  where:\n" \
          "    startUID is the starting userid to add, and\n" \
          "    file is an input file in form:\n" \
          "      first_name last_name password\n" \
        )
        exit
    }

    infile = ARGV[1]
    printf( "Starting UID: %s\n\n", startuid )
}

/^#/ { next }
```

```
{
    ++record
    first = $1
    last = $2
    passwd = $3
    user= substr( tolower( first ), 1, 1 ) tolower( last )
    uid = startuid + record - 1
    gid = uid
    printf( "%s:%s:%d:%d:%s %s,,/home/%s:/bin/bash\n",  \
        user, passwd, uid, gid, first, last, user \
        )
}
```

Debian では、POSIX 互換の `awk` を供給するパッケージ が二種類あります。

- `mawk`
    - Priority: required
    - Installed-Size: 228
    - 小さく高速 – 標準のインストールには良い
    - コンパイル時の制限が存在する
        - NF = 32767
        - sprintf buffer = 1020
- `gawk`
    - Priority: optional
    - Installed-Size: 1708
    - 大きく機能が豊富 – 多くの拡張を実装
        - UNIX System V Release 4 版
        - Bell 研究所版 awk
        - GNU 独自の拡張

13.4 Perl

これは Unix ライクシステムの **インタプリタ** です。

Perl の参考文献をいくつか挙げます。

- `perl(1)`
- *Programming Perl*, 3rd edition (O'Reilly)

(`newusers` のコマンドエントリを作成する) 短いプログラム例 を示します。

```
#!/usr/bin/perl
# (C) Osamu Aoki Sun Aug 26 16:53:55 UTC 2001 Public Domain
$pid=1000;
while (<STDIN>) {
        if (/^#/) { next;}
        chop;
```

```
        $pid++;
        ($n1, $n2, $n3) = split / /;
        print $n1,"_",$n2,":", $n3, ":",$pid,
                  ":",$pid,",,,/home/",$n1,"_",$n2,":/bin/bash\n"
}
```

Perl モジュール *module_name* は次のようにインストールします。

```
# perl -MCPAN -e 'install module_name'
```

13.5 Python

すばらしいオブジェクト指向なインタプリタです。

Python の参考文献をいくつか挙げます。

- `python(1)`
- *Learning Python* (O'Reilly).
- Python Programming Language (`http://www.python.org/`)

(`newusers` のコマンドエントリを作成する) 短いプログラム例 を示します。

```
#! /usr/bin/env python
import sys, string

# (C) Osamu Aoki Sun Aug 26 16:53:55 UTC 2001 Public Domain
# Ported from awk script by KMSelf Sat Aug 25 20:47:38 PDT 2001
# This program is distributed WITHOUT ANY WARRANTY.

def usages():
    print \
"Usage:  ", sys.argv[0], " start_UID [filename]\n" \
"\tstartUID is the starting userid to add.\n" \
"\tfilename is input filename. If not specified, standard input.\n\n" \
"Input file format:\n"\
"\tfirst_name last_name password\n"
                return 1

def parsefile(startuid):
    #
    # main filtering
    #
    uid = startuid
    while 1:
        line = infile.readline()
```

```
        if not line:
            break
        if line[0] == '#':
            continue
        (first, last, passwd) = string.split(string.lower(line))
        # above crashes with wrong # of parameters :-)
        user = first[0] + last
        gid = uid
        lineout = "%s:%s:%d:%d:%s %s,,/home/%s:/bin/bash\n" %  \
            (user, passwd, uid, gid, first, last, user)
        sys.stdout.write(lineout)
        +uid

if __name__ == '__main__':
    if len(sys.argv) == 1:
        usages()
    else:
        uid = int(sys.argv[1])
        #print "# UID start from: %d\n" % uid
        if len(sys.argv) > 1:
            infilename   = string.join(sys.argv[2:])
            infile = open(infilename, 'r')
            #print "# Read file from: %s\n\n" % infilename
        else:
            infile = sys.stdin
        parsefile(uid)
```

13.6 Make

Make の参考文献をいくつか挙げます。

- `info make`
- `make(1)`
- *Managing Projects with make,* 2nd edition (O'Reilly)

シンプルな自動変数。

ルールシンタックスを示します。

```
target: [ prerequisites ... ]
 [TAB]  command1
 [TAB]  -command2 # エラーを無視
 [TAB]  @command3 # suppress echoing
```

ここで `[TAB]` はタブの文字コードです。各行は `make` 変数置換を行った後にシェルにより解釈されます。2行以上にわたってスクリプトを続けるには、行の末尾に `\` を書いてください。シェルスクリプトでの環境変数のため `$` を入力するには、`$$` を使用してください。

ターゲットと前提のための**暗黙のルール**は、例えば次のように書けます。

```
%: %.c header.h
```

又は

```
%.o: %.c header.h
```

ここで、ターゲットには `%` (正確のはこれらのうちの一つ) が含まれます。`%` は実際のターゲットのファイル名にある空でないあらゆる文字列にマッチします。前提は同様に実際のターゲット名に関連する名前がどのようなものかを示すために `%` を使います。

サフィックスルールは `make` のために厳密なルールを定義するためには**時代遅れ**です。互換性のために、GNU `make` ではまだサポートされていますが、可能な限り次のような等価なパターンルールを使用しましょう。

```
古いサフィックスルール    --> 新しいパターンルール
.c:             --> %  : %.c
.c.o:           --> %.o: %.c
```

ルールのための自動変数。

```
foo.o: new1.c new2.c old1.c new3.c
$@ == foo.o                         (ターゲット)
$< == new1.c                        (最初の依存関係)
$? == new1.c new2.c new3.c          (新しい依存関係)
$^ == new1.c new2.c old1.c new3.c   (全ての依存関係)
$* == ‘%’ はターゲットパターンにマッチした語幹です。
```

変数のリファレンス。

```
foo1 := bar    # 一回だけの展開
foo2  = bar    # 再帰展開
foo3 += bar    # 追加
SRCS := $(wildcard *.c)
OBJS := $(foo:c=o)
OBJS := $(foo:%.c=%.o)
OBJS := $(patsubst %.c,%.o,$(foo))
DIRS  = $(dir directory/filename.ext) # "directory" を展開
 $(notdir NAMES...), $(basename NAMES...), $(suffix NAMES...) ...
```

自動の内部ルールを見るには、`make -p -f/dev/null` を起動してください。

13.7 C

次をインストールして C を使う準備をしましょう。

```
# apt-get install glibc-doc manpages-dev libc6-dev gcc
```

C の参考文献をいくつか挙げます。

- `info libc` (C ライブラリ関数のリファレンス)
- `gcc(1)`
- `each_C_library_function_name(3)`
- Kernighan & Ritchie, *The C Programming Language*, 2nd edition (Prentice Hall). `http://cm.bell-labs.com/cm/cs/cbook/`
- ISO C standard. `http://www.open-std.org/jtc1/sc22/wg14/`

13.7.1 シンプルな C プログラム (gcc)

`example.c` を `libm` ライブラリを用いて コンパイルして `run_example` 実行ファイルを作成するための シンプルな例です。

```
$ cat > example.c << EOF
#include <stdio.h>
#include <math.h>
#include <string.h>

int main(int argc, char **argv, char **envp){
        double x;
        char y[11];
        x=sqrt(argc+7.5);
        strncpy(y, argv[0], 10); /* prevent buffer overflow */
        y[10] = '\0'; /* fill to make sure string ends with '\0' */
        printf("%5i, %5.3f, %10s, %10s\n", argc, x, y, argv[1]);
        return 0;
}
EOF
$ gcc -Wall -g -o run_example example.c -lm
$ ./run_example
        1, 2.915, ./run_exam,     (null)
$ ./run_example 1234567890qwerty
        2, 3.082, ./run_exam, 1234567890qwerty
```

ここで、`sqrt()` を使うためにライブラリ `lib`**`m`** をリンクするには `-l`**`m`** が必要です。実際のライブラリは `/lib/` に `libm.so.6` という名前で存在し、 これは `libm-2.1.3.so` のシンボリックリンクです。

出力テキストの最後のパラメータを見てください。`%10s` が指定されているにも係わらず10以上の文字があります。

境界チェックを行わずに `sprintf` や `strcpy` のようなポインタを用いるメモリ操作関数を使うのは上記のオーバーラン効果を利用するバッファオーバーフロー攻撃を防ぐために時代遅れになっています。その代わりに、`snprintf` や `strncpy` をご利用下さい。

13.7.2 デバッグ

gdb を用いたデバッグ

準備

```
# apt-get install gdb
```

`gdb` の参考文献をいくつか挙げます。
- `info gdb` (チュートリアル)
- `gdb(1)`
- `http://www.unknownroad.com/rtfm/gdbtut/gdbtoc.html`

`-g` オプション付きでコンパイルされたプログラムをデバッグするには `gdb` を使用します。多くのコマンドには省略形が存在します。タブ拡張もシェルと同様に扱えます。

```
$ gdb program
(gdb) b 1                  # ブレークポイントを一行目に設定
(gdb) run arg1 arg2 arg3 # プログラムを起動
(gdb) next                 # 次の行
...
(gdb) step                 # 前に進む
...
(gdb) p parm               # parm の内容を表示
...
(gdb) p parm=12            # 値を 12 に設定
```

Emacs からデバッグするには、'エディタのコマンドのサマリ (Emacs, Vim)' on page 209 をごらんください。

インストールされる全てのバイナリは Debian システムでは標準で stripped されるべきですので、ほとんどのデバッグシンボルは削除されています。Debian パッケージのデバッグに役立つように `gdb` を利用するには、次の方法で該当するパッケージを再構築する必要があります。
- パッケージの version (`http://www.debian.org/doc/debian-policy/ch-controlfields#s-f-Version`) を上げるために `debian/control` を編集。
- バイナリのコンパイルのために `CFLAGS=-g -Wall` を使うようにビルドスクリプトをチェック。

- Debian パッケージの構築のために `DEB_BUILD_OPTIONS=nostrip,noopt`をエクスポートする。

詳細は Policy 10.1 (`http://www.debian.org/doc/debian-policy/ch-files#s10.1`) をごらんください。

ライブラリの依存性をチェック

プログラムがどのライブラリに依存しているかを調べるには、`ldd` を使用します。

```
$ ldd /bin/ls
        librt.so.1 => /lib/librt.so.1 (0x4001e000)
        libc.so.6 => /lib/libc.so.6 (0x40030000)
        libpthread.so.0 => /lib/libpthread.so.0 (0x40153000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

`ls` を `chroot` された環境で動かすには、上記に示したライブラリを `chroot` された環境に置く必要があります。

次に挙げるコマンドも役立ちます。

- `strace`: システムコールとシグナルを追跡
- `ltrace`: ライブラリコールを追跡

メモリリーク検知ツールを用いたデバッグ

数種類のメモリリーク検知ツールを Debian で使用できます。

- `njamd`
- `valgrind`
- `dmalloc`
- `electric-fence`
- `memprof`
- `memwatch` (パッケージ化されていないが、memwatch (`http://directory.fsf.org/devel/debug/memwatch.html`) から取得可能です。)
- `mpatrol`
- `leaktracer`
- `libgc6`
- Parasoft (`http://www.parasoft.com`) による `Insure++` (non-free, 商用では有償)

又 Debugging Tools for Dynamic Storage Allocation and Memory Management (`http://www.cs.colorado.edu/homes/zorn/public_html/MallocDebug.html`). も調べてみてください。

13.7.3 Flex – Lex の改良版

`flex` は高速な字句解析器生成器です。

`flex` の参考文献をいくつか挙げます。

- `info flex`(チュートリアル)
- `flex(1)`

ライブラリ無しでコンパイルできるように、次のように見える`main()`と`yywrap()`又は`program.l`を供給する必要があります。(`yywrap`はマクロです。`%option main`は暗黙に`%option noyywrap`になります)

```
%option main
%%
.|\n    ECHO ;
%%
```

その代わりに、`cc`コマンドラインの最後に(AT&T-Lexの`-ll`のように)`-lfl`リンカオプションを付けてコンパイルできます。この場合、`%option`は必要ありません。

13.7.4 Bison – Yacc の改良版

Debian では Yacc 互換な LALR パーサジェネレータパッケージがいくつか供給されています。

- `bison`: GNU LALR パーサジェネレータ
- `byacc`: バークレー LALR パーサジェネレータ
- `btyacc`: `byacc`に基づくバックトラッキングパーサジェネレータ

`bison`の参考文献をいくつか挙げます。

- `info bison`(チュートリアル)
- `bison(1)`

`main()`と`yyerror()`を供給する必要があります。`main()`は`yyparse()`を呼び出し、これは`yyparse()`を呼び出します。さらにこれは通常 FleX により生成される`yylex()`を呼び出します。

```
%%

%%
```

13.7.5 Autoconf

`autoconf`は GNU ビルドシステム全体を使用する多種の UNIX ライクなシステムに適合するようにソフトウェアのソースコードパッケージを自動で設定するシェルスクリプトを生成するためのツールです。

`autoconf`は`configure`と呼ばれる設定スクリプトを生成します。`configure`は`Makefile.in`テンプレートを用いて自動的にカスタマイズされた`Makefile`を生成します。

プログラムのコンパイルとインストール

Debian は`/usr/local/` 以下にあるファイルに触りません。 (‘多様性のサポート’ on page 23 参照) ですから、ソースからプログラムをコンパイルして `/usr/local/` 下に置いても Debian とは干渉しません。

```
$ cd src
$ ./configure --prefix=/usr/local
$ make
$ make install # システム上にファイルをインストールする
```

プログラムのアンインストール

プログラムを削除したいが、ソースをまだ持っており、`autoconf` や `automake` を使っていてコンパイル時の設定を覚えていたならば、次の手順でアンインストールができます。

```
$ ./configure all-of-the-options-you-gave-it
# make uninstall
```

代わりに、インストール手順で `/usr/local/` の下にしかファイルを置いておらず、重要なものは何もないことを絶対確信しているなら、全ての内容を次のように削除できます。

```
# find /usr/local -type f -print0 | xargs -0 rm -f
```

どこにファイルをインストールしか確信できない場合、アンインストールのためのクリーンな道筋を供給してくれる `checkinstall` の使用を考えるべきです。

13.8 Web

基本的なインタラクティブで動的な web ページは以下の手順で作られます。

- HTMLフォームを用いてブラウザユーザに質問が提示されます。。
- フォームに書き込みクリックすることで、エンコードされたパラメータを組み込んだ URL をブラウザから web サーバに送信します。。[1] 例えば、
  - `http://www.foo.dom/cgi-bin/program.pl?VAR1=VAL1&VAR2=VAL2&VAR3=VAL3`
  - `http://www.foo.dom/cgi-bin/program.py?VAR1=VAL1&VAR2=VAL2&VAR3=VAL3`
  - `http://www.foo.dom/program.php?VAR1=VAL1&VAR2=VAL2&VAR3=VAL3`
- web サーバ上の CGI プログラム (`program.*` のいずれか) はデコードされたパラメータ “`VAR1=VAL1 VAR2=VAL2 VAR3=VAL3`” を環境変数 “`QUERY_STRING`” の値として受け取り、実行します。
- のCGI プログラムの `stdout` は web ブラウザに送られ、インタラクティブで動的な web ページとして提示されます。

---

[1] ここで 16進数で `nn` に対応する文字をエンコードするのには、`%nn` を用います。

安全上、勝手に作ったいい加減なプログラムでCGIパラメータを読むのは賢明でないとされています。Perl (‘Perl’ on page 227参照) やPython (‘Python’ on page 228参照) の確立したモジュールが有ります。 PHP (`http://www.php.net/`) はこのような機能を自身で持っています。 クライアントサイドでデータ保持が必要な場合はクッキーが用いられます。 クライアントサイドでデータ処理が必要な場合はjavascriptがよく用いられます。

詳細はThe Common Gateway Interface (`http://hoohoo.ncsa.uiuc.edu/cgi/`) とThe Apache Software Foundation (`http://www.apache.org/`) とJavaScript (`http://www.mozilla.org/js/`) を参照下さい。

Googleで“CGI tutorial”を探すのに`http://www.google.com/search?hl=en&ie=UTF-8&q=CGI+tutori` というエンコードされたURLを直接ブラウザのアドレスに打ち込むことは、 CGIスクリプトのGoogleサーバ上での動きを知るための良い方法です。

13.9 ドキュメントの準備

13.9.1 `roff` 植字

伝統的に、`roff`は主要なUnixテキスト処理システムです。

`roff(7)`, `groff(7)`, `groff(1)`, `grotty(1)`, `troff(1)`, `groff_mdoc(7)`, `groff_man(7)`, `groff_ms(7)`, `groff_me(7)`, `groff_mm(7)` そして `info groff` をご覧下さい。

`-me`マクロに良質のチュートリアルが存在します。 `groff` (1.18以後) をインストールしている場合、 `/usr/share/doc/groff/meintro.me.gz`を見付けて次を実行してください。

```
$ zcat /usr/share/doc/groff/meintro.me.gz | \
     groff -Tascii -me - | less -R
```

次の手順は完全なプレインテキストファイルを作成します。

```
$ zcat /usr/share/doc/groff/meintro.me.gz | \
    GROFF_NO_SGR=1 groff -Tascii -me - | col -b -x > meintro.txt
```

印刷するには、PostScript出力を使いましょう。

```
$ groff -Tps meintro.txt | lpr
$ groff -Tps meintro.txt | mpage -2 | lpr
```

13.9.2 SGML

次のパッケージをインストールしてSGMLを使う準備を行います。

```
# apt-get install debiandoc-sgml debiandoc-sgml-doc
```

`debiandoc-sgml` の参考文献をいくつか挙げます。

- `/usr/share/doc/debiandoc-sgml-doc`
- `debiandoc-sgml(1)`
- *DocBook: The Definitive Guide* (*`/usr/share/doc/docbook-defguide/html/docbook.html`*), by Walsh and Muellner (O'Reilly) (package `docbook-defguide`)

SGML により複数の文書フォーマットの管理を行えます。簡単な SGML システムの一つ が Debiandoc であり、本文書でも用いられています。Debiandoc は オリジナルのテキストファイルから次の文字について些細な変換が必要です。

- "<" –> `<`
- ">" –> `>`
- " " –> ` ` (半角スペース)
- "&" –> `&`
- "%" –> `%`
- "(c)" –> `©`
- "–" –> `–`
- "—" –> `—`

セクションを印刷不可能なコメントとしてマークするには、次に示すような 内容を挿入します。

```
<!-- State issue here ... -->
```

セクションを変換可能なコメントとしてマークするには、次に示すような内容を 挿入します。

```
<![ %FIXME; [ State issue here ... ]]>
```

SGML では、エンティティの 最初の定義 が優先されます。例えば、

```
<!entity % qref "INCLUDE">
<![ %qref; [ <!entity param "Data 1"> ]]>
<!entity param "Data 2">
&param;
```

では、"Data 1" になります。最初の行が "INCLUDE" ではなく "IGNORE" を 含んでいる場合は、"Data 2" となります。(二行目は条件文) 又、フレーズの繰り返しは内容からあらかじめ分けて定義できます。

```
<!entity whoisthis "my">
Hello &whoisthis; friend.
This is &whoisthis; book.
```

これは次のような結果となります。

```
Hello my friend.
This is my book.
```

examples (`http://www.debian.org/doc/manuals/debian-reference/examples/`) にある短い SGML のサンプルの `sample.sgml` をご覧下さい。

SGML 文書が巨大になった場合、バックエンドテキストプロセッサとして 使われる TeX が時々エラーを起こす場合があります。'TeX/LaTeX' on page 238 をごらんください。

13.9.3 TeX/LaTeX

準備:

```
# tasksel # その他 --> TeX/LaTeX 環境 を選択して
TeX/LaTeX をインストールしましょう。
```

LaTeX の参考文献をいくつか示します。

- The teTeX HOWTO: The Linux-teTeX Local Guide (`http://www.tldp.org/HOWTO/TeTeX-HOWTO.html`)
- `tex(1)`
- `latex(1)`
- *The TeXbook*, by Donald E. Knuth, (Addison-Wesley) [2]
- *LaTeX - A Document Preparation System*, by Leslie Lamport, (Addison-Wesley)
- *The LaTeX Companion*, by Goossens, Mittelbach, Samarin, (Addison-Wesley)

TeX は最強の植字環境です。多くの SGML プロセッサは TeX をバックエンド テキストプロセッサとして使っています。`lyx`, `lyx-xforms` 又は `lyx-qt` により供給される Lyx や、`texmacs` により供給される GNU TeXmacs は見栄えの良い LaTeX 用のWYSIWYG 編集環境を提供しますが、多くの人はソースの編集のためには Emacs や Vim を選択しています。

多くのオンラインのリソースが得られます。

- teTeX - A Documentation Guide (`/usr/share/doc/texmf/newhelpindex.html`) (`tetex-doc` パッケージ)
- A Quick Introduction to LaTeX (`http://www.msu.edu/user/pfaffben/writings/`)
- A Simple Guide to Latex/Lyx (`http://www.stat.rice.edu/~helpdesk/howto/lyxguide.html`)

---

[2] この本に収納されている TeX のソースは `ftp://ftp.dante.de/pub/tex/systems/knuth/tex/texbook.tex` で取得可能です。 `ftp://ftp.dante.de/pub/tex/systems/knuth/lib/manmac.tex` には必要なマクロのほとんどが 含まれています。この文書の 7 から 10 行目をコメントアウトし、`\input manmac \proofmodefalse` を追加した後に 文書を `tex` で処理できます。 オンライン版を使わずにこの本(と Donald E. Knuth による他の全ての本) を購入するのを強くお薦めします。しかしソースは TeX 入力の 素晴らしい例です!。

- Word Processing Using LaTeX (`http://www-h.eng.cam.ac.uk/help/tpl/textprocessing/latex_basic/latex_basic.html`)
- Local User Guide to teTeX/LaTeX (`http://supportweb.cs.bham.ac.uk/documentation/LaTeX/lguide/local-guide/local-guide.html`)

文書が巨大になった場合、TeX はエラーを起こす場合がときどきあります。この問題を修正するには、`/etc/texmf/texmf.cnf` の pool サイズ を増大させる (又は `/etc/texmf/texmf.d/95NonPath` を適切に編集して `update-texmf` を起動する) 必要があります。

13.9.4 Literate Programming

文書を含むコードを書く代わりに、文芸的プログラマはコードを含む文書 を書くものです。この手法はプログラムのための良い文書を保証します。

文芸的プログラムについては、Literate Programming (`http://www.literateprogramming.com/`) をごらんください。

Noweb

準備

```
# apt-get install nowebm
```

Noweb の参考文献をいくつか挙げます。

- Noweb — A Simple, Extensible Tool for Literate Programming (`http://www.eecs.harvard.edu/~nr/noweb/`)
- `noweb(1)`

これは WEB ライクな文芸的プログラムツールであり、よりシンプルですが 拡張性と言語との独立性を供給します。[3] `noweb` が起動されると、プログラムのソースコードを noweb ファイルで言及されている出力ファイルに書き出し、文書の植字のために TeX ファイルを出力します。

Debian の `ifupdown` パッケージが良い例です。

```
$ apt-get source ifupdown
$ cd ifupdown*
$ make ifupdown.pdf ifupdown.ps
```

Doxygen

準備

---

[3] WB は World Wide Web とは **何も** 関連はありません。(PASCAL 用の) WEB と (C/C++ 用の) CWEB は伝統的な文芸的プログラム ツールです。

```
# apt-get install doxygen doxygen-doc doxygen-gui
```

Doxygen (created by `doxygen`!) の参考文献をいくつか挙げます。

- Homepage (`http://www.doxygen.org/`)
- `/usr/share/doc/doxygen-doc/html/index.html`

C++, C, Java, IDL, さらにある程度は PHP と C#も含めて プログラムに関する HTML, RTF, Unix マニュアルページ, PostScript そして PDF (LaTeX を用いて) 文書を生成できます。Doxygen は JavaDoc (1.1), Qt-Doc, KDOC と互換性があり、 Troll Tech の Qt (`http://www.trolltech.no/qt/`) ツールキットを利用する プロジェクトが用いるために特別に設計されていました。 Doxygen はドキュメント化されていないプログラムであっても、依存グラフ、 協同ダイアログ、そしてグラフィカルなクラス階層グラフを含む内容を生成します。 この出力は Qt のドキュメントに似ています。

13.10 パッケージング

準備

```
# apt-get install debian-policy developers-reference \
                  maint-guide dh-make debhelper
# apt-get install packaging-manual # if Potato
```

パッケージのための参考文献をいくつか挙げます。

- 'Debian パッケージ管理システム' on page 11 (基礎)
- Debian New Maintainers' Guide (チュートリアル)
- `dh-make(1)`
- Debian Developer's Reference (最良の練習)
- Debian Policy Manual (信頼されている)
- Packaging Manual (Potato 用)

13.10.1 単独のバイナリをパッケージングする

Joey Hess による単独のバイナリをパッケージングするためのクイックハックな 方法です。

```
# mkdir -p mypkg/usr/bin mypkg/DEBIAN
# cp binary mypkg/usr/bin
# cat > mypkg/DEBIAN/control
Package: mypackage
Version: 1
Architecture: i386
Maintainer: Joey Hess <joeyh@debian.org>
Description: my little package
 Don't expect much.
^D
# dpkg-deb -b mypkg
```

13.10.2 ツールを用いたパッケージング

`dh-make` パッケージにある `dh_make` を使って パッケージのベースラインを作成します。そして、`dh-make(1)` にある指示に従って処理します。 これは `debian/rules` 内で `debhelper` を利用します。

古いアプローチでは `debmake` にある `deb-make` を使います。この方法は `debhelper` を利用せず、シェルのみに 依存します。`debmake` は使わないようにしましょう。

複数のソースパッケージの例としては、“mc” (`dpkg-source -x mc_4.5.54.dsc` をごらんください。これは Adam Heath による “sys-build.mk” を利用しています。又、“glibc” (`dpkg-source -x glibc_2.2.4-1.dsc`) は Joel Klecker (<`espy@debian.org`>) による他のシステムを利用しています。

章 14

GnuPG

参考文献をいくつか挙げます。

- `gpg(1)`
- `/usr/share/doc/gnupg/README.gz`
- `/usr/share/doc/gnupg-doc/GNU_Privacy_Handbook/` にある *GNU privacy handbook* (`gnupg-doc` パッケージをインストールする)

14.1 GnuPG のインストール

```
# gpg --gen-key                        # 新しい鍵を生成
# gpg --gen-revoke my_user_ID          # my_user_ID 用の失効鍵を生成
# host -l pgp.net | grep www|less # pgp キーサーバを知る
```

`$HOME/.gnupg/gpg.conf` (又は旧バージョンでは `$HOME/.gnupg/options` に) 設定すべき良質の標準キーサーバには 次が含まれます。

```
keyserver hkp://subkeys.pgp.net
```

ここで 二つ以上のサブ鍵を生成**しない** ように注意する必要が あります。そうしてしまった場合、pgp.net にあるキーサーバはあなたの鍵を **壊す** 可能性があります。これらの壊れたサブ鍵を扱うには、 新しい gnupg (1.2.1-2 以降) を使ってください。 `http://fortytwo.ch/gpg/subkeys` をごらんください。

14.2 GnuPG を使う

ファイルの操作方法を次に挙げます。

```
$ gpg  [options]  command  [args]
```

```
$ gpg {--armor|-a} {--sign|-s} file # ファイルに署名してテキストファイル file.asc に置く
$ gpg --clearsign file              # メッセージを clearsign で署名する
$ gpg --clearsign --not-dash-escaped patchfile  # パッチファイルを clearsign で署名する
$ gpg --verify file                 # clearsign された file を検証する
$ gpg -o file.sig {-b|--detach-sig} file # 単独の署名を作成
$ gpg --verify file.sig file        # file.sig を用いて file を検証する
$ gpg -o crypt_file {--recipient|-r} name {--encrypt|-e} file
        # 名前を対象とした公開鍵の暗号化
$ gpg -o crypt_file {--symmetric|-c} file # 対称暗号化
$ gpg -o file --decrypt crypt_file  # 復号化
```

14.3 GnuPG の管理

GnuPG の主な管理手順をいくつか挙げます。

```
$ gpg --edit-key user_ID                # 対話的なヘルプ
$ gpg -o file --exports                 # 全ての鍵を file にエクスポート
$ gpg --imports file                    # 全ての鍵を file からインポート
$ gpg --send-keys user_ID               # user_ID の鍵をキーサーバに送信
$ gpg --recv-keys user_ID               # キーサーバから user_ID の鍵を取得
$ gpg --list-keys user_ID               # user_ID の鍵をリスト
$ gpg --list-sigs user_ID               # user_ID の署名をリスト
$ gpg --check-sigs user_ID              # user_ID の署名をチェック
$ gpg --fingerprint user_ID             # user_ID のフィンガープリントをチェック
$ gpg --list-sigs | grep '^sig' | grep '[User id not found]' \
  | awk '{print $2}' | sort -u | xargs gpg --recv-keys # 未知の鍵を取得
  # 全ての未知の署名に対する鍵を更新
$ gpg --refresh-keys                                # ローカルの keyring を更新
```

コードの信頼度の指針を示します。

– ownertrust が割り当てられていない/まだ計算されていない

`e`	信頼度の計算に失敗。
`q`	評価のための情報が足りない。
`n`	この鍵を信頼できない。
`m`	ある程度信頼できる。
`f`	完全に信頼できる。
`u`	究極的に信頼できる。

次の手順により、評判の良いキーサーバである `hkp://subkeys.pgp.net` に自分の鍵 *“A8061F32”* をアップロードできます。

```
$ gpg --keyserver hkp://subkeys.pgp.net --send-keys A8061F32
```

14.4 アプリケーションでGnuPGを使う

14.4.1 MuttでGnuPGを使う

次の内容を `~/.muttrc` に追加すれば、処理の重いGnuPGを自動で 起動せずに、index メニューで‘`S`’をタイプするとGnuPGが呼び出される ようになります。

```
macro index S ":toggle pgp_verify_sig\n"
set pgp_verify_sig=no
```

14.4.2 VimでGnuPGを使う

GnuPGを自動的に起動するために、examples subdirectory (`http://www.debian.org/doc/manuals/debian-reference/examples/`) から得られる `_vimrc` の内容を `~/.vimrc`に追加してください。

章 15

Debian のサポート

次のリソースは Debian を使う上でのヘルプ、助言、そしてサポート を提供します。 メーリングリストで泣きつく前にこれらを使って自助努力してみましょう。:)

WWW ブラウザを使ったり、 `dwww` や `dhelp` パッケージ と同名のコマンドを通じてシステム上のたくさんのドキュメントに アクセスできます。

15.1 参考文献

次の参考文献は Debian と Linux 一般のためのものです。 これらの内容はお互いに矛盾する場合は、ドキュメントなどの二次文書よりも ソースなどの一次文書を常に信頼してください。

- インストールマニュアル (primary)
  - インストールやアップグレードの前に読みましょう。
  - Web: `http://www.debian.org/releases/stable/installmanual`
  - Web: `http://www.debian.org/releases/testing/installmanual` (作業中のため時々なくなります)
  - パッケージ: `Not available in install-doc:  Bug#155374`
  - ファイル: `DebianCDunder/doc/`
- リリースノート (primary)
  - 経験者でもインストールやアップグレード前に読むべし。
  - Web: `http://www.debian.org/releases/stable/releasenotes`
  - Web: `http://www.debian.org/releases/testing/releasenotes` (作業中のため時々なくなります)
  - パッケージ: `Not available in install-doc:  Bug#155374`
  - ファイル: `DebianCDunder/doc/`
- FAQ (secondary)
  - 良く尋ねられる質問。

  - Web: `http://www.debian.org/doc/manuals/debian-faq/`
  - パッケージ: `doc-debian`
  - ファイル: `/usr/share/doc/debian/FAQ/index.html`

- Debian レファレンス (secondary)
  - 最も包括的なインストール後のユーザマニュアル。
  - Web: `http://www.debian.org/doc/manuals/debian-reference/`
  - パッケージ: `debian-reference-ja`
  - ファイル: `/usr/share/doc/Debian/reference/`

- APT HOWTO (secondary)
  - Debian パッケージ管理のための詳細なユーザガイドです。(Woody)
  - Web: `http://www.debian.org/doc/manuals/apt-howto/`
  - パッケージ: `apt-howto`
  - ファイル: `/usr/share/doc/Debian/apt-howto/`

- Debian を強固にするマニュアル (secondary)
  - 標準の Debian のインストール後にセキュリティを強化するための 詳細なユーザマニュアル。 (Woody)
  - Web: `http://www.debian.org/doc/manuals/securing-debian-howto/`
  - パッケージ: `harden-doc`
  - ファイル: `/usr/share/doc/harden-doc/html/securing-debian-howto/`

- `dselect` Documentation for Beginners (secondary)
  - `dselect` のチュートリアル。
  - Web: `http://www.debian.org/releases/woody/i386/dselect-beginner`
  - パッケージ: `Not available in install-doc: Bug#155374`
  - ファイル: `DebianCDunder/doc/`

- Debian Policy Manual (primary)
  - Debian の技術的バックボーン。
  - Web: `http://www.debian.org/doc/debian-policy/`
  - パッケージ: `debian-policy`
  - ファイル: `/usr/share/doc/debian-policy/`

- Debian Developer's Reference (primary)
  - 開発者のための基本的な知識。
  - 私達全員が一度は読むべき。
  - Web: `http://www.debian.org/doc/manuals/developers-reference/`
  - パッケージ: `developers-reference`
  - ファイル: `/usr/share/doc/developers-reference/`

- Debian New Maintainers' Guide (primary)
  - 実践的な開発者用ガイド。
  - 私達全員のためのパッケージチュートリアル。
  - Web: `http://www.debian.org/doc/manuals/maint-guide/`

  - パッケージ: `maint-guide`
  - ファイル: `/usr/share/doc/maint-guide/`

- パッケージングマニュアル (Potato)
  - Potato にて `packaging-manual` パッケージにあります。 (*Developer's Reference* の付録に移動)

- Unix スタイルのマニュアルページ (primary)
  - `dlocate -man` *`package-name`* (有るものを羅列)
  - `man` *`section command-name`*

- GNU スタイルの info ページ (primary)
  - `info` (トップレベルにアクセス)
  - `info` *`command-name`*

- パッケージ独自のドキュメント (primary)
  - `/usr/share/doc/`*`package-name`* の下にあります。

- LDP: Linux Documentation Project (secondary)
  - Linux 一般の HOWTO と mini-HOWTO
  - Web: `http://www.tldp.org/`
  - パッケージ: `doc-linux-text` と `doc-linux-html`
  - ファイル: `/usr/share/doc/HOWTO/`

- Linux Gazette (secondary) – new issues monthly
  - The Linux Gazette
  - Web: `http://www.linuxgazette.com/`
  - パッケージ: `lg-all` or `lg-latest-two`
  - ファイル: `/usr/share/doc/lg/`

- DDP: Debian Documentation Project (secondary)
  - Debian に特有のマニュアル
  - Web: `http://www.debian.org/doc/`

- Debian Developers' Corner (secondary)
  - Debian 開発者のための主要な情報
  - エンドユーザのための洞察に満ちている
  - Web: `http://www.debian.org/devel/`

- ソースコード (absolutely primary)
  - だれも文句をいわないでしょう :-)
  - 'ソースコード' on page 11 に従ってソースをダウンロードしよう。

- Internet Assigned Numbers Authority (primary)
  - Web: `http://www.iana.org/`
  - パッケージ: `doc-iana`
  - ファイル: `/usr/share/doc/doc-iana/`

- Internet requests for comments (IETF standards) (primary)
    - Web: `http://www.ietf.org/rfc.html`
    - パッケージ: `doc-rfc`
    - ファイル: `/usr/share/doc/RFC/`

次の参考文献は Unix 一般のためのものです。異なる Unix システムの間には いくつかの些細な相異点が存在することに注意してください。デバイス名 や初期化方法には特に注意が必要です。

- *The UNIX Programming Environment*
    - Unix の動作原理を学ぶために読む本
    - 著: B. W. Kernighan, R. Pike
    - 出版社: Princeton Hall Software Series

- *The C Programming Language* (second edition)
    - ANSI C について学ぶために読む本
    - 著: B. W. Kernighan, D. M. Ritchie
    - 出版社: Princeton Hall Software Series

- *UNIX Power Tools*
    - Unix の tips を学ぶために読む本
    - 著: Jerry Peek, Tim O'Reilly, and Mike Loukides
    - 出版社: O'Reilly and Associates

- *Essential System Administration* (second edition)
    - 多くの Unix フレーバ用の Unix システム管理について学ぶために読む本
    - 著者: Aeleen Frisch
    - 出版社: O'Reilly and Associates

- *Linux: Rute User*の入門解説書
    - GNU/Linuxシステム管理に関する非常に良い オンラインとハードカバー本
    - 著者: Paul Sheer
    - 出版社: Prentice Hall
    - パッケージ: `rutebook` (from `non-free`)
    - ファイル: `/usr/share/doc/rutebook/`

- Bell 研究所: 計算機科学研究所
    - 豊富な Unix の歴史に関するアーカイブ
    - メイン: `http://cm.bell-labs.com/cm/cs/`
    - 精選された技術レポート: `http://cm.bell-labs.com/cm/cs/cstr.html`
    - 論文: `http://cm.bell-labs.com/cm/cs/papers.html`

- Linux の一般的なサポートのオンラインリソース
    - Debian Planet (`http://www.debianplanet.org/`)
    - debianHELP (`http://www.debianhelp.org/`)
    - Linux.com (`http://linux.com/`)

  - The Linux Home Page at Linux Online (`http://www.linux.org/`)
  - Red Hat (commercial Linux vender) (`http://www.redhat.com/`) (RPM, Sys-V init)
  - SuSE, Inc. (commercial Linux vender) (`http://www.suse.de/`) (RPM, Sys-V init)
  - Slackware (`http://www.slackware.com/`) (TGZ, BSD-style init)

- 一般的な Unix ガイドとオンラインリソース
  - The UNIX System by The Open Group (`http://www.unix.org/`)
  - A UNIX Introductory Course from Ohio State University (`http://www-wks.acs.ohio-state.edu/unix_course/unix.html`)
  - UNIXhelp from The University of Edinburgh (`http://unixhelp.ed.ac.uk/`)
  - Unix / Programming Information (`http://arioch.unomaha.edu/~jclark/#info`)
  - comp.unix.questions FAQ (`http://www.faqs.org/faqs/unix-faq/faq/`)
  - comp.unix.user-friendly FAQ (`http://www.camelcity.com/~noel/usenet/cuuf-FAQ.htm`)
  - FreeBSD Documentation (`http://www.freebsd.org/docs.html`)
  - The FreeBSD Handbook (`http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/index.html`)
  - UNIX GUIDE (`http://ieee.uow.edu.au/documents/`)
  - The Unix Heritage Society (`http://www.tuhs.org/`)

- フリーソフトウェアプロジェクトのホームページ
  - GNU Project (`http://www.gnu.org/`)
  - The Linux Documentation Project (`http://www.tldp.org/`)
  - The Linux Kernel Archives (`http://www.linux.org/`)
  - The XFree86 Project, Inc (`http://www.xfree86.org/`)
  - GNOME (`http://www.gnome.org/`)
  - K Desktop Environment (`http://www.kde.org/`)
  - GNU software at Red Hat (`http://sources.redhat.com/`)
  - Mozilla (`http://www.mozilla.org`)
  - FreeBSD (`http://www.freebsd.org/`)
  - OpenBSD (`http://www.openbsd.org/`)
  - NetBSD (`http://www.netbsd.org/`)

15.2 単語の意味を調べる

Debian では暗号的な jargon や短縮形の単語が良く使われています。 次のコマンドはこれらの問題のほとんどを解決してくれます。 (`dict` パッケージと関連パッケージが必要です)

```
$ dict put-a-weird-word-here
```

15.3 Debian パッケージの人気度を調べる

Debian には多くのパッケージが存在するので、まず試すべきパッケージを 知るのが難しい場合があります。他の人が何を使っているかを知るには Debian Popularity Contest Results (`http://popcon.debian.org/`) をごらんください。又、この統計に貢献するには `popularity-contest` パッケージをインストールしてください。

15.4 Debian バグ追跡システム (BTS)

Debian ディストリビューション はユーザや開発者により 報告されたバグを記録する bug tracking system (BTS) (`http://bugs.debian.org/`) を持ちます。報告された各バグには番号が振られ、処理されたとマークされるまでは ファイルに保存されます。

バグレ ポートを提出するまでに、誰か他の人により提出されているかどうかを チェックする必要があります。現在提出されているバグのリストは World Wide Web (`http://bugs.debian.org/`) や elsewhere (`http://www.debian.org/Bugs/Access`) にあります。'Debian のバグをチェックし、助けを求める' on page 85 もご覧下さい。

FTBFS としてマークされているリリースクリティカルな バグレ ポートが多くあります。これは "ソースからビルドすることに失敗" を示します。

バグレ ポートする方法についての指示は `http://www.debian.org/Bugs/Reporting` に与えられています。

15.5 メーリングリスト

最低でも Debian の状況についていくために "debian-devel-announce" (英語、購読のみでトラフィック数は少ない) を 購読してください。

Debian ユーザが最も興味を抱くメーリングリストは "debian-user" (英語、オープンで高トラフィック) と他の "debian-user-*language*" メーリングリスト (他言語用) です。

これらのメーリングリストの情報と購読方法の詳細については、`http://lists.debian.org/` をごらんください。質問を投稿する前に質問の解答が あるかどうか知るためにアーカイブをチェックしてください。又、標準的な メーリングリストのエチケットを守ってください。

メーリングリストの返答で CCを入れてもらいたい場合には、`Mail-Followup-To:` ヘッダを使うのが大変有効です。 これはメーリングリストの非公式の慣習で `http://cr.yp.to/proto/replyto.html` に説明されています。

15.6 Internet Relay Chat (IRC)

IRC (Internet Relay Chat) は世界中の人々が同時に会話するための方法です。 Debian 向けの IRC チャンネルは freenode (`http://www.freenode.info/`) IRC ネットワークで見かりま

す。 接続するには、IRC クライアントが必要です。有名なクライアントとしては、 XChat, BitchX, ircII, irssi, eipc4, や LSirc があり、これらは Debian パッケージとして存在します。 クライアントをインストールした後に、 サーバに接続するようにクライアントに指示する必要があります。 ほとんどのクライアントでは、次のようにタイプすれば接続できます。

```
/server irc.debian.org
```

接続した後、次のようにタイプして #debian チャンネルに参加します。

```
/join #debian
```

#debian チャンネルから離れるには、次をタイプします。

```
/part #debian
```

irc クライアントを終了するには、次をタイプします。

```
/quit
```

foo に個人的なメッセージ “Hello Mr. foo” を送るには、 次をタイプします。

```
/msg foo Hello Mr. Foo
```

/ を前に置かずにタイプした全てはメッセージとしてチャンネルに 送られることに注意してください。

注意: XChat のようなクライアントはサーバ/チャンネルに参加するための グラフィックユーザインターフェースが異なる場合が良くあります。

15.7 サーチエンジン

Debian に関連するドキュメントを提供するサーチエンジンはたくさんあります。

- Debian WWW search site (`http://search.debian.org/`).
- Google (`http://www.google.com/`): 検索ワードとして “site:debian.org” を含めましょう。
- Google Groups (`http://groups.google.com/`): ニュースグループの ための検索エンジンです。検索ワードとして “group:linux.debian.*” を含めましょう。
- AltaVista (`http://www.altavista.com/`)

例えば、“cgi-perl” のような文字列を検索すると、その制御ファイルに ある簡潔な説明よりも、このパッケージのより詳細な説明が得られます。 関連する助言は ‘Debian のバグをチェックし、助けを求める’ on page 85 をごらんください。

15.8 ウェブサイト

私が特別な話題について集めた URL をランダムにいくつか示します。

- IBM developerWorks: Linux (`http://www.ibm.com/developerworks/linux/`)
- Adrian Bunk's latest packages (back port to stable) (`http://www.fs.tum.de/~bunk/`)
- Linux on Laptops (`http://www.linux-laptop.net/`)
- Xterm FAQ (`http://dickey.his.com/xterm/xterm.faq.html`)
- EXT3 File System mini-HOWTO (`http://www.zip.com.au/~akpm/linux/ext3/ext3-usage.html`)
- Large File Support in Linux (`http://www.suse.de/~aj/linux_lfs.html`)
- Window Managers for X (`http://www.xwinman.org`)
- Linux USB Project (`http://www.linux-usb.org/`)
- SuSE pages for CJK (`http://www.suse.de/~mfabian/suse-cjk/suse-cjk.html`)
- LNX-BBC (Business-card-sized boot CD project) (`http://www.lnx-bbc.org/`)
- Linux info by Karsten Self (partitioning, backup, browsers. . .) (`http://kmself.home.netcom.com/Linux/index.html`)
- Backup info HOWTO by Alvin Oga (`http://www.Linux-Backup.net/`)
- Security info HOWTO by Alvin Oga (`http://www.Linux-Sec.net/`)
- Various UNOFFICIAL sources for APT (`http://www.apt-get.org/`)
- Laptop Ethernet Configuration (`http://www.orthogony.com/gjw/lap/lap-ether-intro.html`)

付録 A

付録

A.1 著者

Debian リファレンス はまず Osamu Aoki `<osamu\#at\#debian.org>` により 個人的なインストールメモとして書かれ、結局 “クイックリファレンス” と呼ばれました。 コンテンツの多くは “debian-users” メーリングリストのアーカイブ から取られました。又、“Debian Installation Manual” および “Debian Release Notes” も参照されました。

Debian Documentation Project (`http://www.debian.org/doc/ddp`) (DDP) に関して 非常にアクティブな人物であり、現在の “The Debian FAQ” のメンテナである Josip Rodin の助言に従い、本文書は “Debian リファレンス” と改名し、 “The Debian FAQ” からリファレンス形式の内容としていくつかの章をマージしています。

本文書は次に挙げる QREF チームメンバーにより編集、翻訳と 拡張が行われています。

- 原作者
  - Osamu Aoki (青木 修) `<osamu\#at\#debian.org>` (leader: all contents)
- 英語版の校正及び追加の貢献
  - Esko Araj.rvi `<edu\#at\#iki.fi>` (etch アップデート)
  - Thomas Hood `<jdthood\#at\#yahoo.co.uk>` (ネットワーク関連)
  - Brian Nelson `<nelson\#at\#bignachos.com>` (特に X 関連)
  - David Sewell `<dsewell\#at\#virginia.edu>` (退任)
  - Jan Michael C Alonzo `<jmalonzo\#at\#spaceants.net>`
  - Daniel Webb `<webb\#at\#robust.colorado.edu>`
  - および全翻訳者からのフィードバック
- フランス語訳
  - Guillaume Erbs `<gerbs\#at\#free.fr>` (leader: fr)
  - Renald Casagraude `<rcasagraude\#at\#interfaces.fr>`
  - Jean-Pierre Delange `<adeimantos\#at\#free.fr>`
  - Daniel Desages `<daniel\#at\#desages.com>`
- イタリア語訳
  - Davide Di Lazzaro `<mc0315\#at\#mclink.it>` (leader: it)
- ポルトガル (ブラジル) 語訳
  - Paulo Rogerio Ormenese `<pormenese\#at\#uol.com.br>` (leader: pt-br)

  - Andre Luis Lopes <andrelop\#at\#ig.com.br>
  - Marcio Roberto Teixeira <marciotex\#at\#pop.com.br>
  - Rildo Taveira de Oliveira <to_rei\#at\#yahoo.com>
  - Raphael Bittencourt Simoes Costa <raphael-bsc\#at\#bol.com.br>
  - Gustavo Noronha Silva <kov\#at\#debian.org> (coordinator)
- スペイン語訳
  - Walter Echarri <wecharri\#at\#infovia.com.ar> (leader: es)
  - Jose Carreiro <ffx\#at\#urbanet.ch>
- ドイツ語訳
  - Jens Seidel <tux-master\#at\#web.de> (leader: de)
  - Willi Dyck <wdyck\#at\#gmx.net>
  - Stefan Schroeder <stefan\#at\#fkp.uni-hannover.de>
  - Agon S. Buchholz <asb\#at\#kefk.net>
- ポーランド語訳 PDDP (`http://debian.linux.org.pl`):
  - Marcin Andruszkiewicz
  - Mariusz Centka <mariusz.centka\#at\#debian.linux.org.pl>
  - Bartosz Feński <fenio\#at\#debian.linux.org.pl> (leader: pl)
  - Radosław Grzanka <radekg\#at\#debian.linux.org.pl>
  - Bartosz 'Xebord' Janowski
  - Jacek Lachowicz
  - Rafał Michaluk
  - Leonard Milcin, Jr.
  - Tomasz Z. Napierała <zen\#at\#debian.linux.org.pl>
  - Oskar Ostafin <cx\#at\#debian.linux.org.pl>
  - Tomasz Piękoś
  - Jacek Politowski
  - Mateusz Prichacz <mateusz\#at\#debian.linux.org.pl>
  - Marcin Rogowski
  - Paweł Różański
  - Mariusz Strzelecki
  - Krzysztof Ścierski
  - Przemysław Adam Śmiejek <tristan\#at\#debian.linux.org.pl>
  - Krzysztof Szynter
  - Mateusz Tryka <uszek\#at\#debian.linux.org.pl>
  - Cezary Uchto
  - Krzysztof Witkowski <tjup\#at\#debian.linux.org.pl>
  - Bartosz Zapałowski <zapal\#at\#debian.linux.org.pl>
- 中国語 (簡体字) 訳
  - Hao "Lyoo" LIU <iamlyoo\#at\#163.net>
  - Ming Hua <minghua\#at\#rice.edu>
  - Xiao Sheng Wen <atzlinux\#at\#163.com> (leader: zh-cn)
  - Haifeng Chen <optical.dlz\#at\#gmail.com>
  - Xie Yanbo <xieyanbo\#at\#gmail.com>
  - easthero <easthero\#at\#gmail.com>
- 中国語 (繁体字) 訳
  - Asho Yeh <asho\#at\#debian.org.tw> (leader: zh-tw)

  - Tang Wei Ching <`wctang\#at\#csie.nctu.edu.tw`> (ex-leader: zh-tw)
- 日本語訳
  - Shinichi Tsunoda (角田 慎一) <`tsuno\#at\#ngy.1st.ne.jp`> (leader: ja)
  - Osamu Aoki (青木 修) <`osamu\#at\#debian.org`>
- フィンランド語訳
  - Esko Araj.rvi <`edu\#at\#iki.fi`> (leader: fi)

QREF はもともとの文書のタイトル “Quick Reference…” の短縮であり、qref.sourceforge.net でのプロジェクト名でもあります。

本文書を書くためにはまず Debian システム上の多くの マニュアルページや info ページが参照されました。 作者が公平な引用と考える程度で、マニュアルの多く部分、特に コマンド定義が本文書の形式と目標に適合するように注意深く 編集した後で部分的に成句として使用されました。

‘Debian の基礎知識’ on page 5 のほとんどの内容は元々は “The Debian FAQ” (March 2002) のものです。
- 5. The Debian FTP アーカイブ: `ftparchives.sgml` (章全体)
- 6. Debian パッケージ管理システムの基礎: `pkg_basics.sgml` (章全体)
- 7. Debian パッケージ管理ツール: `pkgtools.sgml` (章全体)
- 8. Debian system を最新に保つ: `uptodate.sgml` (章全体)
- 9. Debian と kernel: `kernel.sgml` (章全体)
- 10. Debian GNU/Linux のインストールをカスタマイズ: `customizing.sgml` (部分的)

“Debian FAQ” の章は Debian システムの変更を反映するために 大きな再構築を行った後に本文書に含まれました。 現在は両文書共に同時に更新されています。

オリジナルの “Debian FAQ” は J. H. M. Dassen (Ray) と Chunk Stickelman により作成と維持が行われました。書き直された “Debian FAQ” の著者は Susan G. Kleinmann と Sven Rudolph です。 その後、“Debian FAQ” は Santiago Vila がメンテナンスしました。 現在のメンテナは Josip Rodin です。

“Debian FAQ” の情報の一部は次に挙げるリストからもたらされました。
- Bruce Perens (`http://www.perens.com/`) による Debian-1.1 リリースアナウンス。
- Ian Jackson (`http://www.chiark.greenend.org.uk/~ijackson/`) による Linux FAQ。
- Debian Mailing List Archives (`http://lists.debian.org/`),
- dpkg programmers’ manual と Debian Policy manual (‘参考文献’ on page 247 参照)
- 多くの開発者、ボランティア、そして β テスタ。
- そして著者たちのあてにならない記憶 :-)

“Tutorial” セクションのうちいくらかは次の文書が由来です。
- Havoc Pennington, Oliver Elphick, Ole Tetlie, James Treacy, Craig Sawyer, そして Ivan E. Moore II による “Debian Tutorial” (本文書は Larry Greenfield による “Linux User’s Guide” が由来です。)
- John Goerzen と Ossama Othman による “Debian GNU/Linux: Guide to Installation and Usage”

著者は本文書を可能にするのを手助けしてくださった全ての人に感謝します。

A.2 保証

私は専門家では無いので、一般に Debian 又は Linux に関する完全な知識を 持っているふりをするつもりはありません。私が考慮するセキュリティは ホームユースにだけ適合しているかもしれません。

本文書は権威のあるガイドの置き換えにはなり得ません。

全ての保証を放棄します。全ての商標はそれぞれの商標保有者が所有しています。

A.3 フィードバック

本文書に対するコメントや追加は大歓迎です。 `debian-reference` パッケージあるいは翻訳版パッケージに 対して Debian BTS system (`http://bugs.debian.org/`) にメールを 送ってください。 `reportbug` を使うと簡単に詳細な bug report を提出できます。 また、`<osamu\#at\#debian.org>` (Osamu Aoki (`http://people.debian.org/~osamu/`)) に英語でメールを送るか、翻訳者に日本語でメールを送ることもできます。

著者はアメリカ合衆国に居住していましたが、英語の非ネーティブスピーカです。 文法の修正は大歓迎です。

最良の形のフィードバックは SGML 版に対するパッチですが、テキスト版に対する パッチも歓迎しています。 公式サイトについては '公式文書' on page 1 をご覧ください。

本文書を作成するのに使用されるオリジナルの SGML ファイルは、`:pserver:anonymous@cvs.sf.net/cvsroot/qref` にある CVS 又は `http://qref.sourceforge.net/Debian/qref.tar.gz` から取得できます。

A.4 文書フォーマット

本文書は DebianDoc SGML DTD (LinuxDoc SGML から書き直された) を用いて書かれました。DebianDoc SGML システムにより、1つのソースから さまざまな形式のフォーマットでファイルを作成できるようになりました。 すなわち、本文書は HTML, プレーンテキスト, TeX DVI, Postscript, PDF そして GNU info 形式で見ることができます。

DebianDOC SGML 用の変換ユーティリティは Debian パッケージ `debiandoc-sgml` にあります。

A.5 Debian 迷路

Linux システムはネットワーク化されたコンピュータのための非常に パワフルなコンピュータプラットフォームです。しかし、Linux の全能力を 利用する方法を学ぶことはたやすいことではありません。 プリンタの設定が良い例です。

“ソースコード” という完全かつ詳細なマップが存在します。 これは非常に正確ですが理解しがたいものです。又、HOWTO や mini-HOWTO と呼ばれるリファレンスもあります。これらは理解しやすいですが、詳細に 渡り過ぎて概観を失いがちです。いくつかのコマンドを実行する必要がある時に、 長大な HOWTO の該当する章を探すのに骨が折れることが時々ありました。

Linux システム設定の迷路を案内するため、自分だけのクイックリファレンスとしてシンプルなメモ書きをテキスト形式で書きはじめました。このメモリストは段々大きくなっていき、さらにメモを作る合間に debiandoc を習いました。その産物が この *Debian* リファレンス です。

A.6 Debian の格言

Debian メーリングリストにいくつかの興味深い格言があります。

- “This is Unix. It gives you enough rope to hang yourself.” —Miquel van Smoorenburg `<miquels@cistron.nl>`
- “Unix IS user friendly… It's just selective about who its friends are.” —Tollef Fog Heen `<tollef@add.no>`