debian

Debian リファレンス

[FAMILY Given]

本 Debian リファレンス (第 2.77 版) (2021-01-10 06:32:51 UTC) はシステムインストール後のユーザー案内書として、Debian システムの広範な概論を提供します。本書は非開発者を対象に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説明します。

COLLABORATORS

	TITLE : Debian リファレンス	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	[FAMILY Given]	January 10, 2021	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

List of Tables

Abstract

序章

このDebian リファレンス (第 2.77 版) (2021-01-10 06:32:51 UTC) はシステムインストール後のユーザー向け案内書として Debian のシステム管理に関する概論の提供を目指しています。

本書が対象とする読者は、GNU/Linux システムがどう機能するかを理解するのに、シェルスクリプトぐらいは学ぶ気はあるが、全ての C のソースまで読む気がない人です。

インストールの方法は、以下を参照ください:

- 現行安定システム用 Debian GNU/Linux インストールガイド
- 現行テスティング (testing) システム用 Debian GNU/Linux インストールガイド

免責事項

一切保証は致しません。全ての商標はそれぞれの商標の所有者の財産です。

Debian システム自体は動く標的です。このため最新状況を反映した正確な記述は困難です。現行の不安定版の Debian システムを用いて本書は記していますが、皆様が読まれる時点ではすでに記載内容が古くなっているでしょう。

本書はあくまで二次的参考文献として扱って下さい。本書は正式の案内書を置き換えません。著者及び本書への貢献者は本書中の誤謬や欠落や曖昧さが引き起こす結果に一切責任を負いません。

Debian とはなにか

Debian プロジェクトはフリーなオペレーティングシステムを作ろうという共通目的を持った個人の集団です。そのディストリビューションは次の特徴があります。

- ソフトウェアの自由へのコミットメント: Debian 社会契約と Debian フリーソフトウエアーガイドライン (DFSG)
- インターネット上の分散型の無償ボランティア活動: https://www.debian.org
- 多数のプリコンパイルされた高品質のソフトウエアーパッケージ
- セキュリティーアップデートへの平易なアクセス提供による、安定性とセキュリティーの重視
- 不安定版 `unstable` やテスト版 `testing` アーカイブによる、最新のソフトウエアーへの円滑なアップグレードの重視
- 多数のサポートされたハードウエアーアーキテクチャー

Debian の中のフリーソフトウエアー構成要素は、GNU や Linux や BSD や X や ISC や Apache や Ghostscript や Common Unix Printing System や Samba や GNOME や KDE や Mozilla や LibreOffice や Vim や TeX や LaTeX や DocBook や Perl や Python や Tcl や Java や Ruby や PHP や Berkeley DB や MariaDB や PostgreSQL や Exim や Postfix や Mutt や FreeBSD や OpenBSD や Plan 9 やその他の多くの独立のフリーソフトウエアーのプロジェクトに由来します。Debian はこの多種多様なフリーソフトウエアーを 1 つのシステムにまとめ上げます。

本書について

編集指針

本書の作成にあたり次の編集指針を守りました。

- 概論を提供し枝葉末節は省略します。(全体像)
- 簡潔を心がけました。(**KISS**)
- 車輪の再発明をしません。(既存の参考文献へのポインターの利用)
- 非 GUI ツールとコンソールを重視します。(シェル例示を使用)
- 客観的であるようにします。(ポプコン等の利用)

ティップ
私はシステムの階層的側面やシステムの低レベルを明らかにしようとしました。

前提条件

警告
本文書だけに頼らず自分で答えを見出す努力をしっかりすることを期待します。本文書は効率的なスタートポイントを提供するだけです。

一義的情報源から自分自身で解決策を探し出すべきです。

- The Debian Administrator's Handbook
- 一般的情報は https://www.debian.org にある Debian サイト
- "`/usr/share/doc/<package_name>`" ディレクトリー下にある文書
- Unix スタイルのマンページ: "`dpkg -L <package_name> |grep '/man/man.*/'`"
- GNU スタイルの **info** ページ: "`dpkg -L <package_name> |grep '/info/'`"
- バグレポート: http://bugs.debian.org/<package_name>
- 変化中の事や特定案件に関しては、https://wiki.debian.org/ にある Debian の Wiki
- http://tldp.org/ にある Linux 文書プロジェクト (TLDP) の HOWTO 文書
- http://www.unix.org/ にある Open Group の The UNIX System Home Page 中の Single UNIX Specification
- https://www.wikipedia.org/ にある Wikipedia のフリーの百科事典

注意
詳細な文書を読むには、"`-doc`" をサフィクスとする対応する文書パッケージをインストールする必要があるかもしれません。

文書様式

bash(1) シェルコマンドの例示をする次のような簡略化した表現スタイルで本書は情報を提供します。

```
# <root b'' ア b''b'' カ b''b'' ウ b''b'' ン b''b'' ト b''b'' か b''b'' ら b''b'' の b''b'' コ
    b''b'' マ b''b'' ン b''b'' ド b''>
$ <b'' ユ b''b'' ー b''b'' ザ b''b'' ー b''b'' ア b''b'' カ b''b'' ウ b''b'' ン b''b'' ト
    b''b'' か b''b'' ら b''b'' の b''b'' コ b''b'' マ b''b'' ン b''b'' ド b''>
```

これらのシェルプロンプトは使われるアカウントを区別します。これはちょうど環境変数として: "PS1='\$'" と"PS2=' '" を設定した場合に相当します。これらの環境変数値はあくまで本書の読みやすさのためで、実際のインストール済みシステではほとんど見かけません。

注意
"PS1='\$'" と"PS2=' '" という環境変数値の意味は bash(1) を参照下さい。

システム管理者が行うべきアクションは命令文で書かれています: 例えば、「シェルに各コマンド文字列をタイプ後毎にエンターキーをタイプします。」(必ずしも「〜しましょう。」とはせず簡潔に訳しています。)

英語では、テーブル中の説明や類似のコラムには、パッケージ説明の慣習に従い、定冠詞抜も不定冠詞も抜きの名詞句が入ります。これらには、マンページのコマンドの短い説明の慣習に従った頭の"to" 抜きの不定詞句が代わりに名詞句として入ることもあります。変だなとお考えの方もあるとは存じますが、これは本文書をできるだけ簡潔にするための著者の恣意的な文体の選択です。(対応部分を文切り型の名詞句的表現に訳しています。)

注意
コマンド名を含めて固有名詞はその位置によらず大文字・小文字の区別を保持します。

本文中に引用されるコマンドの断片はダブルクォーテションマーク間にタイプライターフォントで書き"`aptitude safe-upgrade`" のように表現されます。

本文中に設定ファイルから引用された文字データーはダブルクォーテションマーク間にタイプライターフォントで書き"`deb-src`" のように表現されます。

コマンドはその名前をタイプライターフォントで書き、場合によってはその後ろにマンページのセクション番号を括弧中に入れて書き bash(1) のように表現されます。読者は次の様にタイプして情報を得るように心がけて下さい。

```
$ man 1 bash
```

マンページはその名前をタイプライターフォントで書き、その後ろにマンページのセクション番号を括弧中に入れて書き sources.list(5) のように表現されます。読者は次の様にタイプして情報を得るように心がけて下さい。

```
$ man 5 sources.list
```

info ページはダブルクォーテションマーク間にタイプライターフォントというコマンドの断片形式で書き"`info make`" のように表現されます。読者は次の様にタイプして情報を得るように心がけて下さい。

```
$ info make
```

ファイル名はダブルクォーテションマーク間にタイプライターフォントで書き"`/etc/passwd`" のように表現されます。読者は次の様にタイプして情報を得るように心がけて下さい。

```
$ sensible-pager "/etc/passwd"
```

ディレクトリー名はダブルクォーテションマーク間にタイプライターフォントで書き"`/etc/apt/`" のように表現されます。読者は次の様にタイプして情報を得るように心がけて下さい。

```
$ mc "/etc/apt/"
```

パッケージ名はその名をタイプライターフォントで書き”vim” のように表現されます。読者は次の様にタイプして情報を得るように心がけて下さい。

```
$ dpkg -L vim
$ apt-cache show vim
$ aptitude show vim
```

文書は､その場所のファイル名でダブルクォーテションマーク間にタイプライターフォントで書き”/usr/share/doc/ba
や”/usr/share/doc/base-passwd/users-and-groups.html” のように表現されたり、その場所の URL で https://www.debian.org のように表現されます。読者は次の様にタイプして情報を得るように心がけて下さい。

```
$ zcat ”/usr/share/doc/base-passwd/users-and-groups.txt.gz” | sensible-pager
$ sensible-browser ”/usr/share/doc/base-passwd/users-and-groups.html”
$ sensible-browser ”https://www.debian.org”
```

環境変数は、頭に”$” がついた名前をダブルクォーテションマーク間にタイプライターフォントで書き、”$TERM” のように表現されます。読者は次の様にタイプして情報を得るように心がけて下さい。

```
$ echo ”$TERM”
```

ポプコン

ポプコンのデーターは各パッケージの客観的人気の指標として提示されいます。それがダウンロードされた日付は 2021-01-10 06:32:17 UTC で、178828 を越すバイナリーパッケージ数と 26 のアーキテクチャーにまたがる 197967 つの提出レポートからなります。

注意
amd64 の不安定版 unstable アーカイブは現在高々 62716 つのパッケージしか含みません。ポプコンデーターは多くの旧式設置システムからのレポートを含みます。

”votes” を意味する”V:” が前についたポプコンの数は”1000 * (PC で最近実行されたパッケージに関するポプコン提出)/(全ポプコン提出)” として計算される。

”installs” を意味する”I:” が前についたポプコンの数は”1000 * (PC にインストールされているパッケージに関するポプコン提出)/(全ポプコン提出)” として計算される。

注意
Popcon の数字はパッケージの重要性の絶対指標と考えるべきでません。統計を曲げる多くの因子があります。例えば、Popcon に参加しているシステムの一部は”/bin” などのディレクトリーをシステム性能向上のために”noatime” オプションでマウントすることで当該システムから”vote” することを実質的に禁止しているかもしれません。

パッケージサイズ

各パッケージの客観的指標としてパッケージサイズデーターも提供されます。それは”apt-cache show” や”aptitude show” コマンドが (現在の amd64 アーキテクチャー上の unstable リリース上で) 表示する”Installed-S
です。サイズは KiB (Kibibyte = 1024 バイト単位) で表示されます。

注意
小さなパッケージサイズのパッケージは unstable リリース中の当該パッケージが内容のある他パッケージを依存関係でインストールするためのダミーパッケージだからかもしれません。

注意
”(*)” が後ろについたパッケージのサイズは、unstable リリース中にパッケージが無く experimental リリース中のパッケージサイズが代用されたことを示します。

本書へのバグ報告

何かこの文書に問題を発見した場合には、debian-reference パッケージに対して reportbug(1) を用いてバグ報告をして下さい。プレーンテキストバージョンかソースに対する”diff -u” による修正提案を含めて下さい。

新規ユーザーへのリマインダー

新規ユーザーへのリマインダーを以下に記します:

- データをバックアップしましょう
- パスワードとセキュリティーキーを保護する
- KISS (keep it simple stupid、簡潔性尊重原則)
  - システムを過剰にエンジニアリングしてはいけません
- ログファイルを読みましょう
  - 最初のエラーが大事なエラーです
- RTFM (read the fine manual、良く書かれているマニュアルを読みましょう)
- 質問する前にインターネットを検索しましょう
- 必要もないのに root になってはいけません
- パッケージ管理システムを改変してはいけません
- 自分が理解していないことを入力してはいけません
- (全セキュリティレビューを受ける前に) ファイルのパーミッションを変更してはいけません
- あなたの変更をテストするまで root シェルを離れてはいけません
- 常に代替ブートメディア (USB メモリースティック、CD、…) を確保しましょう

新規ユーザーへの引用文

新規ユーザーを啓蒙する Debian のメーリングリストで見つけた興味深い引用文を記します。

- ”This is Unix. It gives you enough rope to hang yourself.” 「これは Unix です。首を括るのに十分なロープをあてがってくれますよ。」--- Miquel van Smoorenburg <miquels at cistron.nl>
- ”Unix IS user friendly... It’s just selective about who its friends are.” 「Unix はユーザーフレンドリー (使う人に優しい) です... 誰にフレンドリー (優しく) にするかの人見知りするだけです。」--- Tollef Fog Heen <tollef at add.no>

ウイキペディアの”Unix philosophy” という記事に、おもしろい格言集があります。

Chapter 1

GNU/Linux チュートリアル

コンピューター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、実際に自ら使って学ぶことが必要です。円滑なスタートが出きるように、いくつかの基本的なポイントを説明します。

Debian GNU/Linux の強力なデザインはマルチユーザー、マルチタスクという Unix オペレーティングシステムに由来します。これら Unix と GNU/Linux の特徴や類似点の強力さを活用することを覚えましょう。

Unix 対象の文書を避けたり、GNU/Linux に関する文書だけに頼ることは、有用な情報を見逃すことになるので止めましょう。

注意
Unix 的システムをコマンドラインツールで少々使った経験があれば、私がここで説明することはすべてご存知でしょう。リアリティーチェックと記憶を呼び戻すのにこれを使って下さい。

1.1 コンソールの基礎

1.1.1 シェルプロンプト

X Window システムを `gdm3` 等のディスプレーマネージャーとともにインストールした場合以外には、システム起動の際に文字の login スクリーンが現れます。あなたのホスト名が `foo` と仮定すると、login プロンプトは次に示すような見えます。

```
foo login:
```

GNOME や KDE のような GUI 環境をインストールした場合には、Ctrl-Alt-F1 とすることで login プロンプトが出て、Alt-F7 とすることで GUI 環境に戻れます (詳細は下記の項1.1.6参照下さい)。

login プロンプトであなたのユーザー名 (例えば `penguin`) を打鍵し Enter キーを押します。さらにあなたのパスワードを打鍵し Enter キーを再び押します。

注意
Unix の伝統に従い、Debian システムではユーザー名とパスワードに関して大文字小文字の区別をします。ユーザー名は通常小文字のみから選ばれます。最初のユーザーアカウントは通常インストールの際に作られます。追加のユーザーアカウントは root によって adduser(8) を用いて作られます。

”`/etc/motd`” (本日のメッセージ: Message Of The Day) に保存されている歓迎メッセージとコマンドプロンプトを表示しシステムが起動されます。

```
Debian GNU/Linux jessie/sid foo tty1
foo login: penguin
Password:
Last login: Mon Sep 23 19:36:44 JST 2013 on tty3
Linux snoopy 3.11-1-amd64 #1 SMP Debian 3.11.6-2 (2013-11-01) x86_64

The programs included with the Debian GNU/Linux system are free software;
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.
foo:~$
```

これであなたはシェルの中にいます。シェルはあなたからのコマンドを解釈します。

1.1.2 X の下でのシェルプロンプト

インストールの際に”Desktop environment” タスクを選定し GNOME の gdm3 とともに X Window システムをインストールした場合には、システムの起動するとグラフィカルな login プロンプトのスクリーンが表示されます。あなたのユーザー名とパスワードを入力することで非特権ユーザーアカウントに login できます。タブ (tab) を用いたりマウスの第一クリックを用いるとユーザー名とパスワードの間を行き来できます。

gnome-terminal(1) や rxvt(1) や xterm(1) のような `x-terminal-emulator` プログラムを X の下で起動するとシェルプロンプトが得られます。GNOME デスクトップ環境下では、”Applications” → ”Accessories” → ”Terminal” とクリックしてもうまくいきます。

次の項1.1.6も参照下さい。

デスクトップ環境 (例えば `fluxbox`) 次第ではメニューの起点がよく分からないことがあります。そんな時はデスクトップスクリーンの背景を (右) クリックしてメニューが表示されることを期待しましょう。

1.1.3 root アカウント

root アカウントはスーパーユーザーとか特権ユーザーとも呼ばれます。このアカウントからは次のようなシステム管理活動ができます。

- ファイルパーミッションによらずシステム上の任意ファイルに関しての、読出し・書込み・削除
- システム上のいかなるファイルに関して、ファイルの所有者やパーミッション設定
- システム上の非特権ユーザーのパスワードを設定
- パスワード無しに任意アカウントへの login

root アカウントの権限を使うには、この無制限の権限ゆえ配慮と責任ある行動が求められます。

警告
root のパスワードを他人に決して教えてはいけません。

注意
ファイル (Debian システムにとってはファイルの一種である CD-ROM 等のハードウエアーデバイスも含む) パーミッションは、非 root ユーザーによるそのファイルの使用やアクセスをできなくなくすることがあります。この様な状況の下では root アカウントを使うことが簡便なテスト法ですが、問題解決はファイルパーミッションとユーザーのグループのメンバーシップを適正に設定する必要があります (項1.2.3参照下さい)。

1.1.4 root シェルプロンプト

root のパスワードを使って root のシェルプロンプトを使えるようにする基本的な方法を次に記します。

- 文字ベースのログインプロンプトに `root` と入力します。
- GNOME デスクトップ環境下で、”Applications” → ”Accessories” → ”Root Terminal” とクリックします。
- どのユーザーシェルプロンプトからでも”`su -l`” と入力します。
  - 現ユーザーの環境を一切引き継がません。
- どのユーザーシェルプロンプトからでも”su” と入力します。
  - 現ユーザーの環境を一部引き継ぐ。

1.1.5 GUI のシステム管理ツール

デスクトップのメニューが GUI のシステム管理ツールを適切な権限とともに自動的に起動しない場合、gnome-terminal(1) や rxvt(1) や xterm(1) のような X ターミナルエミュレーターの root シェルプロンプトから起動できます。項1.1.4 and 項7.8.5を参照下さい。

警告
gdm3(1) 等のディスプレーマネージャーのプロンプトに `root` と入力して、X ディスプレー / セッションマネージャーを root アカウントのもとで決して起動してはいけません。

警告
クリチカルな情報が表示されている際には、あなたの X スクリーンを覗き見られるかもしれないのでリモートの信頼できない GUI プログラムを決して実行してはいけません。

1.1.6 仮想コンソール

デフォールトの Debian システムでは、6 つの切り替え可能な VT100 様の文字コンソールが利用でき、Linux ホスト上で直接コマンドシェルを起動できます。GUI 環境下でない場合は、`Left-Alt-key` と `F1`—`F6` の中の一つのキーを同時に押すことで仮想コンソール間の切り替えができます。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、。マルチユーザー環境を提供します。このマルチユーザー環境は Unix の偉大な機能で、癖になります。

X Window システムの下では、`Ctrl-Alt-F1` キーを押す、つまり `left-Ctrl-key` と `left-Alt-key` と `F1-key` キーを同時に押すと文字コンソール 1 にアクセスできます。通常仮想コンソール 7 で実行されている X Window システムへは `Alt-F7` を押すことにより戻れます。

これとは別の方法で、例えば仮想ターミナル 1 という今とは違う仮想ターミナルへの変更がコマンドラインから出来ます。

```
# chvt 1
```

1.1.7 コマンドプロンプトからの退出方法

コマンドプロンプトで `Ctrl-D`、つまり `left-Ctrl-key` と `d-key` の同時押しをするとシェルでのアクティビティーを終了できます。文字コンソールの場合は、こうすると login プロンプト戻ります。これらのコントロール文字は通常”control D” と大文字を使って表記されますが、Shift キーを押す必要はありません。また `Ctrl-D` に関する簡略表記 `^D` も使われます。この代わりに”exit” とタイプすることができます。

x-terminal-emulator(1) にあっては、このようにすることで `x-terminal-emulator` のウィンドウが閉じることができます。

1.1.8 システムをシャットダウンする方法

ファイル操作の際にパーフォーマンス向上のためにメモリーへのデーターのキャッシュがされる他の現代的な OS と同様に、Debian システムでも電源を安全に切る前に適正なシャットダウン手順を取る必要があります。これはすべてのメモリー上の変更を強制的にディスクに書き出しすことで、ファイルの完全性を維持するためです。ソフトウエアー電源コントロールが利用できる場合、シャットダウン手続きはシステムの電源を自動的に落とします。(これがうまくいかない時には、シャットダウン手続きの後で数秒間電源ボタンを押す必要があるかもしれません。)

通常のマルチユーザーモードからのシステムのシャットダウンがコマンドラインから出来ます。

```
# shutdown -h now
```

シングルユーザーモードからのシステムのシャットダウンがコマンドラインから出来ます。

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

この他に、”`/etc/inittab`” に”`ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -h now`” と書かれていれば、`Ctrl-Alt-Delete` (`left-Ctrl-key` と `left-Alt-Key` と `Delete` の同時押し) を入力するシャットダウン方法もあります。

項6.9.6を参照下さい。

1.1.9 まともなコンソールの復元

例えば”`cat <some-binary-file>`” のような変な事をした後でスクリーンが無茶苦茶になった場合、コマンドプロンプトに”`reset`” と入力して下さい。このときコマンドを入力してもスクリーンには読み取れる表示がされないかもしれません。”`clear`” とすればスクリーンが消去できます。

1.1.10 初心者向け追加パッケージの提案

デスクトップ環境タスク抜きの最小限インストレーション Debian システムですら基本的な Unix 機能は提供されますが、コマンドラインや curses に基づく `mc` や `vim` 等のいくつかの文字ターミナルパッケージを apt-get(8) を使って次のように追加インストールすることから始めることを初心者にお薦めします。

```
# apt-get update
 ...
# apt-get install mc vim sudo
 ...
```

既にこれらのパッケージがインストールされている場合には、新しいパッケージはインストールされません。

いくつかの参考資料を読むのも良いことです。

これらのパッケージの一部を次のようにしてインストールします。

```
# apt-get install package_name
```

パッケージ	ポプコン	サイズ	説明
mc	V:59, I:236	1482	テキストモードの全画面ファイルマネージャー
sudo	V:563, I:806	4555	ユーザーに限定的な root 権限を与えるプログラム
vim	V:106, I:398	3231	Unix テキストエディター Vi IMproved (改良版 Vi)、プログラマーのためのテキストエディター (標準版)
vim-tiny	V:62, I:970	1553	Unix テキストエディター Vi IMproved (改良版 Vi)、プログラマーのためのテキストエディター (軽量版)
emacs-nox	V:4, I:17	18364	GNU project Emacs, the Lisp based extensible text editor
w3m	V:31, I:284	2289	テキストモード WWW ブラウザー
gpm	V:11, I:17	530	テキストコンソール上の Unix 式のカットアンドペースト (daemon)

Table 1.1: 興味あるテキストモードのプログラムパッケージのリスト

パッケージ	ポプコン	サイズ	説明
doc-debian	I:854	166	Debian プロジェクトの文書、(Debian FAQ) 他
debian-policy	I:36	4306	Debian ポリシーマニュアルと関連文書
developers-reference	I:6	1917	Debian 開発者のためのガイドラインと情報
maint-guide	I:4	987	Debian 新メンテナ向けガイド
debian-history	I:1	4285	Debian プロジェクトの歴史
debian-faq	I:849	817	Debian FAQ (よくある質問集)

Table 1.2: 有用な文書パッケージのリスト

1.1.11 追加のユーザーアカウント

次の練習のためにあなたのメインのユーザーアカウントを使いたくない場合には、例えば `fish` という追加のユーザーアカウントを作成できます。root シェルプロンプトで次のように入力します。

```
# adduser fish
```

すべての質問に返事をします。

こうすることで `fish` という名前の新規アカウントが作られます。練習の後で、このユーザーとそのホームディレクトリーは次のようのすれば削除できます。

```
# deluser --remove-home fish
```

1.1.12 sudo の設定

ラップトップ PC 上のデスクトップの Debian システム等のような典型的単一ユーザーワークステーションでは次のような単純な sudo(8) の設定をして、非特権ユーザー (例えば `penguin`) に管理者権限を (root パスワードではなく) ユーザー自身のパスワードで与えることがよくあります。

```
# echo "penguin  ALL=(ALL) ALL" >> /etc/sudoers
```

これに代え、次のようにして非特権ユーザー `penguin` にパスワード一切無しに管理者権限を与えることもよくあります。

```
# echo "penguin  ALL=(ALL) NOPASSWD:ALL" >> /etc/sudoers
```

このトリックの使用は、単一ユーザーワークステーション上であなた自身が管理者でユーザーである際のみに限るべきです。

警告
システムセキュリティー上非常に悪い事態を招くので、マルチユーザーワークステーション上の通常ユーザーアカウントに対してこの様な設定をしてはいけません。

注意
上記例のような `penguin` のパスワードとアカウントは root パスワードや root アカウント同様の保護が必要です。

注意
この文脈上の管理者権限はワークステーションに関するシステム管理業務をする権限を与えられた人に属します。そのような権限と能力を持っていなければ、あなたの会社の管理部門の管理職や上司とはいえこのような権限を与えてはいけません。

注意
特定デバイスや特定ファイルへのアクセスの権限を与えるには、sudo(8) をつかって得た `root` 権限を用いるのではなく、**group** を使って限定的アクセス与えることを考えるべきです。

注意
sudo(8) を使ってもう少し工夫された注意深い設定をすれば、共有システム上の他のユーザーに root パスワードを教えること無く限定的管理権限を許可することができます。こうすることは、誰が何をしたかを明らかにするので、複数の管理者がいるホストにおける責任の所在を明らかにします。ただ、誰にもそんな権限を与えたく無いかもしれません。

1.1.13 お遊びの時間

非特権ユーザーアカウントを使う限り全くリスク無く Debian システムでお遊びをする準備万端です。

何故なら、たとえデフォールトのインストール後ですら Debian システムは非特権ユーザーがシステムに損害を与えられないように適正なファイルパーミッションが設定されているからです。もちろん悪用可能な穴が残っているかもしれませんが、こんな問題まで心配する人はこのセクションを読んでいるべきではなく、Securing Debian Manual を読むべきです。

Debian システムを Unix 的システムとして次に学びましょう:

- 項1.2 (基本コンセプト)
- 項1.3 (サバイバル手法)
- 項1.4 (基本手法)
- 項1.5 (シェルのメカニズム)
- 項1.6 (文字処理手法)

1.2 Unix-like ファイルシステム

GNU/Linux や他の Unix 的オペレーティングシステムでは、ファイルはディレクトリーに整理されています。すべてのファイルやディレクトリーは、”/” を根 (root) に持つ一本の大きな木 (ツリー) のようにアレンジされています。

このようなファイルやディレクトリーはいくつかのデバイスに展開することができます。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マウントするのに mount(8) が使われます。その逆に、それを切り離すのに umount(8) が使われます。最近の Linux カーネルでは、mount(8) をオプションとともに用いると、ファイルツリーの一部を別のところと結びつけたり、共有・非共有・従属・バインド不可としてファイルシステムをマウントもできます。各ファイルシステムごとの利用可能なマウントオプションは”`/usr/share/doc/linux-doc-*/Documentation/filesystems/`” にあります。

Unix システム上のディレクトリーは、一部の他システム上ではフォルダと呼ばれます。Unix システム上では”`A:`” のようなドライブというコンセプトが無いこと覚えておいて下さい。単一のファイルシステムがあって、そこにすべてが含まれています。これは Windows と比べた際の大きな利点です。

1.2.1 Unix ファイルの基礎

Unix ファイルの基礎は以下です。

- ファイル名は大文字と小文字を区別します。”`MYFILE`” と”`MyFile`” は異なるファイルです。
- ルートディレクトリーはフィルシステムの根 (ルート、root) を意味して、単に”/” と記載されます。これを root ユーザーのホームディレクトリー”`/root`” とは混同しないで下さい。
- 全てのディレクトリーには **”/”** 以外の文字・記号からなる名前がついています。ルートディレクトリーは例外で、その名前は”/” (” スラッシュ” とか” ルートディレクトリー” と読まれます) でその名前を変えることはできません。
- 各ファイルやディレクトリーは、たどっていくとファイルに到達するディレクトリーの列が示される、完全に記述したファイル名とか絶対ファイル名とかパスにより指定されます。これらの 3 つの表現は同義語です。
- 全ての完全に記述したファイル名は”/” ディレクトリーで始まり、ファイル名中の各ディレクトリーやファイル名の間には”/” がはさまります。最初の”/” はディレクトリー名です。その他の”/” は、次のサブディレクトリーとの区別をします。そして最後には実際のファイルの名前がきます。ちょっと混乱しそうですので、次の完全に記述したファイル名の例をご覧下さい: ”`/usr/share/keytables/us.map.gz`”。一方このベース名である、”`us.map.gz`” だけをファイル名と呼ぶ人もあります。
- ルート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ーもまた”`/etc/init.d/`” や”`/usr/local/`” のように、さらにサブディレクトリーに枝別れします。これらの全体をまとめてディレクトリーツリーと呼びます。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ーツリーをルートディレクトリー (”/”) という単一人物の全直系に広がる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ーに親があるとし、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。ルートディレクトリーではない他の場所から始まる相対パスもあります。ディレクトリー”`../`” は親ディレクトリーを参照していることを覚えておきましょう。このような呼び方はディレクトリーのような構造を持つ他の階層的ツリー状のデーター構造体でもよく使われます。
- ハードディスクのような物理デバイスに対応したパス名の要素は存在しません。ここが、パス名に”`C:\`” のようなデバイス名が含まれる RT-11 や CP/M や OpenVMS や MS-DOS や AmigaOS や Microsoft Windows と違う点です。(但し、通常のファイルシステム中に物理デバイスを示すディレクトリー項目はあります。項1.2.2参照下さい。)

注意
ほとんど全ての文字や記号をファイル名中に使えますが、実際そうすることは賢明ではありません。スペースやタブや改行や他の特殊文字: `{ } ( ) [ ] ′ ` ″ \ / > < | ; ! # & ^ * % @ $` はコマンドラインで特別な意味を持つので避けるべきです。名前の中の単語間には、ピリオドやハイフンや下線を選んで区別します。各語頭を”`LikeThis`” のように語頭を大文字にすることもできます。経験を積んだ Linux のユーザーはファイル名中にスペースが入ることを避けます。

注意
”root” (ルート) と言う言葉は”root ユーザー” と言う意味でも” ルートディレクトリー” と言う意味でも使われます。それがいずれかは使われている文脈から明かです。

注意
パスと言う言葉は上述の完全に記述したファイル名に関して使われるばかりではなくコマンドサーチパスにも使われます。どちらの意味かは文脈から明かです。

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/fh
や hier(7)) 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。

ディレクトリー	ディレクトリーの用途
/	ルートディレクトリー
/etc/	システム全体の設定ファイル
/var/log/	システムのログファイル
/home/	全ての非特権ユーザーのホームディレクトリー

Table 1.3: 重要ディレクトリーの使い方のリスト

1.2.2 ファイルシステムの内側

Unix の伝統に従い、Debian/Linux 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に存在する物理データーを表す ファイルシステムを提供し、コンソールスクリーンやリモートのシリアルコンソールなどのハードウェアデバイスとの相互作用が”/dev/” の下に統一された形式で表されています。

Debian/Linux システム上の、各々のファイルやディレクトリーや名前付きパイプ (2 つのプログラムがデーターを共有する方法) や物理デバイスは、それぞれの所有者 (owner) やデーターが所属するグループ (group) や最終アクセス時間などの付帯属性 (attribute) を記述する inode と呼ばれるデーター構造を持ちます。ほとんど全てをファイルシステム表現しようというアイデアは Unix の発明でしたし、現代的な Linux カーネルはこのアイデアを一歩進めています。コンピューター上で実行されているプロセス情報さえファイルシステム中に見つけられます。

このような物理的実体と内部プロセスの抽象的かつ統一された表現は非常にパワフルなので、多くの全く異なるデバイスに同じコマンドを使用して同種の操作が行えます。実行中のプロセスに繋がった特殊なファイルにデーターを書き込むことでカーネルが如何に動作するかまで変更できます。

ティップ
ファイルツリーや物理的実体の間の関係を確認する必要がある際には、mount(8) を引数無しで実行して下さい。

1.2.3 ファイルシステムのパーミッション

Unix 的システムのファイルシステムのパーミッションは次の 3 つの影響されるユーザーのカテゴリーのために定義されています。

- ファイルを所有するユーザー **(user)** (**u**)
- ファイルが所属するグループ **(group)** 中の他ユーザー (**g**)
- ” 世界” や” 全員” とも呼ばれる、全他ユーザー **(other)** (**o**)

ファイルでは、それぞれに対応するパーミッションは次のようになります。

- 読出し **(read)** (**r**) パーミッションはファイル内容確認を可能にします。
- 書込み **(write)** (**w**) パーミッションはファイル内容変更を可能にします。
- 実行 **(execute)** (**x**) パーミッションはファイルをコマンド実行を可能にします。

ディレクトリーでは、対応するパーミッションはそれぞれ次のようになります。

- 読出し **(read)** (**r**) パーミッションはディレクトリー内容リストを可能にします。
- 書込み **(write)** (**w**) パーミッションはディレクトリーへのファイルの追加削除を可能にします。
- 実行 **(execute)** (**x**) パーミッションはディレクトリー内のファイルへのアクセスを可能にします。

ここで、ディレクトリーに関する実行 **(execute)** パーミッションとはディレクトリー内のファイルへの読出しを許可するのみならず、サイズや変更時間のようなアトリビュート閲覧を許可します。

ファイルやディレクトリーのパーミッション情報他を表示するには、ls(1) が使われます。”`-l`” オプション付きでこれを実行すると、次の情報がこの順序で表示されます。

- ファイルのタイプ (最初の文字)
- ファイルのアクセスパーミッション (次の 9 文字。ユーザーとグループと他者の順にそれぞれに対して 3 文字から構成されている)
- 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ー **(user)** の名前
- ファイルが所属するグループ **(group)**
- ファイルのサイズ (文字数、バイト)
- ファイルの日時 (mtime)
- ファイルの名前

文字	意味
`-`	通常ファイル
`d`	ディレクトリー
`l`	シムリンク
`c`	文字デバイス名
`b`	ブロックデバイス名
`p`	名前付きパイプ
`s`	ソケット

Table 1.4: ”`ls -l`” の出力の最初の文字のリスト

root アカウントから chown(1) を使用することでファイルの所有者を変更します。ファイルの所有者又は root アカウントから chgrp(1) を使用することでファイルのグループを変更します。ファイルの所有者又は root アカウントから chmod(1) を使用することでファイルやディレクトリーのアクセスパーミッションを変更します。`foo` ファイルの操作の基本的文法は次の通り。

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

例えば次のようにするとディレクトリーツリーの所有者をユーザー `foo` に変更しグループ `bar` で共有できます。

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

更に特殊なパーミッションビットが 3 つ存在します。

- セットユーザー **ID** ビット (ユーザーの **x** に代えて **s** か **S**)
- セットグループ **ID** ビット (グループの **x** に代えて **s** か **S**)
- スティッキビット (他ユーザーの **x** に代えて **t** か **T**)

ここで、これらのビットの”`ls -l`” のアウトプットはこれらの出力によってかくされた実行ビットが非設定 **(unset)** の場合大文字となります。

セットユーザー **ID** を実行ファイルにセットすると、ユーザーはファイルの所有者 ID (例えば、**root**) を使って実行ファイルを実行することを許可されます。同様に、セットグループ **ID** を実行ファイルにセットすると、ユーザーはファイルのグループ ID (例えば、**root**) を使って実行ファイルを実行することを許可されます。これらの設定はセキュリティーを破壊す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。

セットグループ **ID** をディレクトリーに対して有効にすると、ディレクトリーに作成した全ファイルがディレクトリーのグループに所属するという BSD 的ファイル生成手法が有効になります。

スティッキビットをディレクトリーに対して有効にすると、ディレクトリー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`” のような全員書込み可能ディレクトリーやグループ書込み可能なディレクトリーなどのにあるファイルの内容を安全にするためには、書込みパーミッションを無効にするだけでなく、ディレクトリーにスティッキビットもセットする必要があります。さもなければ、ディレクトリーに書込みアクセスできるユーザーにより、ファイルが削除され、同じ名前で新規ファイルが作成されることを許してしまいます。

ファイルパーミッションの興味ある例を次にいくつか示します。

```
$ ls -l /etc/passwd /etc/shadow /dev/ppp /usr/sbin/exim4
crw------T 1 root root   108, 0 Oct 16 20:57 /dev/ppp
-rw-r--r-- 1 root root     2761 Aug 30 10:38 /etc/passwd
-rw-r----- 1 root shadow   1695 Aug 30 10:38 /etc/shadow
-rwsr-xr-x 1 root root   973824 Sep 23 20:04 /usr/sbin/exim4
$ ls -ld /tmp /var/tmp /usr/local /var/mail /usr/src
drwxrwxrwt 14 root root  20480 Oct 16 21:25 /tmp
drwxrwsr-x 10 root staff  4096 Sep 29 22:50 /usr/local
drwxr-xr-x 10 root root   4096 Oct 11 00:28 /usr/src
drwxrwsr-x  2 root mail   4096 Oct 15 21:40 /var/mail
drwxrwxrwt  3 root root   4096 Oct 16 21:20 /var/tmp
```

chmod(1) を用いて、ファイルパーミッション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 3 桁から 4 桁の数字を用います。

数字	意味
1 桁目 (任意)	セットユーザー **ID** (=4) とセットグループ **ID** (=2) とスティキービット (=1) の和
2 桁目	ユーザーに関する、読出し **(read)** (=4) と書込み **(write)** (=2) と実行 **(execute)** (=1) のファイルパーミッションの和
3 桁目	グループに関して、同上
4 桁目	ユーザーに関して、同上

Table 1.5: chmod(1) コマンドで用いられるファイルパーミッションの数字モード

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列 (2〜10 列) を見て、それをファイルパーミッションのバイナリー表記 (2 進数) (”-” 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パーミッションの 8 進数表現として意味を持ちます。

例えば、次を試してみて下さい:

```
$ touch foo bar
$ chmod u=rw,go=r foo
$ chmod 644 bar
$ ls -l foo bar
-rw-r--r-- 1 penguin penguin 0 Oct 16 21:39 bar
-rw-r--r-- 1 penguin penguin 0 Oct 16 21:35 foo
```

ティップ
シェルスクリプトから”ls -l” で表示される情報にアクセスする必要がある際には、test(1) や stat(1) や readlink(1) のような適切なコマンドの使用を考えるべきです。シェル組込みコマンドの”[” や”test” を使うのも手です。

1.2.4 新規作成ファイルのパーミッションのコントロール: umask

新規作成ファイルのやディレクトリーに適用されるパーミッションは umask シェル組込みコマンドを使うことにより制限できます。dash(1) か bash(1) か builtins(7) をご覧下さい。

```
(b'' フ b''b'' ァ b''b'' イ b''b'' ル b''b'' パ b''b'' ー b''b'' ミ b''b'' ッ b''b'' シ b''b'' ョ
    b''b'' ン b'') = (b'' 要 b''b'' 求 b''b'' さ b''b'' れ b''b'' た b''b'' パ b''b'' ー b''b'' ミ
    b''b'' ッ b''b'' シ b''b'' ョ b''b'' ン b'') & ~(umask b'' 値 b'')
```

umask	作成されるファイルパーミッション	作成されるディレクトリーパーミッション	使い方
0022	-rw-r--r--	-rwxr-xr-x	ユーザーのみにより書込み可
0002	-rw-rw-r--	-rwxrwxr-x	グループにより書込み可

Table 1.6: **umask** 値の例

Debian システムはユーザー専用グループ (UPG) 方式がデフォールト方式です。新規ユーザーがシステムに追加される毎に UPG は作成されます。UPG はそのグループを作成したユーザーと同じ名前を持ち、そのユーザーが UPG の唯一のメンバーです。UPG 方式では、全ユーザーが各自専用のグループを持つので umask を 0002 と設定しても安全です。(一部 Unix 系システムでは全一般ユーザーを 1 つの **users** グループに所属させることがよく行われます。そのような場合には安全のため 0022 と umask を設定します。)

ティップ
~/.bashrc ファイル中に”umask 002” と書いて UPG を有効にしましょう。

1.2.5 ユーザーのグループ (group) のパーミッション

特定のユーザーにグループ許可を適用するには、/etc/group に関しては”sudo vigr” と /etc/gshadow に関しては”sudo vigr -s” を用いて、そのユーザーをグループのメンバーにする必要があります。新規のグループ設定を有効にするにはログアウト後ログイン (もしくは”exec newgrp” を実行) する必要があります。

注意
も し”auth optional pam_group.so” 行 が”/etc/pam.d/common-auth” に 書 き 加 え れ、”/etc/security/group.conf” に対応する設定がされていれば、実際のユーザーのグループメンバーシップは動的に割り当てられます。(第4章参照下さい。)

ハードウエアーデバイスは Debian システム上では一種のファイルでしかありません。CD-ROM や USB メモリースティックのようなデバイスをユーザーアカウントからアクセスするのに問題があった場合にはそのユーザーを該当するグループのメンバーにします。

いくつかのシステムが供給するグループはそのメンバーに root 権限無しに特定のファイルやデバイスにアクセスすることを可能にします。

グループ	アクセスできるファイルやデバイスの説明
dialout	シリアルポート (”/dev/ttyS[0-3]”) への全面的かつ直接のアクセス
dip	信頼できるピアーにダイヤルアップ **IP** 接続をするためのシリアルポートへの制限付きアクセス
cdrom	CD-ROM や DVD+/-RW のドライバー
audio	音声デバイス
video	映像デバイス
scanner	スキャナー
adm	システムモニターのログ
staff	下級管理業務のためのディレクトリー: ”/usr/local”、”/home”

Table 1.7: ファイルアクセスのためにシステムが供給する特記すべきグループのリスト

ティップ
モデムの設定をしたりどこにでも電話したり等するには dialout グループに所属する必要があります。もし信頼できるピアーに関する事前定義された設定ファイル”/etc/ppp/peers/” が root によって作成されていると、dip グループに属するだけで pppd(8) や pon(1) や poff(1) コマンドを用いてダイヤルアップ **IP** 接続が作成できます。

いくつかのシステムが供給するグループはそのメンバーに root 権限無しに特定のコマンドを実行することを可能にします。

グループ	実行可能なコマンド
sudo	パスワード無しに sudo を実行
lpadmin	プリンターのデーターベースからプリンターを追加・変更・削除するコマンドを実行

Table 1.8: 特定コマンド実行のためにシステムが供給する特記すべきグループのリスト

システムが供給するユーザーやグループの完全なリストは、base-passwd パッケージが供給する”/usr/share/doc/bas
の中にある最新バージョンの”Users and Groups” 文書を参照下さい。

ユーザーやグループシステムを管理するコマンドは passwd(5) や group(5) や shadow(5) や newgrp(1) や vipw(8) や vigr(8) や pam_group(8) を参照下さい。

1.2.6 タイムスタンプ

GNU/Linux ファイルのタイムスタンプには 3 種類あります。

タイプ	意味 **(**歴史的 **Unix** 定義**)**
mtime	ファイル内容変更時間 (ls -l)
ctime	ファイル状態変更時間 (ls -lc)
atime	ファイル最終アクセス時間 (ls -lu)

Table 1.9: タイムスタンプのタイプのリスト

注意
ctime はファイル作成日時ではありません。

注意
GNU/Linux システム上では、実際の **atime** 値は歴史的 Unix 定義とは異なる場合があります。

- ファイルが上書きされると、ファイルの **mtime** と **ctime** と **atime** の属性すべてが変更されます。
- ファイルの所有者やパーミッションの変更をすると、ファイルの **ctime** や **atime** アトリビュートを変えます。
- 伝統的 Unix システム上ではファイルを読むとファイルの **atime** 属性が変更されます。
- GNU/Linux システム上では、”`strictatime`” でファイルシステムをマウントした場合にファイルを読むとファイルの **atime** が変更されます。
- ファイルを初めて読み込んだときか、1 日空けてアクセスした場合、ファイルの **atime** 属性の更新が GNU/Linux (Linux 2.6.30 以降) の `relatime` でマウントされているファイルシステムでは生じます。
- **atime** 属性は `noatime` でマウントされているファイルシステムでは、読み込み時に更新されることはありません。

注意
”`noatime`” や”`relatime`” マウントオプションは通常使用状況下でのファイルシステムの読み出しパーフォーマンスを向上させるために導入されました。”`strictatime`” オプション下の単純なファイル読み出しオペレーションは **atime** 属性を更新する時間のかかる書き込み操作を引き起こします。しかし、**atime** 属性は mbox(5) ファイルを除くとほとんど使われることはありません。mount(8) を参照下さい。

既存ファイルのタイムスタンプを変更するには touch(1) コマンドを使って下さい。

タイムスタンプに関して、現代の英語ロケール (”`fr_FR.UTF-8`”) では旧来のロケール (”`C`”) と異なる文字列が `ls` コマンドから出力されます。

```
$ LANG=fr_FR.UTF-8  ls -l foo
-rw-rw-r-- 1 penguin penguin 0 oct. 16 21:35 foo
$ LANG=C  ls -l foo
-rw-rw-r-- 1 penguin penguin 0 Oct 16 21:35 foo
```

ティップ
”`ls -l`” の出力のカスタム化は項9.2.5を参照下さい。

1.2.7 リンク

”`foo`” というファイルを異なるファイル名”`bar`” に結びつけるのには 2 つの方法があります。

- ハードリンク
  - 既存ファイルの重複名
  - ”`ln foo bar`”
- シンボリックリンクもしくはシムリンク
  - 他のファイルをその名前で指す特殊ファイル
  - ”`ln -s foo bar`”

リンク数の変化と rm コマンドの結果の微妙な違いについての次の例をご覧下さい。

```
$ umask 002
$ echo ”Original Content” > foo
$ ls -li foo
1449840 -rw-rw-r-- 1 penguin penguin 17 Oct 16 21:42 foo
$ ln foo bar     # hard link
$ ln -s foo baz  # symlink
$ ls -li foo bar baz
1449840 -rw-rw-r-- 2 penguin penguin 17 Oct 16 21:42 bar
1450180 lrwxrwxrwx 1 penguin penguin  3 Oct 16 21:47 baz -> foo
1449840 -rw-rw-r-- 2 penguin penguin 17 Oct 16 21:42 foo
$ rm foo
$ echo ”New Content” > foo
$ ls -li foo bar baz
1449840 -rw-rw-r-- 1 penguin penguin 17 Oct 16 21:42 bar
1450180 lrwxrwxrwx 1 penguin penguin  3 Oct 16 21:47 baz -> foo
1450183 -rw-rw-r-- 1 penguin penguin 12 Oct 16 21:48 foo
$ cat bar
Original Content
$ cat baz
New Content
```

ハードリンクは同一ファイルシステム内に作れ、ls(1) コマンドに”-i” オプションを使って表示される inode 番号が同じです。

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パーミッション”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パーミッションが有効ファイルアクセスパーミッションとなります。

注意
もし特段の理由がないなら複雑なシンボリックリンクやハードリンクを作らない方が一般的には良いでしょう。シンボリックリンクの論理的組み合わせがファイルシステム中でループになっているという悪夢を引き起こすかもしれません。

注意
もしハードリンクを使う特段の理由がなければ、ハードリンクよりシンボリックリンクを使う方が一般的には良いでしょう。

”.” ディレクトリーは、それ自身が中にあるディレクトリーとリンクしていますので、新規ディレクトリーのリンク数は 2 から始まります。”..” ディレクトリーは親ディレクトリーとリンクしているので、ディレクトリーのリンク数は新規サブディレクトリーの増加に伴い増加します。

もし最近あなたが Windows から Linux に移動してきたなら、Unix のファイル名のリンクは Windows 上でもっとも似ている”shortcuts” との比較で如何にうまくできているかにすぐ気づくでしょう。ファイルシステム中に実装されているのでアプリケーションからはリンクされたファイルなのかオリジナルなのかの区別がつきません。ハードリンクの場合は実際全く違いはありません。

1.2.8 名前付きパイプ (FIFO)

名前付きパイプは、パイプのように働くファイルです。何かをファイルに入れると、もう一方の端からそれが出てきます。こうしてこれは FIFO または First-In-First-Out (先入れ先出し) と呼ばれます。つまり、最初にパイプに入れられたものが最初にもう一方の端から出てきます。

名前付きパイプに書き込む場合、パイプに書き込むプロセスは情報がパイプから読出されるまで終了しません。名前付きパイプから読み出す場合、読出すプロセス何か読出すものが無くな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ーを保存せず、シェルの”|” というシンタクッスが提供する機能ように 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なくても良いし、同じユーザーにより実行される必要さえありません。パイプは Unix の非常に影響力ある発明でした。

例えば、次を試してみて下さい:

```
$ cd; mkfifo mypipe
$ echo ”hello” >mypipe & # put into background
[1] 8022
$ ls -l mypipe
prw-rw-r-- 1 penguin penguin 0 Oct 16 21:49 mypipe
$ cat mypipe
hello
[1]+  Done                    echo ”hello” >mypipe
$ ls mypipe
mypipe
$ rm mypipe
```

1.2.9 ソケット

ソケットはインターネットのコミュニケーションやデーターベースやオペレーティングシステム自身によって頻繁に使われます。ソケットは名前つきパイプ (FIFO) に似ており、異なるコンピューター間でさえプロセス間の情報交換を可能にします。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も、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 (IPC) の終端点です。ネットワーク越しで異なるホストの間で情報の交換をすることも可能です。2 つの典型的なソケットは、インターネットソケットと Unix ドメインソケットです。

ティップ
”netstat -an” を実行すると特定のシステム上のソケットの全般状況がよく分かります。

1.2.10 デバイスファイル

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、ディスプレー、キーボードなどの物理デバイス又は仮想デバイス等を意味します。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。

2 タイプのデバイスファイルがあります。

- 文字デバイス
  - 1 文字毎にアクセス可能
  - 1 文字 = 1 バイト
  - 例: キーボードデバイス、シリアルポート等
- ブロックデバイス
  - 比較的大きなブロック単位でアクセス可能
  - 1 ブロック > 1 バイト
  - 例: ハードディスク等

デバイスファイルの読出し書込みが可能ですが、人間にとっては意味不明のバイナリーデーターがファイル中に多分含まれています。データー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” にテキストファイルをダンプしたり、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。くれぐれも気をつけて下さい。

注意
通常のプリンターへのアクセスは lp(1) を使います。

次のように ls(1) を実行するとデバイスノード番号が表示されます。

```
$ ls -l /dev/sda /dev/sr0 /dev/ttyS0 /dev/zero
brw-rw---T  1 root disk     8,  0 Oct 16 20:57 /dev/sda
brw-rw---T+ 1 root cdrom   11,  0 Oct 16 21:53 /dev/sr0
crw-rw---T  1 root dialout  4, 64 Oct 16 20:57 /dev/ttyS0
crw-rw-rw-  1 root root     1,  5 Oct 16 20:57 /dev/zero
```

- "/dev/sda" はメジャーデバイス番号 8 とマイナーデバイス番号 0 を持ちます。これは disk グループに所属するユーザーにより、読出し / 書込みアクセスが可能です。
- "/dev/sr0" はメジャーデバイス番号 11 とマイナーデバイス番号 0 を持ちます。これは cdrom グループに所属するユーザーにより、読出し / 書込みアクセスが可能です。
- "/dev/ttyS0" はメジャーデバイス番号 4 とマイナーデバイス番号 64 を持ちます。これは dialout グループに所属するユーザーにより、読出し / 書込みアクセスが可能です。
- "/dev/zero" はメジャーデバイス番号 1 とマイナーデバイス番号 5 を持ちます。これは誰によっても読出し / 書込みアクセスが可能です。

最近の Linux システムでは、"/dev/" の下のファイルは udev(7) メカニズムで自動的に充足されます。

1.2.11 特別なデバイスファイル

いくつかの特別なデバイスファイルがあります。

デバイスファイル	アクション	レスポンスの説明
/dev/null	読出し	" 行末 (EOF) 文字" を返す
/dev/null	書込み	何も返さず (底なしのデーターのゴミ捨て場)
/dev/zero	読出し	"\0 (NUL) 文字" を返す (ASCII の数字のゼロとは違う)
/dev/random	読出し	真の乱数発生機から真のエントロピーのあるランダムな文字を返す (遅い)
/dev/urandom	読出し	暗号学的にセキュアーな擬似乱数発生機からランダムな文字を返す
/dev/full	書込み	ディスクフル (ENOSPC) エラーを返す

Table 1.10: スペシャルなデバイスファイルのリスト

以上はシェルのリディレクションとともによく使われます。(項1.5.8参照下さい)。

1.2.12 procfs と sysfs

procfs と sysfs は"/proc" や"/sys" 上にマウントされる仮想ファイルシステムであり、カーネルの内部データー構造をユーザー空間にさらけ出します。言い換えると、オペレーティングシステムのオペレーションへの便利なのぞき窓となると言う意味で仮想といえます。

"/proc" ディレクトリー中には、システム上で実行されている各プロセスに対応したそのプロセス ID (PID) の名前がついたサブディレクトリー他があります。プロセス情報をアクセスする ps(1) のようなシステムユーティリティーはこのディレクトリー構造からその情報を得ています。

”/proc/sys/” の下のディレクトリーには実行時のカーネル変数を変更するインターフェースがあります。(専用の sysctl(8) コマンドもしくはその起動 / 設定ファイル”/etc/sysctl.conf” によっても同様のことができます。)

特にあるファイル - ”/proc/kcore” - に気づくと、パニックになる人がよくいます。これは一般に巨大です。これは (おおよそ) コンピューターのメモリー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。コンピューターのメモリーを指す仮想ファイルなので、そのサイズに関して心配する必要は全くありません。

”/sys” の下のディレクトリーはカーネルから引き出されたデーター構造、その属性、それらの関連を含んでいます。一部カーネル変数を実行時に変更する機構もまた含まれたりします。

linux-doc-* パッケージで供給される Linux カーネル文書 (”/usr/share/doc/linux-doc-2.6.*/Documentation/
中の”proc.txt(.gz)” や”sysfs.txt(.gz)” や関連する他の文書を参照下さい。

1.2.13 tmpfs

tmpfs は 仮想記憶 中にすべてのファイルを保持する一時的なファイルシステムです。メモリー上のページキャッシュ中にある tmpfs のデーターは必要に応じてディスク上の swap 空間 へと書き出せます。

”/run” ディレクトリは初期ブートプロセスに tmpfs としてマウントされます。こうすることで”/” が読み取り専用でマウントされていてもそこへの書き込みが可能です。これは過渡的な状態ファイルの保管のための新たな場所で、Filesystem Hierarchy Standard のバージョン 2.3 に規定されたいくつかの場所を置き換えます:

- ”/var/run” → ”/run”
- ”/var/lock” → ”/run/lock”
- ”/dev/shm” → ”/run/shm”

linux-doc-* パッケージで供給される Linux カーネル文書 (”/usr/share/doc/linux-doc-*/Documentation/file
中の”tmpfs.txt(.gz)” を参照下さい。

1.3 ミッドナイトコマンダー (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。標準 Unix コマンドを習うよりもより簡単なメニューを使ったコンソール経験が初心者にもできます。

”mc” と名づけられた Midnight Commander パッケージを次のようにしてインストールする必要があります。

```
$ sudo apt-get install mc
```

Debian システムを探検するために mc(1) コマンドを使います。これは学習するための最良の方法です。カーソールキーとエンターキーを使うだけで興味深い場所をちょっと探検します。

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

1.3.1 MC のカスタム化

終了時に作業ディレクトリーを MC に変更させそのディレクトリーへ cd させるためには、mc パッケージが提供するスクリプトを”~/.bashrc” が含むように変更します。

```
. /usr/lib/mc/mc.sh
```

この理由は mc(1) (”-P” オプション項目) を参照下さい (今言っていることがよく分からないなら、これは後日しても大丈夫です。)

1.3.2 MC の始動

MC は次のようにして起動します。

```
$ mc
```

MC を使うとメニューを通じた最小限のユーザーの努力で全てのファイル操作の面倒が見られます。ヘルプ表示を出すには、ただ F1 を押すだけです。カーソールキーとファンクションキーの操作だけで MC を使えます。

注意
gnome-terminal(1) のようなコンソールでは、ファンクションキーのキーストロークがコンソールプログラムに横取りされる事があります。`gnome-terminal` の場合、”Edit” → ”Keyboard Shortcuts” とするとこの機能を無効にできます。

もし文字化け表示がされる文字符号化 (エンコーディング) 問題に出会った際には、MC のコマンドラインに”`-a`” を加えると解消する事があります。

これでも MC の表示の問題が解消しない際には、項9.4.6を参照下さい。

1.3.3 MC のファイルマネージャー

2 つのディレクトリーパネルがありそれぞれファイルリストを含むのが標準です。他の便利なモードとしては、右側のウィンドウを”information” とセットしてファイルアクセス権情報などを表示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(8) デーモンを実行すると、Linux の文字ターミナルでマウスも使えます。(MC で通常の挙動のカットアンドペーストをさせるには、shift キーを押して下さい。)

キー	キーバインディング
`F1`	ヘルプメニュー
`F3`	内部ファイルビューワー
`F4`	内部エディター
`F9`	プルダウンメニュー有効
`F10`	MC を終了
`Tab`	二つのウィンドウの間を移動
`Insert` もしくは `Ctrl-T`	コピーのような複数ファイル操作のためにファイルをマーク
`Del`	ファイルの削除 (気をつけましょう -- MC を安全削除モードに設定)
カーソールキー	自明

Table 1.11: MC のキーバインディング

1.3.4 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ーを変更します。
- `Ctrl-Enter` と `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(1) や mv(1) コマンドで御使用下さい。
- `Alt-Tab` はシェルファイル名の自動展開の選択肢を表示します。
- MC の引数で両ウィンドウのスタートディレクトリーを指定できます。例えば”`mc /etc /root`”。
- `Esc` + `n-key` → `Fn` (つまり、`Esc` + `1` → `F1`、等々、`Esc` + `0` → `F10`)
- `Esc` をキーの前に押すのは `Alt` をキーと同時に押すのと同様の効果があります。つまり、`Esc` + `c` は `Alt-C` と同じです。`Esc` はメタキーとよばれ時々”`M-`” と表記されます。

1.3.5 MC の内部エディター

MC の内部エディター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、選択範囲のスタートとマークし、次に F3 を押すと、選択範囲のエンドとマークし、選択範囲を強調します。そしてカーソールを動かすことができます。F6 を押すと、選択範囲はカーソー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、カーソー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。ほとんどのカーソールキーは直感的に働きます。

このエディターは次のコマンドの内のひとつを使いファイルに対し直接起動できます。

```
$ mc -e filename_to_edit
```

```
$ mcedit filename_to_edit
```

これはマルチモードエディターではありませんが、複数の Linux コンソール上で使用すると同じ効果を発揮させされます。ウィンドウ間のコピーを行うには、Alt-<n> キーを押して仮想コンソールを切替えて、”File→Insert file” や”File→Copy to file” を用いてファイルの一部を他のファイルに動かします。

この内部エディターはお好きな他の外部エディター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ーを決定するために環境変数”$EDITOR” や”$VISUAL” を使用します。最初 vim(1) や nano(1) が使いにくい場合には”~/.bashrc” に次に示す行を追加してエディターを”mcedit” に設定するのも一計です。

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

できればこれは”vim” に設定することを推奨します。

vim(1) が使いにくい場合には、mcedit(1) をほとんどのシステム管理業務のために使い続けられます。

1.3.6 MC の内部ビューワー

MC は非常に賢明なビューワーです。文書内の単語を検索するための素晴らしいツールです。私は”/usr/share/doc” ディレクトリー内のファイルに対していつもこれを使います。これは大量にある Linux 情報を閲覧する最速の方法です。このビューワーは次のコマンドの内のひとつを使い直接起動できます。

```
$ mc -v path/to/filename_to_view
```

```
$ mcview path/to/filename_to_view
```

1.3.7 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内容を処理します (項9.3.11参照下さい)。これは非常に便利な MC の機能です。

ファイルタイプ	**enter** キーへの反応
実行ファイル	コマンド実行
man ファイル	ビューワーソフトに内容をパイプ
html ファイル	ウェッブブラウザーに内容をパイプ
”*.tar.gz” や”*.deb” ファイル	サブディレクトリーであるかのように内容を表示

Table 1.12: enter キー入力への MC の反応

これらのビューワーや仮想ファイルの機能を有効にするためには、閲覧可能なファイルには実行可能と設定されていてはいけません。chmod(1) コマンドを使うか、MC のファイルメニュー経由で状態を変更して下さい。

1.3.8 MC の FTP 仮想ファイルシステム

MC を Internet 越しでの FTP を用いたファイルアクセスに使えます。F9 を押してメニューに行き、”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、あたかもローカルにあるかのようにリモートディレクトリーを取得します。

”[deb.debian.org/debian]” を URL として Debian アーカイブを閲覧します。

1.4 基本の Unix 的作業環境

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 的な作業環境に親しむのは非常に重要なことです。

1.4.1 login シェル

ログインシェルは chsh(1) を使えば選択できます。

パッケージ	ポプコン	サイズ	POSIX シェル	説明
bash	V:791, I:999	6469	はい	Bash: GNU Bourne Again SHell (デファクトスタンダード)
tcsh	V:9, I:29	1316	いいえ	TENEX C Shell: 拡張バージョンの Berkeley csh
dash	V:907, I:992	221	はい	Debian の Almquist シェル、シェルスクリプトに好適
zsh	V:37, I:73	2442	はい	Z shell: 多くの拡張された標準シェル
mksh	V:4, I:12	1469	はい	Korn シェル の 1 バージョン
csh	V:2, I:8	343	いいえ	OpenBSD の C シェル、Berkeley csh の派生
sash	V:0, I:6	1054	はい	組み込みコマンド付きの独立シェル (標準の”/bin/sh” には不向き)
ksh	V:3, I:16	3284	はい	真の AT&T バージョンの Korn シェル
rc	V:0, I:2	169	いいえ	AT&T Plan 9 の rc シェルの実装
posh	V:0, I:0	190	はい	ポリシー準拠の通常シェル (pdksh の派生)

Table 1.13: シェルプログラムのリスト

ティップ
POSIX-ライクなシェルは基本シンタックスはにていますが、シェル変数や glob の展開のような基本事項の挙動が異なることがあります。詳細に関しては個々の文書を確認してください。

このチュートリアル章内では、インタラクティブなシェルは常に bash です。

1.4.2 Bash のカスタム化

vim(1) の挙動は”~/.vimrc” を使ってカスタム化できます。

例えば、次を試してみて下さい。

```
# enable bash-completion
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
```

```
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

# CD upon exiting MC
. /usr/lib/mc/mc.sh

# set CDPATH to a good one
CDPATH=.:/usr/share/doc:~:~/Desktop:~
export CDPATH

PATH="${PATH+$PATH:}/usr/sbin:/sbin"
# set PATH so it includes user's private bin if it exists
if [ -d ~/bin ] ; then
  PATH="~/bin${PATH+:$PATH}"
fi
export PATH

EDITOR=vim
export EDITOR
```

ティップ
bash に関する更なるカスタム化方法は、第9章中の項9.2.7等にあります。

ティップ
bash-completion パッケージは bash で入力のプログラムによる補完を可能にします。

1.4.3 特別のキーストローク

Unix 的環境下では、特別の意味を持ったいくつかのキーストロークがあります。通常の Linux の文字ターミナルでは左側の Ctrl や Alt キーのみが期待にそって機能することに配慮下さい。次に特記すべき暗記するべきキーストロークを記します。

キー	キーバインディングの説明
Ctrl-U	カーソールの前の 1 行を消去
Ctrl-H	カーソールの前の 1 文字を削除
Ctrl-D	入力を終了 (シェルを使用中の場合、シェルを終了)
Ctrl-C	実行中のプログラムを終了
Ctrl-Z	プログラムをバックグラウンドジョブに移動し一時停止
Ctrl-S	スクリーンへの出力を停止
Ctrl-Q	スクリーンへの出力を再開
Ctrl-Alt-Del	システムをリブート / 停止、inittab(5) 参照下さい
Left-Alt キー (もしくは、Windows キー)	Emacs および同様の UI でのメタキー
Up-arrow	bash でコマンド履歴検索をスタート
Ctrl-R	bash でインクリメンタルなコマンド履歴検索をスタート
Tab	bash のコマンドラインのファイル名入力を完結
Ctrl-V Tab	bash のコマンドラインで Tab を展開することなく入力

Table 1.14: Bash のキーバインディングのリスト

ティップ
ターミナルの `Ctrl-S` 機能は stty(1) で無効にできます。

1.4.4 Unix 流のマウス操作

Unix 流のマウス操作は 3 ボタンマウスが基本です。

アクション	反応
マウスの左クリックアンドドラッグ	選択とクリップボードへのコピー
左クリック	選択スタート点の選択
右クリック	選択エンド点の選択とクリップボードへのコピー
中クリック	クリップボードをカーソール位置に挿入 (ペースト)

Table 1.15: Unix 流のマウス操作

現代的なホイールマウスの真ん中のホイールは中マウスボタンと見なされ、中クリックに使えます。2 ボタンマウス状況では左右のボタンの同時押しが中クリックとして使えます。Linux の文字コンソールでマウスを使うには gpm(8) をデーモンで実行する必要があります。

1.4.5 ページャー

less(1) は機能拡張されたページャー (ファイル内容のブラウザー) です。コマンドアーギュメントに指定されたファイル、もしくは標準入力を読みます `less`。コマンドで閲覧中にヘルプが必要なら、”h” を押すしましょう。これは、more(1) よりもはるかに高機能で、”`eval $(lesspipe)`” または”`eval $(lessfile)`” をシェルのスタートスクリプト中で実行することで更に機能が拡充されます。詳しくは、”`/usr/share/doc/less/LESSOPEN`” を参照下さい。”`-R`” オプションを用いると、生の文字出力や ANSI カラーエスケープシーケンスが有効になります。less(1) を参照下さい。

1.4.6 テキストエディター

Unix 的システムで人気のある、Vim か Emacs プログラムのいずれかのバリアントに習熟するべきです。

著者としては Vim コマンドに慣れることは正しいことだと考えています。なぜなら Vi エディターは Linux/Unix の世界では必ず存在するからです。(実際はオリジナルの `vi` か、新しい `nvi` がどこででも見つけられるプログラムです。これにもかかわらず Vim を著者が初心者のために選んだのは、より強力かつ動作が充分似ているのと、`F1` キーを通じてヘルプが表示されるからです。)

これとは違い、Emacs か XEmacs をエディターとして選ぶのも、特にプログラムをするには、非常に良い選択です。Emacs には、ニュースリーダ機能、ディレクトリーの編集機能、メール機能他の、過多な機能があります。プログラミングやシェルスクリプトの編集に使うときは、作業中のフォーマットをインテリジェントに認識し助力をしようとします。Linux 上で必要なプログラムは Emacs だけと考える人もいます。Emacs を今 10 分間学ぶことは将来何時間もの節約になります。Emacs を学ぶ際には GNU の Emacs マニュアルを持っておくことを高く推薦します。

これら全てのプログラムには練習しながら学べるようにチュータリングプログラムが普通付いてきます。Vim を”`vim`” とタイプして起動し、F1 キーを押します。最初の 35 行を読みます。カーソールを”`|tutor|`” に移動し `Ctrl-]` を押してオンラインの訓練コースを始めます。

注意
Vim や Emacs のような良いエディターは、UTF-8 や他のエギゾチックな符号化方式 (エンコーディング) のテキストを正しく扱えます。それには UTF-8 ロケール中の X 環境で、必要なプログラムとフォントをインストールをするのが賢明です。マルチバイトテキストに関するそれぞれの文書を参照下さい。

1.4.7 デフォールトのテキストエディターの設定

Debian にはいくつかの異なったエディターがあります。上述のように vim パッケージをインストールすることを推薦します。

Debian ではシステムのデフォールトのエディターへの統一されたアクセスを”/usr/bin/editor” コマンドを通じて提供しているので、他のプログラム (例えば reportbug(1) 等) が起動できます。設定変更は次で出来ます。

```
$ sudo update-alternatives --config editor
```

著者が”/usr/bin/vim.tiny” より”/usr/bin/vim.basic” を初心者に推薦するのはシンタクスハイライトをサポートしているからです。

ティップ
多くのプログラムは”$EDITOR” か”$VISUAL” という環境変数を使ってどのエディターを使うかを決めます (項1.3.5と項9.3.11参照下さい)。Debian システムの整合性のために、これらを”/usr/bin/editor” と設定しましょう。(歴史的には”$EDITOR” は”ed” で、”$VISUAL” は”vi” でした。)

1.4.8 Vim のカスタム化

vim(1) の挙動は”~/.vimrc” を使ってカスタム化できます。

例えば、次を試してみて下さい:

```
" -----------------------------
" Local configuration
"
set nocompatible
set nopaste
set pastetoggle=<f2>
syn on
if $USER == "root"
 set nomodeline
 set noswapfile
else
 set modeline
 set swapfile
endif
" filler to avoid the line above being recognized as a modeline
" filler
" filler
```

1.4.9 シェル活動の記録

シェルコマンドの出力はスクリーンから押し出されると永久に無くなってしまうかもしれません。シェルでの活動を後で見直せるようにファイルに記録しておくのは良いことです。この種の記録は何らかのシステム管理作業をする際には非常に重要です。

シェル活動の記録の基本方法は script(1) の下で実行することです。

例えば、次を試してみて下さい:

```
$ script
Script started, file is typescript
```

script の下で何なりのシェルコマンドを実行します。

Ctrl-D を押して script から脱出します。

```
$ vim typescript
```

項9.2.3を参照下さい。

1.4.10 基本 Unix コマンド

基本的 Unix コマンドを学びます。ここでは一般的意味で”Unix” を使っています。いかなる Unix クローンの OS も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。今一部コマンドが思うように機能しなくても心配しないで下さい。エリアスがシェルで使われた場合は、対応するコマンドの出力は変わります。次は順番に実行すると言う意味の例ではありません。

非特権ユーザーのアカウントから次のコマンドを全て実行します。

注意
Unix は”.” で始まるファイル名を隠す伝統があります。それらは伝統的には特定の設定情報やユーザーの嗜好を含むファイルです。

注意
cd コマンドに関しては builtins(7) を参照下さい。

注意
最小限の Debian システムのデフォールトのページャーは more(1) で、スクロールバックができません。less パッケージを”apt-get install less” と言うコマンドラインでインストールすると、less(1) がデフォールトのページャーになりカーソールキーでスクロールバック出来るようになります。

注意
上記の”ps aux | grep -e ”[e]xim4*”” コマンド中に現れる正規表現中の”[” 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グロブでも正規表現ででも使われますが、これらの意味は異なります。grep(1) から正規表現を学びましょう。

上記のコマンドを訓練として用いて、ディレクトリーを渡り歩き、システムの中を覗き込んで下さい。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下さい。

例えば、次を試してみて下さい:

```
$ man man
$ man bash
$ man builtins
$ man grep
$ man ls
```

マンページのスタイルは慣れるのに少々大変かもしれません。なぜなら特に比較的旧式の非常に伝統的なマンページは比較的言葉が少ないからです。しかし一旦慣れるとその簡潔さの良さが分かります。

GNU や BSD 由来を含む多くの Unix 的なコマンドは次のように (場合によっては一切の引数無しで) 起動すると簡単なヘルプ情報を表示します。

```
$ <b'' コ b''b'' マ b''b'' ン b''b'' ド b''b'' 名 b''> --help
$ <b'' コ b''b'' マ b''b'' ン b''b'' ド b''b'' 名 b''> -h
```

コマンド	説明
pwd	カレント / ワーキングディレクトリーの名前を表示
whoami	現在のユーザー名を表示
id	現在のユーザーのアイデンティティ (名前と uid と gid と関連する group) を表示
file <foo>	"<foo>" ファイルのファイルタイプを表示
type -p <commandname>	"<commandname>" コマンドのファイルの位置を表示
which <commandname>	, ,
type <commandname>	"<commandname>" コマンドに関する情報を表示
apropos <key-word>	"<key-word>" に関連したコマンドを発見
man -k <key-word>	, ,
whatis <commandname>	"<commandname>" コマンドに関する 1 行の説明を表示
man -a <commandname>	"<commandname>" コマンドに関する説明を表示 (Unix スタイル)
info <commandname>	"<commandname>" コマンドに関する比較的長い説明を表示 (GNU スタイル)
ls	ディレクトリーの内容をリスト (非ドットファイルおよびディレクトリー)
ls -a	ディレクトリーの内容をリスト (全ファイルおよびディレクトリー)
ls -A	ディレクトリーの内容をリスト (ほとんど全ファイルおよびディレクトリー、".." と"." をスキップ)
ls -la	ディレクトリーの内容を詳細情報とともにリスト
ls -lai	ディレクトリーの内容を inode 番号と詳細情報とともにリスト
ls -d	現ディレクトリーの中の全ディレクトリーをリスト
tree	ファイルツリーの内容を表示
lsof <foo>	"<foo>" ファイルのオープンの状態をリスト
lsof -p <pid>	プロセス ID: "<pid>" によってオープンされたファイルをリスト
mkdir <foo>	現ディレクトリー中に"<foo>" という新規ディレクトリー作成
rmdir <foo>	現ディレクトリー中の"<foo>" というディレクトリーを削除
cd <foo>	現ディレクトリー中もしくは"$CDPATH" 変数中にリストされたディレクトリー中の"<foo>" というディレクトリーにディレクトリーを変更
cd /	ディレクトリーをルートディレクトリーに変更
cd	現在のユーザーのホームディレクトリーにディレクトリーを変更
cd /<foo>	絶対ディレクトリーパス"/<foo>" にディレクトリーを変更
cd ..	親ディレクトリーにディレクトリーを変更
cd ~<foo>	ユーザー"<foo>" のホームディレクトリーにディレクトリーを変更
cd -	一つ前のディレクトリーにディレクトリーを変更
</etc/motd pager	"/etc/motd" の内容をデフォールトのページャーで表示
touch <junkfile>	空ファイル"<junkfile>" 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
rm <junkfile>	ファイル"<junkfile>" を削除
mv <foo> <bar>	既存のファイル"<foo>" の名前を新しい名前"<bar>" に変更 (ディレクトリー"<bar>" が存在不可)
mv <foo> <bar>	既存のファイル"<foo>" を新しい場所"<bar>/<foo>" に移動 (ディレクトリー"<bar>" が存在しなければいけない)
mv <foo> <bar>/<baz>	既存のファイル"<foo>" を新しい場所の新しい名前のファイル"<bar>/<baz>" に移動 (ディレクトリー"<bar>" が存在しなければいけないが、ディレクトリー"<bar>/<baz>" は存在してはいけない)
chmod 600 <foo>	既存のファイル"<foo>" を他人から読出し不可かつ書込み不可 (全員実行不可)
chmod 644 <foo>	既存のファイル"<foo>" を他人からは読出し可だが書込み不可 (全員実行不可)
chmod 755 <foo>	既存のファイル"<foo>" を他人からは読出し可だが書込み不可 (全員実行可能)
find . -name <pattern>	シェルで"<pattern>" にマッチするファイル名を探索 (比較的遅い)
locate -d . <pattern>	シェルで"<pattern>" にマッチするファイル名を探索 (定期的に生成されるデーターベースを使い比較的早い)

1.5 シェルプロンプト

Debian システムの使い方が少し分かったでしょう。Debian システム上でのコマンド実行のメカニズムを掘り下げます。初心者のためにちょっと簡略化してみました。正確な説明は bash(1) を参照下さい。

シンプルなコマンドは、次の要素のシーケンスとなります。

1. 変数代入 (任意)
2. コマンド名
3. 引数 (任意)
4. リダイレクト (任意: > と >> と < と << 等。)
5. 制御演算子 (任意: && と || と < 改行 > と ; と & と (と))

1.5.1 コマンド実行と環境変数

環境変数の値は Unix コマンドの挙動を変えます。

環境変数のデフォールト値は PAM システムが初期設定されます。その後次のような何らかのアプリケーションプログラムにより再設定されているかもしれません。

- gdm3 のようなディスプレーマネージャーは環境変数を再設定します。
- "~/.bash_profile" や"~/.bashrc" にあるシェル起動コードの中でシェルは環境変数を再設定します。

1.5.2 "$LANG" 変数

"$LANG" 変数に与えられる完全なロケール値は 3 つの部分からなります: "xx_YY.ZZZZ"。

ロケールの値	意味
xx	ISO 639 言語コード (小文字)、例えば"en"
YY	ISO 3166 国コード (大文字)、例えば"US"
ZZZZ	コードセット、常に"UTF-8" と設定

Table 1.17: ロケールの値の 3 つの部分

言語コードと国コードは"info gettext" 中の該当記述を参照下さい。

現代的な Debian システム上では、十分な理由と必要な知見をもって歴史的なコードセットを特段希望しない限り、常にコードセットを **UTF-8** と設定すべきです。

ロケールの詳細に関しては、項8.4を参照下さい。

注意
On Debian system, make sure to install the locales-all package to use all locales.

注意
"LANG=en_US" は、"LANG=C" でも、"LANG=en_US.UTF-8" でもありません。それは"LANG=en_US.ISO-8859-1" です (項8.4.1参照下さい)。

推奨ロケール	言語 (地域)
en_US.UTF-8	英語 (米国)
en_GB.UTF-8	英語 (英国)
fr_FR.UTF-8	フランス語 (フランス)
de_DE.UTF-8	ドイツ語 (ドイツ)
it_IT.UTF-8	イタリア語 (イタリア)
es_ES.UTF-8	スペイン語 (スペイン)
ca_ES.UTF-8	カタラン語 (スペイン)
sv_SE.UTF-8	スウェーデン語 (スウェーデン)
pt_BR.UTF-8	ポルトガル語 (ブラジル)
ru_RU.UTF-8	ロシア語 (ロシア)
zh_CN.UTF-8	中国語 (中華人民共和国)
zh_TW.UTF-8	中国語 (台湾 R.O.C.)
ja_JP.UTF-8	日本語 (日本)
ko_KR.UTF-8	韓国語 (大韓民国)
vi_VN.UTF-8	ベトナム語 (ベトナム)

Table 1.18: 推奨ロケールのリスト

典型的なコマンドの実行は次のようなシェルの行シーケンスを用います。

```
$ date
Sun Jun  3 10:27:39 JST 2007
$ LANG=fr_FR.UTF-8 date
dimanche 3 juin 2007, 10:27:33 (UTC+0900)
```

以上で、date(1) プログラムは異なる環境変数"$LANG" 値で実行されます。

- 最初のコマンドでは、"$LANG" はシステムでフォルトのロケール値"en_US.UTF-8" に設定されます。
- 二番目のコマンドでは、"$LANG" はフランス語の UTF-8 ロケール値"fr_FR.UTF-8" に設定されます。

ほとんどのコマンド実行は頭に環境変数定義をつけないのが普通です。上記の例の代わりに次のように実行します。

```
$ LANG=fr_FR.UTF-8
$ date
dimanche 3 juin 2007, 10:27:33 (UTC+0900)
```

ここで確認できるように、コマンドの出力は環境変数に影響されフランス語の出力となっています。もし環境変数を (例えばシェルスクリプトを呼んでいて) サブプロセスに引き継ぎたい際には、次のように環境変数を **export** (エクスポート) しなければいけません。

```
$ export LANG
```

注意
典型的なコンソールターミナルを用いる際には、"$LANG" 環境変数は通常デスクトップ環境によって **export** されるように設定されています。上記例は export の効果を検証するあまりいい例ではありません。

ティップ
バグ報告をする際には、非英語環境を使っているなら、プログラムを"LANG=en_US.UTF-8" の下で実行し確認することが望ましいです。

"$LANG" とこれに関連した環境変数に関しては、locale(5) と locale(7) を参照下さい。

注意
特段必要がなければ”$LC_*” 変数を避けて、”$LANG” 変数のみを用いてシステム環境設定する事をお薦めします。

1.5.3 ”$PATH” 変数

シェルにコマンドを打ち込んだ際に、シェルは”$PATH” 環境変数にリストされたディレクトリーのリストから検索します。”$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、ユーザーアカウントの”$PATH” 環境変数には”/sbin” や”/usr/sbin” が含まれないかもしれません。例えば、ifconfig コマンドは”/sbin/ifconfig” とフルパスを使って実行する必要があります。(類似の ip コマンドは”/bin” にあります。)

Bash シェルの”$PATH” 環境変数は、”~/.bash_profile” か”~/.bashrc” ファイルで変更できます。

1.5.4 ”$HOME” 変数

多くのコマンドはユーザー特定の設定をホームディレクトリーに保存し、その内容でコマンドの挙動が変わります。ホームディレクトリーは”$HOME” 環境変数で指定されます。

”$HOME” の値	プログラム実行状況
/	init プロセスが実行するプログラム (デーモン)
/root	通常の root シェルから実行されるプログラム
/home/<normal_user>	通常のユーザーシェルから実行されるプログラム
/home/<normal_user>	通常のユーザーの GUI デスクトップメニューから実行されるプログラム
/home/<normal_user>	”sudo program” を用いて root として実行されるプログラム
/root	”sudo -H program” を用いて root として実行されるプログラム

Table 1.19: ”$HOME” の値のリスト

ティップ
シェルは、”~/” を現ユーザーのホームディレクトリーである”$HOME/” へと展開します。シェルは、”~foo/” をユーザー foo のホームディレクトリーである”/home/foo/” へと展開します。

1.5.5 コマンドラインオプション

プログラムコマンドによっては引数があります。引数は”-” か”--” で始まり、オプションと呼ばれ、コマンドの挙動をコントロールします。

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

上記で、コマンドライン引数”-R” が date(1) の挙動を RFC2822 準拠の日付文字列出力と変えています。

シェルグロブパターン	マッチルールの説明
*	”.” で始まらないファイル (部分) 名
.*	”.” で始まるファイル (部分) 名
?	1 文字
[…]	括弧中の 1 文字
[a-z]	”a” と”z” の範囲間の 1 文字
[^…]	括弧内 (”^” 以外) に含まれる文字以外の 1 文字

Table 1.20: シェルグロブパターン

1.5.6 シェルグロブ

ファイル名を全てタイプせずにファイルのグループをコマンド処理したいことがよくあります。シェルのグロブ (ワイルドカードとも時々呼ばれる) を用いたファイル名のパターン展開を用いるとこのニーズに答えられます。

例えば、次を試してみて下さい:

```
$ mkdir junk; cd junk; touch 1.txt 2.txt 3.c 4.h .5.txt ..6.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 ..6.txt
$ echo .*[^.]*
.5.txt ..6.txt
$ echo [^1-3]*
4.h
$ cd ..; rm -rf junk
```

glob(7) を参照下さい。

注意
通常のシェルのファイル名の展開と違い、find(1) が”-name” テスト他でシェルパターン”*” をテストする際にはファイル名先頭の”.” ともマッチします。(新 POSIX 機能)

注意
BASH は shopt 組み込みオプションで”dotglob” や”noglob” や”nocaseglob” や”nullglob” や”extglob” などとすることでグロブ挙動を色々変更できます。bash(1) を参照下さい。

1.5.7 コマンドの戻り値

各コマンドは終了ステータスを戻り値 (変数: ”$?”) として返します。

コマンドの終了状態	戻り値の数値	戻り値の論理値
成功	ゼロ、0	真
失敗	非ゼロ、-1	偽

Table 1.21: コマンドの終了コード

例えば、次を試してみて下さい。

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

注意
シェルの論理的な観点では、成功は、0 (ゼロ) の値を持つ論理的真として扱われることに注意して下さい。少々これは非直感的なのでここで再確認する必要があります。

1.5.8 典型的なコマンドシーケンスとシェルリディレクション

次に挙げるシェルコマンドの一部として一行でタイプするシェルコマンドの慣用句を覚えましょう。

コマンドの慣用句	説明
`command &`	command をサブシェル中でバックグラウンド実行
`command1 | command2`	command1 の標準出力を command2 の標準入力に パイプ (同時並行で実行)
`command1 2>&1 | command2`	command1 の標準出力と標準エラー出力を command2 の標準入力にパイプ (同時進行で実行)
`command1 ; command2`	command1 を実行し、後に続いて command2 を実行
`command1 && command2`	command1 を実行; もし成功したら、後に続いて command2 を実行 (command1 と command2 の両方が成功したら、正常終了を返す)
`command1 || command2`	command1 を実行; もし成功しなかったら、後に続いて command2 を実行 (command1 か command2 のどちらかが成功したら、正常終了を返す)
`command > foo`	command の標準出力を foo ファイルにリダイレクト (上書き)
`command 2> foo`	command の標準エラー出力を foo ファイルにリダイレクト (上書き)
`command >> foo`	command の標準出力を foo ファイルにリダイレクト (追記)
`command 2>> foo`	command の標準エラー出力を foo ファイルにリダイレクト (追記)
`command > foo 2>&1`	command の標準出力と標準エラー出力を foo ファイルにリダイレクト
`command < foo`	command の標準入力を foo ファイルからリダイレクト
`command << delimiter`	command の標準入力を”delimiter” に出会うまでのこれに続く行からリダイレクト (ヒアドキュメント)
`command <<- delimiter`	command の標準入力を”delimiter” に出会うまでのこれに続く行からリダイレクト (ヒアドキュメント、行頭のタブ文字は入力から削除)

Table 1.22: シェルコマンドの慣用句

Debian システムはマルチタスクシステムです。バックグラウンドジョブを使うと単一シェルの下で複数プログラムを実行可能にします。バックグラウンドジョブの管理にはシェル内部組み込みコマンドの `jobs` や `fg` や `bg` や `kill` を使います。bash(1) マンページ中の”SIGNALS” と”JOB CONTROL” セクションや builtins(1) を参照下さい。

例えば、次を試してみて下さい:

```
$ </etc/motd pager
```

```
$ pager </etc/motd
```

```
$ pager /etc/motd
```

```
$ cat /etc/motd | pager
```

4 つ全ての例が全く同じ表示をしますが、最後の例は余計な cat コマンドを実行するので理由なくリソースの無駄遣いをします。

シェルでは exec 組み込みコマンドを任意のファイルディスクリプタとともに使いファイルをオープンすることができます。

```
$ echo Hello >foo
$ exec 3<foo 4>bar  # open files
$ cat <&3 >&4       # redirect stdin to 3, stdout to 4
$ exec 3<&- 4>&-    # close files
$ cat bar
Hello
```

ファイルデスクリプタの 0-2 は事前定義されています。

デバイス	説明	ファイルデスクリプタ
stdin	標準出力	0
stdout	標準出力	1
stderr	標準エラー出力	2

Table 1.23: 事前定義されたファイルデスクリプタ

1.5.9 コマンドエリアス

良く使うコマンドにエリアスを設定できます。

例えば、次を試してみて下さい:

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

こうすると、"la" が"ls -la" の短縮形として機能し、全てのファイルを長いリスト形式でリストします。

既存のエリアスは alias でリストできます (bash(1) の"SHELL BUILTIN COMMANDS" 参照下さい)。

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

type 内部コマンドを使うと正確なパスやコマンドの正体を識別できます (bash(1) の"SHELL BUILTIN COMMANDS" 下参照下さい)。

例えば、次を試してみて下さい:

```
$ 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
```

上記で、ls は最近探索されましたが"file" は最近探索されていませので、"ls" は" ハッシュされた" つまりシェルには"ls" コマンドの場所を高速アクセスのために内部記録していると表示されます。

ティップ
項9.2.7を参照下さい。

1.6 Unix 的テキスト処理

Unix 的作業環境では、テキスト処理はテキストを標準テキスト処理ツールの連鎖パイプを通す行います。これは決定的な Unix の発明です。

1.6.1 Unix テキストツール

Unix 的システムでしばしば使われる標準テキスト処理ツールがいくつかあります。

- 正規表現無使用:
  - cat(1) はファイルをつなぎ合わせ全てを出力します。
  - tac(1) はファイルをつなぎ合わせ逆順で出力します。
  - cut(1) は行の一部を選択し出力します。
  - head(1) はファイルの最初の部分を選択し出力します。
  - tail(1) はファイルの最後の部分を選択し出力します。
  - sort(1) は行を順番に並び替えます。
  - uniq(1) は順番に並べられたファイルから重複行を削除します。
  - tr(1) は文字を変換削除します。
  - diff(1) は 1 行ごとにファイルを比較します。
- 基本正規表現 (**BRE**) 使用:
  - egrep(1) はテキストのパターンマッチをします。
  - ed(1) は原始的な行エディター。
  - sed(1) はストリームエディター。
  - vim(1) はスクリーンエディター。
  - emacs(1) はスクリーンエディター。(ちょっと拡張された **BRE**)
- 拡張正規表現 (**ERE**) 使用:
  - egrep(1) はテキストのパターンマッチをします。
  - awk(1) は単純なテキスト処理をします。
  - tcl(3tcl) は考え得る全てのテキスト処理をします: re_syntax(3)。時々 tk(3tk) とともに使用されます。
  - perl(1) は考え得る全てのテキスト処理をします。perlre(1).
  - `pcregrep` パッケージの pcregrep(1) はテキストのパターンマッチを Perl 互換正規表現 (PCRE) パターンを使ってします。
  - `re` モジュールとともに使うことで python(1) は考え得る全てのテキスト処理をします。”`/usr/share/doc/python/h`
    を参照下さい。

もしこれらのコマンドが正確にどう動作するかを確認したいなら、”`man command`” を使って自分で見つけましょう。

注意
ソート順や範囲表現はロケールに依存します。コマンドの伝統的挙動を得たい際には、”LANG=C” をコマンドの前に付けて **UTF-8** ロケールではなく **C** ロケールでコマンドを使います。(項1.5.2と項8.4を参照下さい)。

注意
Perl 正規表現 (perlre(1)) と Perl 互換正規表現 (PCRE) と `re` モジュールで提供される Python 正規表現は **ERE** に多くの共通の拡張をしています。

1.6.2 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。シェルグロブに類似していますがより複雑で強力です。

正規表現はマッチするパターンを表現し、テキスト文字とメタ文字からなっています。

メタ文字は特別な意味を持った文字です。上記のようにテキストツールによって、**BRE** と **ERE** の 2 つの主要なスタイルがあります。

BRE	**ERE**	正規表現の説明
`\ . [ ] ^ $ *`	`\ . [ ] ^ $ *`	共通のメタ文字
`\+ \? \( \) \{ \} |`		”\” でエスケープされた、BRE のみで用いるメタ文字
	`+ ? ( ) { } |`	”\” でエスケープされ無い、ERE のみで用いるメタ文字
`c`	`c`	非メタ文字”c” にマッチ
`\c`	`\c`	c” 自身がメタ文字でも”c” というそのままの文字とマッチ
`.`	`.`	改行を含めた如何なる文字ともマッチ
`^`	`^`	文字列の最初とマッチ
`$`	`$`	文字列の最後とマッチ
`\<`	`\<`	単語先頭とマッチ
`\>`	`\>`	単語末尾とマッチ
`[abc…]`	`[abc…]`	”abc…” のいずれかの文字にマッチ
`[^abc…]`	`[^abc…]`	”abc…” 以外の文字にマッチ
`r*`	`r*`	”r” という正規表現の 0 回以上にマッチ
`r\+`	`r+`	”r” という正規表現の 1 回以上にマッチ
`r\?`	`r?`	”r” という正規表現の 0 回か 1 回にマッチ
`r1|r2`	`r1|r2`	”r1” か”r2” という正規表現のいずれかにマッチ
`\(r1|r2\)`	`(r1|r2)`	”r1” か”r2” という正規表現のいずれかにマッチし、それを括弧で囲まれた正規表現と見なす

Table 1.24: BRE と ERE のメタ文字

emacs の正規表現は、**ERE** 同様の”+” と”?” をメタ文字と扱う拡張をしてはありますが、基本的に **BRE** です。これら文字を emacs の正規表現で”\” でエスケープする必要はありません。

grep(1) をつかうと正規表現を使って文字列探索ができます。

例えば、次を試してみて下さい:

```
$ 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
```

ティップ
項9.2.7を参照下さい。

1.6.3 置換式

置換式の場合、一部の文字に特別な意味があります

置換式	置換式を置換する文字の説明
&	正規表現がマッチしたもの (emacs では \& を使用)
\n	n 番目の括弧で囲まれた正規表現にマッチしたもの (”n” は数字)

Table 1.25: 置換式

Perl の代替文字列には”&” に代えて”$&” が使われ、”\n” に代えて”$n” が使われます。

例えば、次を試してみて下さい:

```
$ 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=1abc2efg3hij4=
```

ここで、括弧で囲まれた正規表現のスタイルと、マッチした文字列が異なるツール上でテキスト置換処理にどう使われるかとに注目下さい。

これらの正規表現は一部エディター内でカーソールの動きやテキスト置換アクションに対しても使えます。

シェルコマンドラインの行末のバックスラッシュ”\” は改行をホワイトスペース文字としてエスケープするので、シェルコマンドライン入力を次行に継続させます。

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下さい。

1.6.4 正規表現を使ったグローバル置換

ed(1) コマンドは次のようにすると”file” 中に存在する全ての”FROM_REGEX” を”TO_TEXT” で置換できます。

```
$ ed file <<EOF
,s/FROM_REGEX/TO_TEXT/g
w
q
EOF
```

sed(1) コマンドは次のようにすると”file” 中に存在する全ての”FROM_REGEX” を”TO_TEXT” で置換できます。

```
$ sed -i -e 's/FROM_REGEX/TO_TEXT/g' file
```

vim(1) コマンドは ex(1) コマンドを使い次のようにすると”file” 中に存在する全ての”FROM_REGEX” を”TO_TEXT” で置換できます。

```
$ vim '+%s/FROM_REGEX/TO_TEXT/gc' '+w' '+q' file
```

ティップ
上記の"c" フラグをにより各置換毎に対話型の確認をします。

複数ファイル ("file1" と"file2" と"file3") を vim(1) や perl(1) で同様に正規表現を用いて処理できます。

```
$ vim '+argdo %s/FROM_REGEX/TO_TEXT/ge|update' '+q' file1 file2 file3
```

ティップ
上記の"e" フラグをにより"No match" エラーでマッピングが停止することを防ぎます。

```
$ perl -i -p -e 's/FROM_REGEX/TO_TEXT/g;' file1 file2 file3
```

perl(1) の例中で、"-i" はその場で各ターゲットファイルの編集、"-p" は与えられたすべてのファイルに関する暗黙的なループを意味します。

ティップ
"-i" の代わりに"-i.bak" という引数を用いるとオリジナルファイル名に".bak" をつけたファイル名でオリジナルファイルが保管されます。複雑な置換のエラーからの復元が簡単にできます。

注意
ed(1) や vim(1) は **BRE** です。一方、perl(1) は **ERE** です。

1.6.5 テキストファイルからのデーター抽出

2004 年以前の元 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
```

ティップ
最新のDebian のリーダーの歴史に関しては、"A Brief History of Debian" を参照下さい。

Awk はこういったタイプのファイルからデーターを抽出するために良く使われます。

例えば、次を試してみて下さい:

```
$ awk '{ print $3 }' <DPL                    # month started
August
April
January
January
April
April
March
$ awk '($1=="Ian") { print }' <DPL           # DPL called Ian
Ian     Murdock   August  1993
Ian     Jackson   January 1998
$ awk '($2=="Perens") { print $3,$4 }' <DPL # When Perens started
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。

例えば、次を試してみて下さい:

```
$ while read first last month year; do
    echo $month
  done <DPL
... b'' 最 b''b'' 初 b''b'' の b'' Awk b'' 例 b''b'' と b''b'' 同 b''b'' じ b''b'' 出 b''b'' 力
    b''
```

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" をシェルでうまく解読できます。

```
$ oldIFS="$IFS"   # save old value
$ IFS=':'
$ while read user password uid gid rest_of_line; do
    if [ "$user" = "bozo" ]; then
      echo "$user's ID is $uid"
    fi
  done < /etc/passwd
bozo's ID is 1000
$ IFS="$oldIFS"   # restore old value
```

(同じことを Awk を使って行うには、フィールドセパレータ設定は"FS=':'" とします。)

IFS はパラメーター展開、コマンド置換、数式展開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> と <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下さい。シェルがスクリプトの一部を入力として解釈した場合に、奇妙なことが起きるかもしれません。

```
$ IFS=":,"                          # use ":" and "," as IFS
$ echo IFS=$IFS,   IFS="$IFS"     # echo is a Bash builtin
IFS=  , IFS=:,
$ date -R                           # just a command output
Sat, 23 Aug 2003 08:30:15 +0200
$ echo $(date -R)                   # sub shell --> input to main shell
Sat  23 Aug 2003 08 30 36 +0200
$ unset IFS                         # reset IFS to the default
$ echo $(date -R)
Sat, 23 Aug 2003 08:30:50 +0200
```

1.6.6 コマンドをパイプするためのスクリプト断片

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。

スクリプト断片 (1 行入力)	コマンドの効果
`find /usr -print`	"/usr" の下の全ファイル発見
`seq 1 100`	1 から 100 までプリント
`| xargs -n 1 <command>`	パイプからの各項目を引数としてコマンドを反復実行
`| xargs -n 1 echo`	パイプからのホワイトスペースで分離された項目を行に分割
`| xargs echo`	パイプからの全ての行を 1 行にマージ
`| grep -e <regex_pattern>`	<regex_pattern> を含む行を抽出
`| grep -v -e <regex_pattern>`	<regex_pattern> を含まない行を抽出
`| cut -d: -f3 -`	":" で区切られた 3 番目のフィールドを抽出 (passwd ファイルなど)
`| awk '{ print $3 }'`	ホワイトスペースで区切られた 3 番目のフィールドを抽出
`| awk -F'\t' '{ print $3 }'`	タブで区切られた 3 番目のフィールドを抽出
`| col -bx`	バックスペースを削除し、タブをスペースに変換
`| expand -`	タブをスペースに変換
`| sort| uniq`	入力をソートし重複箇所を削除
`| tr 'A-Z' 'a-z'`	大文字を小文字に変換
`| tr -d '\n'`	複数行を 1 行に連結
`| tr -d '\r'`	キャリッジリターンを削除
`| sed 's/^/# /'`	各行頭に"#" を追加
`| sed 's/\.ext//g'`	".ext" を削除
`| 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

Table 1.26: コマンドをパイプするためのスクリプト断片

1 行のシェルスクリプトは find(1) や xargs(1) を使って非常に複雑な操作を多くのファイルに繰り返し実行できます。項10.1.5と項9.3.9を参照下さい。

シェルの対話モードを使うのが複雑過ぎるようになったときには、シェルのスクリプトを書くのも一計です (項12.1参照下さい)。

Chapter 2

Debian パッケージ管理

注意
本章は最新安定版リリースがコード名: `bullseye` と言う前提で書かれています。

Debian は、フリーソフトウエアーのコンパイル済みバイナリーパッケージからなる整合性あるディストリビューションを作り、そのアーカイブを通じてそれらを頒布するボランティア組織です。

Debian のアーカイブは、HTTP や FTP 法によるアクセスされるための多くのリモートのミラーサイトとして提供されています。それは、CD-ROM/DVD によっても提供されています。

Debian のパッケージ管理システムは、適正に使われれば、バイナリーパッケージの整合性ある組み合わせがアーカイブからシステムにインストールされるようになっています。現在、amd64 アーキテクチャーでは 62716 つのパッケージが利用できます。

Debian のパッケージ管理システムは、多彩な歴史があり、使用されるフロントエンドのユーザープログラムやバックエンドのアーカイブへのアクセス方法に多くの選択肢があります。現在は以下を推薦します。

- パッケージのインストール・削除や dist-upgrade を含む全ての対話的コマンドライン操作を提供する、apt(8)。Debian Jessie (Debian 8) から提供。
- スクリプトから Debian のパッケージ管理をするためによぶ、apt-get(8)。(古い Debian システム等で)`apt` が使えない際の控えのオプション。
- インストールされたパッケージを管理したり、使用可能なパッケージを探索するためのインタラクティブなテキストインターフェースを提供する、aptitude(8)

2.1 Debian パッケージ管理の前提条件

2.1.1 パッケージ設定

Debian システム上でのパッケージ設定の要点を次に記します。

- システム管理者による手動の設定は尊重されます。言い換えれば、パッケージ設定システムは利便性のために勝手な設定をしません。
- 各パッケージは、パッケージの初期インストールプロセスを助けるための debconf(7) と呼ばれる標準化されたユーザーインターフェースを使用し、それぞれ毎の設定スクリプトとともに提供されます。
- Debian の開発者はパッケージの設定スクリプトによりユーザーのアップグレードが滞りなく進むように最大限の努力を行います。

パッケージ	ポプコン	サイズ	説明
apt	V:868, I:999	4299	アドバンスドパッケージツール (APT)、”http” や”ftp” や”file” というアーカイブへのアクセス方法を dpkg に提供するフロントエンド (apt/apt-get/apt-cache コマンドを含む)
aptitude	V:80, I:431	4249	aptitude(8) を使うインタラクティブなターミナルベースのパッケージマネージャー
tasksel	V:36, I:975	393	Debian システムにタスクをインストールするための選択ツール (APT のフロントエンド)
unattended-upgrades	V:324, I:447	325	セキュリティーアップデートの自動インストールを可能にする APT の拡張パッケージ
dselect	V:3, I:32	2591	ターミナルベースのパッケージマネージャー (過去の標準、APT や他の旧式のアクセス法のフロントエンド)
dpkg	V:925, I:999	6856	Debian のためのパッケージ管理システム
synaptic	V:47, I:320	7873	グラフィカルなパッケージマネージャー (APT の GNOME フロントエンド)
apt-utils	V:339, I:996	1162	APT ユーティリティープログラム: apt-extracttemplates(1) と apt-ftparchive(1) と apt-sortpkgs(1)
apt-listchanges	V:369, I:851	421	パッケージ変更履歴の通知ツール
apt-listbugs	V:7, I:12	465	APT による各インストール前にクリチカルバグをリストする
apt-file	V:18, I:78	90	APT パッケージ探索ユーティリティー -- コマンドラインインターフェース
apt-rdepends	V:0, I:6	40	パッケージの依存関係を再帰的にリスト

Table 2.1: Debian のパッケージ管理ツールのリスト

- システム管理者にはパッケージされたソフトウエアーの全機能が利用可能です。ただしセキュリティーリスクのある機能はデフォールトのインストール状態では無効にされています。
- セキュリティーリスクのあるサービスを手動でアクティベートした場合は、リスクの封じ込めはあなたの責任です。
- システム管理者は難解奇異な設定を手動で有効にはできます。ただこんなことをすればポピュラーな一般の補助プログラムと干渉してしまうかもしれません。

2.1.2 基本的な注意事項

警告
ランダムな混合のスイーツからパッケージをインストールしてはいけません。コンパイラーの ABI とかライブラリー のバージョンとかインタープリターの機能等のシステム管理に関する深い知見が必要なパッケージの整合性がきっと破壊されます。

初心者の Debian システム管理者は **Debian** の安定版 **stable** リリースをセキュリティーアップデートを適用しながら使うべきです。Debian システムを非常によく理解するまでは、用心として次の有効なアクションですら避けておくべきと考えます。次は留意点です。

- ”/etc/apt/sources.list” の中にテスト版 **testing** とか不安定版 **unstable** とかを含めません。
- ”/etc/apt/sources.list” の中に標準の Debian と Debian 以外の Ubuntu のようなアーカイブを混在させません。
- ”/etc/apt/preferences” を作成しません。

- パッケージ管理ツールのデフォールトを影響を理解せずに変更しません。
- ランダムなパッケージを”dpkg -i <random_package>” でインストールしません。
- ランダムなパッケージを”dpkg --force-all -i <random_package>” で絶対インストールしません。
- ”/var/lib/dpkg/” の中のファイルを消去や改変しません。
- ソースから直接コンパイルしたソフトウエアープログラムをインストールする際にシステムファイルを上書きしません。
  - 必要な場合は”/usr/local/” か”/opt/” 中にインストールします。

上記のアクションで起きる Debian パッケージシステムへのコンパチブルでない効果はシステムを使えなくするかもしれません。

ミッションクリティカルなサーバーを走らせる真剣な Debian システム管理者は更なる用心をすべきです。

- 安全な条件下であなたの特定の設定で徹底的にテストすることなくセキュリティーアップデートをも含めた如何なるパッケージもインストールをしてはいけません。
  - システム管理者のあなたがシステムに対して最終責任があります。
  - Debian システムの長い安定性の歴史それ自体は何の保証でもありません。

2.1.3 永遠のアップグレード人生

私が上記で警告したとはいえ、自分自身で管理するデスクトップ環境では Debian のテスト版 testing や不安定版 unstable のスイーツを自分のメインのシステムとして使おうと多くの本書の読者が望むことは分かっています。システムは非常に快調に動くし、頻繁に更新されるし、最新の機能が提供されるからです。

注意
あなたの業務サーバーには、セキュリティーアップデートをした安定版 stable スイーツを推薦します。管理に限られた時間しか割けないデスクトップ PC に関しても同様の事が言えます。

”/etc/apt/sources.list” の中のディストリビューション文字列を、”testing” とか”unstable” というスイーツ名、もしくは”bookworm とか”sid” というコード名に単に設定するだけで十分です。

testing や unstable を使うことは大変楽しいけれど、リスクがついてきます。Debian システムの unstable スイーツさえおおむね非常に安定に見えますが、Debian システムの testing や unstable スイーツでは過去パッケージ上の問題をいくつか経験して来てるし、その一部は簡単には解決できないものでした。結構痛い目に会うことになるかもしれませんよ。時々、壊れたパッケージや機能の欠損が数週間続くことが起こります。

Debian パッケージのバグからの早急かつ簡単な復元を確実にするいくつかのアイデアがここにあります。

- Debian システムの安定版 stable スイーツを別のパーティションにインストールし、システムをデュアルブータブル化
- レスキューブートのためのインストール用 CD を手元に確保
- apt-listbugs をインストールしてアップグレードの前に Debian バグトラッキングシステム (BTS) をチェックを考慮
- 問題回避するのに十分なだけのパッケージシステムの基盤を学習
- chroot か類似の環境を作り事前に最新のシステムを実行 (項9.10参照下さい)

(これらの用心のための方策の何れもできないなら、テスト版 testing や不安定版 unstable スイーツを使うのにはあなたはきっと準備不足です。)

以下に記すことにより悟りを開けば、アップグレード地獄という果てしない因果応報の葛藤から人は解脱し、Debian の涅槃の境地に到達できます。

2.1.4 Debian アーカイブの基本

Debian アーカイブをシステムユーザーの視点から見てみます。

ティップ
Debian アーカイブの正式のポリシーは Debian ポリシーマニュアル、第 2 章 - Debian アーカイブに規定されています。

典型的な HTTP アクセスの場合、現在の安定版 stable=bullseye システムを例にとると、次の様に”/etc/apt/sources ファイルの中にアーカイブは規定されています。

```
deb http://deb.debian.org/debian/ bullseye main contrib non-free
deb-src http://deb.debian.org/debian/ bullseye main contrib non-free

deb http://security.debian.org/ bullseye/updates main contrib
deb-src http://security.debian.org/ bullseye/updates main contrib
```

上記で、次の安定版 stable がリリースされて驚かされ無いように、私はスイート名の”stable” でなくコード名の”bullseye” を使います。

”/etc/apt/sources.list” の意味は sources.list(5) に記載されていて、要点は以下です。

- ”deb” 行がバイナリーパッケージのための定義です。
- ”deb-src” 行がソースパッケージのための定義です。
- 一番目の引数は、Debian アーカイブの root URL です。
- 二番目の引数は、スイーツ名かコード名のどちらかで与えられるディストリビューション名です。
- 三番目次の引数は、Debian アーカイブの中の有効なアーカイブのエリア名のリストです。

ソース関連のメタデーターにアクセスしない aptitude のためだけなら”deb-src” 行は安全に省略 (もしくは”#” を行頭に挿入してコメントアウト) することができます。こうするとアーカイブのメタデーターの更新速度が向上します。URL は”http://” や”ftp://” や”file://” 等々の何れも可能です。

ティップ
もし上記の例で”bullseye” ではなく”sid” が使われる場合には、セキュリティーアップデートのための”deb: http://security.debian.org/ ...” 行は不要です。安定版 stable とテスト版 testing (即ち bullseye と bookworm) にのみセキュリティーアップデートがあります。”sid” (不安定版 unstable) のためにはセキュリティーアップデートのアーカイブが存在しません。

次は設定ファイル内に用いられる Debian アーカイブサイトの URL とスイーツ名もしくはコード名です。

注意
セキュリティーアップデートされた純粋な安定版 **stable** リリースのみが最善の安定性を提供します。一部 **testing** や **unstable** 由来のパッケージを混用してほとんど **stable** リリースを走らせることは、純粋な **unstable** リリースを走らせるよりリスクがあります。**stable** リリースの下で最新バージョンのいくつかのプログラムが本当に必要なら、bullseye-updatesや http://backports.debian.org (項2.7.4参照下さい) サービスからのパッケージを使って下さい。これらのサービスは細心の注意を持って使う必要があります。

アーカイブの URL	スイート名 (コード名)	目的
http://deb.debian.org/debian/	stable (bullseye)	安定版 (bullseye) のリリース
http://deb.debian.org/debian/	testing (bookworm)	テスト版 (bookworm) のリリース
http://deb.debian.org/debian/	unstable (sid)	不安定版 (sid) のリリース
http://deb.debian.org/debian/	experimental	実験的プリリリース (任意、開発者専用)
http://deb.debian.org/debian/	stable-proposed-updates	次回安定版ポイントリリース用のアップデート (任意)
http://security.debian.org/	stable/updates	安定版用のセキュリティーアップデート (重要)
http://security.debian.org/	testing/updates	テスト版用のセキュリティーアップデート (重要)
http://deb.debian.org/debian/	bullseye-updates	bullseye のためのスパムフィルターや IM クライアント他用のコンパチブルなアップデート
http://deb.debian.org/debian/	bullseye-backports	bullseye のための新しくバックポートされたパッケージ (任意)

Table 2.2: Debian アーカイブサイトのリスト

注意
基本的に、stable か testing か unstable のスイーツの内の 1 つだけを"deb" 行に書くべきです。もし、stable と testing と unstable のスイーツの何らかの組み合わせを"deb" 行に書けば、APT プログラムは、最新のアーカイブのみが有効であるにもかかわらず、実行速度が低下します。"/etc/apt/preferences" ファイルがはっきりとした目的を持って使われている場合 (項2.7.3) のみ複数のリストに意味があります。

ティップ
stable や testing スイーツの Debian システムでは、上記の例のようにセキュリティーアップデートを有効とするように"/etc/apt/sources.list" の中に"http://security.debian.org/" の行を含めることはいいことです。

注意
stable アーカイブのセキュリティーバグは Debian のセキュリティーチームにより修正されます。本活動は非常に厳格で信頼できるものです。testing アーカイブのセキュリティーバグは Debian の testing セキュリティーチームにより修正されます。諸所の 事情で、本活動は stable ほどは厳格ではなく、修正された unstable パッケージの移行を待つ必要があるかもしれません。unstable アーカイブのセキュリティーバグは個別のメンテナにより修正されます。活発にメンテされている unstable パッケージはアップストリームのセキュリティー修正を使うことで通常比較的良い状態です。Debian がセキュリティーバグへ如何に対応するかに関しては Debian security FAQ を参照下さい。

エリア	パッケージ数	パッケージ構成要素のクライテリア
main	61595	DFSG に完全準拠し、non-free のパッケージに非依存 (main = 主要)
contrib	349	DFSG に完全準拠だが、non-free のパッケージに依存有り (contrib = 寄与)
non-free	772	非 DFSG 準拠

Table 2.3: Debian アーカイブエリアのリスト

ここで、上記にあるパッケージ数は amd64 アーキテクチャーに関する数字です。main エリアのアーカイブのみが Debian システムです (項2.1.5参照)。

Debian アーカイブの構成は、各アーカイブの URL の後ろに dists か pool をつけた URL にブラウザーを向ければ学習できます。

ディストリビューションは、スイーツとコード名の 2 つの方法で言及されます。この他にディストリビューションと言う言葉は多くの文書でスイーツの同義語としても使われています。スイーツとコード名の関係は次のようにまとめられます。

タイミング	スイーツ = 安定版 **stable**	スイーツ = テスト版 **testing**	スイーツ = 不安定版 **unstable**
bullseye リリース後	コード名 = bullseye	コード名 = bookworm	コード名 = sid
bookworm リリース後	コード名 = bookworm	コード名 = trixie	コード名 = sid

Table 2.4: スイーツとコード名の関係

コード名の歴史は、Debian FAQ: 6.2.1 Which other codenames have been used in the past? に記載されています。

比較的厳格な Debian アーカイブの用語法では、” セクション” という言葉はアプリケーションの分野によるパッケージ分類に特化して使われます。(しかし、”main セクション” という言葉は main エリアを提供する Debian アーカイブ部分を表現するのにしばしば使われています。)

Debian デベロッパー (DD) が不安定版 unstable アーカイブに新たなアップロードを (incoming での処理を経由して) する度毎に、アップロードするパッケージが最新の不安定版 unstable アーカイブの最新のパッケージ集合とコンパチブルであるようにする義務が DD にはあります。

重要なライブラリーのアップグレード他の理由で DD がこのコンパチビリティーを壊す際には、debian-devel のメーリングリスト他に通常アナウンスがされます。

Debian のアーカイブ管理スクリプトによって非安定版 unstable アーカイブからテスト版 testing アーカイブへパッケージ集合が移動される前に、アーカイブ管理スクリプトはパッケージの成熟度 (約 10 日経過) と RC バグレポート状況を確認するばかりでなく、テスト版 testing アーカイブの最新パッケージ集合とのコンパチブルであるようにするように努めます。このプロセスがあるので、テスト版 testing アーカイブは非常に新しくかつ使いやすいのです。

リリースチームによる徐々のアーカイブ凍結過程を通じて、少々の手動の介入を伴いつつテスト版 testing アーカイブは完全に整合性をもったバグの無い状態へと徐々に熟成されます。そして、古いテスト版 testing アーカイブのコード名を新たな安定版 stable アーカイブへと割り当て、新たなコード名を新たなテスト版 testing アーカイブへと割り当てることで、新たな安定版 stable がリリースされます。新たなテスト版 testing アーカイブの当初の内容は、新たにリリースされた安定版 stable アーカイブとまったく同じです。

不安定版 unstable もテスト版 testing アーカイブもともにいくつかの要因で一時的に細かな問題発生があるかもしれません。

- ブロークンなパッケージのアーカイブへのアップロード (主に unstable にて)
- 新規パッケージをアーカイブに受け入れる際の遅延 (主に unstable にて)
- アーカイブの同期のタイミング問題 (testing と unstable の両方にて)。
- パッケージの除去などのアーカイブへの手動の介入 (どちらかといえば testing にて)、等。

もしこれらのアーカイブを使おうと考えるなら、この種の細かな問題の修復や回避は必須技能です。

注意
たとえいつも非安定版 unstable やテスト版 testing アーカイブを使っていようとも、ほとんどのデスクトップユーザーは新たな安定版 stable リリースの後約数ヶ月はセキュリティーアップデートされた安定版 stable アーカイブを使うべきです。この移行期は、非安定版 unstable もテスト版 testing アーカイブの何れももほとんどの人に良いものではありません。非安定版 unstable アーカイブを使おうとすると、核となるパッケージが大アップグレードの嵐に見舞われるので、あなたのシステムをうまく使える状態に保つのは困難です。テスト版 testing アーカイブを使おうとしても、安定版 stable アーカイブとほとんど同じ内容でセキュリティーサポートはありません (Debian testing-security-announce 2008-12)。1 ヶ月ほど経てば、非安定版 unstable アーカイブなら注意を払えば使えるかもしれません。

ティップ
テスト版 testing アーカイブを追跡している際には、除去されたパッケージによって引き起こされる問題は該当するバグ修正のためにアップロードされたパッケージを非安定版 unstable アーカイブからインストールすれば通常回避できます。

アーカイブの定義は、Debian ポリシーマニュアルを参照下さい。

- ”セクション”
- ”優先度 (priorities)”
- ”ベースシステム”
- ”必須パッケージ”

2.1.5 Debian は 100% フリーソフトウェアーです

Debian は以下の理由で 100% フリーソフトウエアーです:

- Debian はユーザーの自由を尊重すべくデフォルトではフリーソフトウエアーのみをインストールします。
- Debian は main 中にはフリーソフトウエアーのみを提供します。
- Debian は main からのフリーソフトウエアーのみを実行することを推奨します。
- main 中のいかなるパッケージも non-free や contrib 中のいずれのパッケージに依存しないし、これらを推薦することもありません。

一部の人は以下の 2 つの事実が矛盾するのでは無いかとの疑問を持ちます。

- 「Debian は 100% フリーソフトウェアーであり続けます」。Debian 社会契約の第一項)
- Debian サーバーは non-free や contrib パッケージをホストします。

これらは以下の理由で矛盾しません。

- Debian システムは 100% フリーソフトウエアーでそのパッケージは Debian サーバーの main エリア中にホストされます。
- Debian システム外のパッケージは Debian サーバーの non-free と contrib エリア中にホストされます。

これらは Debian 社会契約の第 4 項と第 5 項中に正確に説明されています:

- 私たちはユーザーとフリーソフトウェアーを大切にします
  - 私たちはユーザーとフリーソフトウェアーコミュニティーからの要求に従います。彼らの関心を最優先に考えます。私たちはさまざまな状況におけるコンピューター利用環境の運用に関して、ユーザーの必要を満たすように行動します。私たちは Debian システム上での利用を目的としたフリーではない著作物に敵対することはありません。またそのような著作物を作成または利用する人々に対して、料金を徴収することはありません。私たちは、Debian システムとその他の著作物の両方を含むディストリビューションを、第三者が作成することも認めています。その際、私たちは料金を徴収しません。私たちはこれらの目標を増進させるために、これらのシステムの使用を妨げるような法的な制約のない、高品質な素材を統合したシステムを提供します。
- 私たちのフリーソフトウェアー基準に合致しない著作物について

  - 私たちは、Debian フリーソフトウェアーガイドラインに適合していない著作物を使わなければならないユーザーがいることを認めています。このような著作物のために、私たちはアーカイブに「`contrib`」と「`non-free`」という領域を作りました。これらの領域にあるパッケージは、Debian 上で使用できるよう設定されていますが、Debian システムの一部ではありません。私たちは、CD 製造業者がこれらの領域にあるパッケージを彼らの CD に収録して配布できるかどうか判断する際に、それぞれのパッケージのライセンスを読んで決めるよう奨めています。このように、フリーではない著作物は Debian の一部ではありませんが、その使用をサポートし、フリーではないパッケージのための (バグ追跡システムやメーリングリストのような) インフラストラクチャーを用意しています。

ユーザーは `non-free` や `contrib` エリア中のパッケージを使用するリスクを認識すべきです。

- そのようなソフトウエアーパッケージに関する自由の欠如
- そのようなソフトウエアーパッケージに関する Debian からのサポートの欠如 (Debian はソフトウエアーのソースコードに適切なアクセスなしにはソフトウエアーをサポートできません。)
- あなたの 100% フリーソフトウエアーの Debain システムへの汚染

Debian フリーソフトウェアーガイドラインは Debian のフリーソフトウエアー基準です。Debian は「ソフトウエアー」に関して、パッケージ中の文書、ファームウエアー、ロゴ、アート作品を含む最も広義の解釈をします。このことにより Debian のフリーソフトウエアー基準は非常に厳格なものとなります。

`main` に関するこの厳格なフリーソフトウエアー基準に合致させるため、Debian は Firefox や Thunderbird や Seamonkey 等のソフトウエアーパッケージからそれらのロゴやアート作品データーを削除した 過去ブランドを削除された Mozilla ソフトウエアーを、それぞれ Iceweasel や Icedove や Iceape として出荷していました。この様な問題が解決した Debian Stretch (Debian 9) 以降のリリースで、これらのパッケージはそれらの元来の名前に戻されました。

典型的な `non-free` や `contrib` パッケージは以下のタイプの自由に頒布できるパッケージを含んでいます。

- GCC や Make 等の変更不可部分付きの GNU フリー文書利用許諾契約書 に基づく文書パッケージ。(主に `non-free/doc` セクション中にある)
- 項9.9.6 に列記された中で non-free とあるソースコード無しのバイナリーデーターを含むファームウエアーパッケージ。(主に `non-free/kernel` セクション中にある)
- 商用使用やコンテント変更に関する制約のあるゲームやフォントのパッケージ。

`non-free` と `contrib` パッケージの数は `main` パッケージの数の 2% 以下です。`non-free` や `contrib` エリアへのアクセスを有効にしてもパッケージ起源は不明瞭になりません。aptitude(8) をインタラクティブでフルスクリーンに使用すると、どのエリアからどのパッケージをインストールするのかを完全に可視化しコントロールできるので、あなたのシステムをあなたの意向通りの自由の程度に合わせて維持できます。

2.1.6 パッケージ依存関係

Debian システムはコントロールファイル中のバージョン情報付きのバイナリー依存関係宣言を通して整合性のあるバイナリーパッケージの集合を提供します。ここにその少々簡素化し過ぎの定義を示します。

- ”Depends”
  - これは絶対依存を宣言し、このフィールドにリストされた全てのパッケージは同時または事前にインストールされていなければいけません。
- ”Pre-Depends”
  - これは、リストされたパッケージが事前にインストールを完了している必要がある以外は、Depends と同様です。
- ”Recommends”

  - これは強いが絶対でない依存を宣言します。多くのユーザーはこのフィールドにリストされたパッケージ全てがインストールされていなければ、当該パッケージを望まないでしょう。

- ”Suggests”
  - これは弱い依存を宣言します。このパッケージの多くのユーザーはこのフィールドにリストされたパッケージをインストールすればメリットを享受できるとは言え、それら抜きでも十分な機能が得られます。

- ”Enhances”
  - これは Suggests 同様の弱い依存を宣言しますが、依存作用の方向が逆です。

- ”Breaks”
  - これは通常バージョン制約付きでパッケージのインコンパチビリティーを宣言します。一般的にこのフィールドにリストされた全てのパッケージをアップグレードすることで解決します。

- ”Conflicts”
  - これは絶対的排他関係を宣言します。このフィールドにリストされた全てのパッケージを除去しない限り当該パッケージをインストールできません。

- ”Replaces”
  - 当該パッケージによりインストールされるファイルがこのフィールドにリストされたパッケージのファイルを置き換える際にこれを宣言します。

- ”Provides”
  - 当該パッケージがこのフィールドにリストされたパッケージのファイルと機能の全てを提供する際にこれを宣言します。

注意
正常な設定として”Provides” と”Conflicts” と”Replaces” とを単一バーチャルパッケージに対し同時宣言することがあります。こうするといかなる時にも当該バーチャルパッケージを提供する実パッケージのうち確実に一つだけがインストールされます。

ソースの依存関係をも含む正式の定義は the Policy Manual: Chapter 7 - Declaring relationships between packages にあります。

2.1.7 パッケージ管理のイベントの流れ

パッケージ管理の簡略化されたイベントの流れをまとめると次のようになります。

- 更新 (”`apt update`” か”`aptitude update`” か”`apt-get update`”):
  1. アーカイブメタデーターをリモートアーカイブから取得
  2. APT が使えるようローカルメタデーターの再構築と更新

- 更新 (”`apt upgrade`” と”`apt full-upgrade`” か、”`aptitude safe-upgrade`” と”`aptitude full-upgrade`” か、”`apt-get upgrade`” と”`apt-get dist-upgrade`”):
  1. 全てのインストール済みパッケージに関して、通常最新の利用可能なバージョンが選ばれる候補バージョンを選択 (例外については項2.7.3参照下さい)
  2. パッケージ依存関係解決の実行

3. もし候補バージョンがインストール済みバージョンと異なる際には、選ばれたバイナリーパッケージをリモートアーカイブから取得
4. 取得バイナリーパッケージの開梱
5. **preinst** スクリプトの実行
6. バイナリーファイルのインストール
7. **postinst** スクリプトの実行

- インストール (”`apt install ...`” か”`aptitude install ...`” か”`apt-get install ...`”):

1. コマンドラインにリストされたパッケージの選択
2. パッケージ依存関係解決の実行
3. 選ばれたバイナリーパッケージをリモートアーカイブから取得
4. 取得バイナリーパッケージの開梱
5. **preinst** スクリプトの実行
6. バイナリーファイルのインストール
7. **postinst** スクリプトの実行

- 削除 (”`apt remove ...`” か”`aptitude remove ...`” か”`apt-get remove ...`”):

1. コマンドラインにリストされたパッケージの選択
2. パッケージ依存関係解決の実行
3. **prerm** スクリプトの実行
4. 設定ファイル以外のインストール済みファイルの削除
5. **postrm** スクリプトの実行

- 完全削除 (”`apt purge ...`” か”`aptitude purge ...`” か”`apt-get purge ...`”):

1. コマンドラインにリストされたパッケージの選択
2. パッケージ依存関係解決の実行
3. **prerm** スクリプトの実行
4. 設定ファイルを含めたインストール済みファイルの削除
5. **postrm** スクリプトの実行

上記では全体像の理解のためにわざと技術詳細を端折っています。

2.1.8 パッケージ管理のトラブルへの応急対処法

内容が正確な正式文書を読むように心がけるべきです。まず Debian に特定のことが記載された”`/usr/share/doc/<pack`
を最初に読むべきです。また”`/usr/share/doc/<package_name>/`” の中にある他の文書も参照すべきです。
項1.4.2に書かれたようなシェル設定がされていれば、次のようにタイプして下さい。

```
$ cd <package_name>
$ pager README.Debian
$ mc
```

さらに詳しい情報を得るには”`-doc`” というサフィクスを持った対応する文書パッケージをインストールする必要があるかもしれません。

特定パッケージに関する問題に出会った際には、Debian バグトラッキングシステム (BTS) サイトを必ず確認します。

”`site:debian.org`” や”`site:wiki.debian.org`” や”`site:lists.debian.org`” 等を含む検索語で Google を検索します。

バグ報告をする際には、reportbug(1) コマンドを使います。

ウェッブサイト	コマンド
Debian バグトラッキングシステム (BTS) のホームページ	`sensible-browser "http://bugs.debian.org/"`
既知のパッケージに関するバグレポート	`sensible-browser "http://bugs.debian.org/<package_name>"`
既知のバグ番号に関するバグレポート	`sensible-browser "http://bugs.debian.org/<bug_number>"`

Table 2.5: 特定パッケージの問題解決のためのキーとなるウェッブサイトのリスト

2.2 基本的パッケージ管理操作

Debian システム上でのレポジトリーを使ったパッケージ管理操作は Debian システム上にある多くの APT を使うパッケージ管理ツールを使いできます。ここでは、`apt` /`apt-get` / `apt-cache` や `aptitude` といった 3 つの基本的なパッケージ管理ツールを説明します。

パッケージをインストールしたりパッケージのメタデーターを更新するようなパッケージ管理操作には root 権限が必要です。

2.2.1 `apt` と `apt-get`/`apt-cache` と `aptitude` の比較

`aptitude` は筆者が主に使う非常に良いインタラクティブツールではありますが、注意すべき事実があることを知っておくべきです。

- `stable`(安定版) Debian システムにおいて、新リリースがあった後の新リリースシステムへのアップグレードに `aptitude` コマンドを使用することは推薦されません。
  - それには、"`apt full-upgrade`" か"`apt-get dist-upgrade`" を使うことが推薦されます。Bug #411280 参照ください。
- `aptitude` コマンドは時折 `testing`(試験版) や `unstable` (不安定版) Debian システム上でシステムアップグレードをしようとする際に、大量のパッケージ削除を提案することが時々あります。
  - この状況は多くのシステム管理者を驚かせて来ました。パニックしないでください。
  - このようなことは `gnome-core` の様なメタパッケージにより依存・推薦されるパッケージ間のバージョンのずれにより発生するようです。
  - この状況は `aptitude` コマンドのメニューから" 未実行アクションの取り消し" を選択し、`aptitude` を終了し、"`apt full-upgrade`" を使うことで解決できます。

`apt-get` や `apt-cache` コマンドは APT を使う最も基本的なパッケージ管理ツールです。

- `apt-get`/`apt-cache` はコマンドラインのユーザーインターフェースのみを提供します。
- `apt-get` はリリース間のような大掛かりなシステムアップグレードに最適です。
- `apt-get` は頑強で安定なパッケージリゾルバーを提供します。
- `apt-get` はハードウエアリソースへの要求が楽である。メモリーの消費は少なく、実行速度が早い。
- `apt-cache` はパッケージ名や説明に関して標準の regex を使った検索機能を提供します。
- `apt-get` と `apt-cache` は `/etc/apt/preferences` を使って複数のバージョンのパッケージを管理できますが、それはとても面倒です。

`apt` コマンドはパッケージ管理のための上位コマンドラインインターフェースです。基本的に `apt-get` や `apt-cache` 等のコマンドのラッパーで、インタラクティブな用途に良いオプションをデフォルトで有効にしてエンドユーザーインターフェース向けとなっています。

- apt は、apt install としてパッケージをインストールするとフレンドリーなプログレスバーを提供します。
- apt は、ダウンロードされたパッケージが上手くインストールされた後、デフォルトでキャッシュされた .deb パッケージを削除します。

ティップ
ユーザーは インタラクティブ用途には apt(8) コマンドを使うことが推奨されますし、シェルスクリプト中では apt-get(8) や apt-cache(8) コマンドを使うことが推奨されます。

---

aptitude コマンドは最も多芸な APT を使うパッケージ管理ツールです。

- aptitude はフルスクリーンのインタラクティブなテキストユーザーインターフェースを提供します。
- aptitude はコマンドラインのユーザーインターフェースも提供します。
- aptitude はインストールされたパッケージを検査したり利用可能なパッケージを探索したりするような日常のインタラクティブなパッケージ管理に最適です。
- aptitude はハードウエアリソースへの要求が厳しい。メモリーの消費は多く、実行速度も遅い。
- aptitude はパッケージメタデータ全てに関する拡張された regex を使った探索を提供します。
- aptitude は /etc/apt/preferences を使わずに複数のバージョンのパッケージを管理できますし、それは非常に直感的です。

2.2.2 コマンドラインによる基本的なパッケージ管理操作

apt(8) や aptitude(8) や apt-get(8) /apt-cache(8) を使うコマンドラインによるパッケージ管理操作を次に記します。

注意
aptitude コマンドはその拡張されたパッケージリゾルバーのような豊富なフィーチャーとともに提供されますが、この複雑さは Bug #411123 や Bug #514930 や Bug #570377 のようないくつかのリグレッションを引き起こしました (また起こしているかもしれません)。疑義のある場合には、aptitude コマンドに代えて apt や apt-get や apt-cache コマンドを使ってください。

注意
lenny 以降、apt と apt-get と aptitude は自動インストールされたパッケージ状態を共有しているので (項2.5.5参照下さい)、これらのツールを特段の問題なく混用できます (Bug #594490 参照下さい)。

”aptitude why <regex>” は”aptitude -v why <regex>” とすることで、さらに詳しい情報を表示します。同様の情報は”apt rdepends <package>” や”apt-cache rdepends <package>” とすることでも得られます。

aptitude コマンドが最初コマンドラインモードで実行されパッケージ間のコンフリクトのような問題に直面した場合は、プロンプトがでた際に”e” を押すことでフルスクリーンのインタラクティブモードに切り替えられます。

”aptitude” のすぐ後ろにコマンドオプションをつけられます。

詳細は aptitude(8) や”/usr/share/doc/aptitude/README” にある”aptitude user’s manual” を参照下さい。

ティップ
現在でも利用可能な dselect パッケージは、過去のリリースでは推薦されたフルスクリーンのインタラクティブなパッケージ管理ツールでした。

---

apt シンタックス	**aptitude** シンタックス	**apt-get/apt-cache** シンタックス	説明
`apt update`	`aptitude update`	`apt-get update`	パッケージアーカイブメタデーター更新
`apt install foo`	`aptitude install foo`	`apt-get install foo`	”foo” パッケージの候補バージョンをその依存関係とともにインストール
`apt upgrade`	`aptitude safe-upgrade`	`apt-get upgrade`	他のパッケージを削除すること無くインストール済みパッケージの候補バージョンをインストール
`apt full-upgrade`	`aptitude full-upgrade`	`apt-get dist-upgrade`	必要なら他のパッケージを削除しながらインストール済みパッケージの候補バージョンをインストール
`apt remove foo`	`aptitude remove foo`	`apt-get remove foo`	設定ファイルを残したまま”foo” パッケージを削除
`apt autoremove`	N/A	`apt-get autoremove`	既に必要なくなっている自動済みパッケージを削除
`apt purge foo`	`aptitude purge foo`	`apt-get purge foo`	設定ファイルを含めて”foo” パッケージを完全削除
`apt clean`	`aptitude clean`	`apt-get clean`	収集されローカルに貯蔵されたパッケージファイルを完全消去
`apt autoclean`	`aptitude autoclean`	`apt-get autoclean`	収集されローカルに貯蔵されたパッケージファイルのうち古くなったパッケージを消去
`apt show foo`	`aptitude show foo`	`apt-cache show foo`	”foo” パッケージに関する詳細情報を表示
`apt search <regex>`	`aptitude search <regex>`	`apt-cache search <regex>`	<regex> とマッチするパッケージを検索
N/A	`aptitude why <regex>`	N/A	なぜ <regex> とマッチするパッケージがインストールされるのかを説明
N/A	`aptitude why-not <regex>`	N/A	なぜ <regex> とマッチするパッケージがインストールされないのかを説明
N/A	`aptitude search '~i!~M'`	`apt-mark showmanual`	手動インストールされたパッケージをリスト

Table 2.6: apt(8) や aptitude(8) や apt-get(8) /apt-cache(8) を使うコマンドラインによる基本パッケージ管理操作

コマンドオプション	説明
`-s`	コマンド結果のシミュレート
`-d`	インストール / アップグレード無しにダウンロードのみする
`-D`	自動的なインストールや削除の前に簡単な説明を表示

Table 2.7: aptitude(8) に関する特記すべきコマンドオプション

2.2.3 aptitude のインタラクティブな使用

インタラクティブなパッケージ管理のためには aptitude をインタラクティブモードでコンソールのシェルプロンプトから次のように立ち上げます。

```
$ sudo aptitude -u
Password:
```

これによりアーカイブ情報のローカルコピーは更新され、フルスクリーンのパッケージリストがメニュー付きで表示されます。Aptitude の設定ファイルは”~/.aptitude/config” にあります。

ティップ
user の設定ファイルでなく root の設定ファイルを使いたい際には、上記の例で”sudo aptitude ...” の代わりに”sudo -H aptitude ...” を使います。

ティップ
Aptitude はインタラクティブに起動されると次にするアクションを自動的に設定します。その設定が好ましくない場合はメニュー:”Action” → ”Cancel pending actions” からリセットすることができます。

2.2.4 aptitude のキーバインディング

パッケージの状態を閲覧し、” 予定のアクション” の設定をこのフルスクリーンモードで各パッケージするための重要なキーを次に記します。

キー	キーバインディング
F10 もしくくは Ctrl-t	メニュー
?	(より詳細な) キーの意味のヘルプの表示
F10 → ヘルプ → ユーザーマニュアル	ユーザーマニュアルの表示
u	パッケージアーカイブ情報の更新
+	パッケージをアップグレードまたはインストールするとマーク
-	パッケージを削除するとマーク (設定ファイルは温存)
_	パッケージを完全削除するとマーク (設定ファイルも削除)
=	パッケージをホールド
U	全てのアップグレード可能なパッケージをマーク (**full-upgrade** として機能)
g	選ばれたパッケージのダウンロードとインストールをスタート
q	現在のスクリーンを終了し変更を保存
x	現在のスクリーンを終了し変更を廃棄
Enter	パッケージに関する情報閲覧
C	パッケージの変更履歴を閲覧
l	表示されるパッケージの制限を変更
/	最初のマッチを検索
\	最終検索の反復

Table 2.8: aptitude のキーバインディングのリスト

コマンドラインのファイル名の規定や、”l” や”/” を押した後のメニュープロンプトは次に記す aptitude の regex (正規表現) が使われます。aptitude の regex は”~n” で始めそれにパッケージ名を続けた文字列を使うことで明示的にパッケージ名とマッチさせられます。

ティップ
ビジュアルインターフェースで全てのインストール済みパッケージを候補バージョンにアップグレードさせるには”U” を押さなければいけません。これをしないと選ばれたパッケージとそれにバージョン付きの依存関係のある特定のパッケージのみがアップグレードされます。

2.2.5 aptitude の下でのパッケージの表示

インタラクティブなフルスクリーンモードの aptitude(8) はパッケージリスト中のパッケージは次の例のように表示されます。

```
idA   libsmbclient                              -2220kB 3.0.25a-1  3.0.25a-2
```

上記の行は左から次に記すような意味です。

- ” 現状” フラグ (1 番目の文字)
- ” 予定のアクション” フラグ (2 番目の文字)
- ” 自動” フラグ (3 番目の文字)
- パッケージ名
- ” 予定のアクション” に帰属されるディスク空間の使用の変化
- パッケージの現バージョン
- パッケージの候補バージョン

ティップ
”?” を押して表示されるヘルプスクリーンの一番下に全フラグのリストがあります。

現在のローカルの環境設定によって候補バージョンは選ばれます (apt_preferences(5) と項2.7.3を参照下さい)。
” 表示” メニューの下にある数種のパッケージ表示が利用できます。

表示	状況	ビューの説明
パッケージ画面	良好	表 2.10参照 (デフォールト)
推奨を監査	良好	何らかのインストール済みパッケージによって推薦されているがインストールされていないパッケージをリスト
平坦なパッケージリスト	良好	パッケージを分類せずにリスト (regex とともに使用)
Debtags 表示	十分使える	パッケージの debtags のエントリーにより分類したパッケージをリスト
カテゴリー別表示	非推奨	パッケージのカテゴリー別に分類してパッケージをリスト (これに代えて Debtags 表示を利用しましょう)

Table 2.9: aptitude の表示のリスト

注意
パッケージの debtags によるタグ付け状況を改善するのにご協力下さい！

標準” パッケージ画面” はパッケージを dselect にいくつかの機能を加えた感じで分類します。

ティップ
Tasks ビューはあなたのタスクに使うパッケージをいいとこ取りするのに使えます。

分類	ビューの説明
更新可能なパッケージ	section → area → package と整理してパッケージをリスト
新規パッケージ	, ,
インストール済みのパッケージ	, ,
インストールされていないパッケージ	, ,
廃止された、またはローカルで作成されたパッケージ	, ,
仮想パッケージ	同一機能のパッケージをリスト
タスク	タスクに一般的に必要な機能を持つパッケージのリスト

Table 2.10: 標準パッケージ画面の分類

2.2.6 aptitude を使った探索方法

Aptitude はその regex 式機能を通してパッケージを探索する方法をいくつか提供します。

- シェルコマンドライン:
  - マッチするパッケージのインストール状態やパッケージ名や短い説明をリストをすには、”aptitude search '<aptitude_regex>'”
  - パッケージの詳細説明のリストをするには、”aptitude show '<package_name>'”
- 対話型フルスクリーンモード:
  - マッチするパッケージにパッケージビューを絞る、”l”
  - マッチするパッケージを探す、”/”
  - マッチするパッケージを逆方向に探す、”\”
  - 次を探す、”n”
  - 次を逆方向に探す、”N”

ティップ
<package_name> という文字列は、”~” で始めて regex 式と明示されていない限り、パッケージ名との完全な一致検索として扱います。

2.2.7 aptitude の regex 式

aptitude の regex 式は mutt 的な拡張 **ERE** (項1.6.2参照下さい) で aptitude に特定なマッチ規則の拡張は次に示すとおりです。

- regex 部分は、”^” や”.*” や”$” などを使う egrep(1) や awk(1) や perl(1) といった典型的な Unix 的テキストツールで使われる **ERE** と同様です。
- 依存関係を表す <type> はパッケージの相互関係を指定する (depends, predepends, recommends, suggests, conflicts, replaces, provides) の内の 1 つです。
- デフォールトの依存関係は”depends” です。

ティップ
<regex_pattern> がヌル文字列の場合は”~T” をコマンドの直後に使って下さい。

次がショートカットです。

拡張マッチ規則の説明	**regex** 式
パッケージ名とのマッチ	`~n< 名前の regex>`
記述とのマッチ	`~d< 記述の regex>`
タスク名とのマッチ	`~t< タスクの regex>`
debtag とのマッチ	`~G<debtag の regex>`
メンテナとのマッチ	`~m<maintainer の regex>`
パッケージセクションとのマッチ	`~s< セクションの regex>`
パッケージバージョンとのマッチ	`~V< バージョンの regex>`
アーカイブ (archive) とのマッチ	`~A{bullseye,bookworm,sid}`
オリジン (origin) とのマッチ	`~O{debian,…}`
優先度 (priority) とのマッチ	`~p{extra,important,optional,required,standard}`
必須 (essential) パッケージとのマッチ	`~E`
仮想パッケージとのマッチ	`~v`
新規パッケージとのマッチ	`~N`
次のアクションとのマッチ	`~a{install,upgrade,downgrade,remove,purge,hold,keep}`
インストール済みパッケージとのマッチ	`~i`
A-マークのついたインストール済みパッケージとマッチ (自動インストール済みパッケージ)	`~M`
A-マークのついていないインストール済みパッケージとマッチ (管理者が選択したパッケージ)	`~i!~M`
インストール済みかつアップグレード可能なパッケージとマッチ	`~U`
削除済みだが完全削除されていないパッケージとマッチ	`~c`
削除済みか完全削除済みか削除可能なパッケージとマッチ	`~g`
壊れた依存関係宣言をしたパッケージとマッチ	`~b`
<type> の壊れた依存関係を宣言しているパッケージとマッチ	`~B<type>`
<type> の壊れた依存関係を宣言している <pattern> パッケージとマッチ	`~D[<type>:]<pattern>`
<type> の壊れた依存関係を宣言している <pattern> パッケージとマッチ	`~DB[<type>:]<pattern>`
<pattern> マッチするパッケージが <type> の依存関係を宣言しているパッケージとマッチ	`~R[<type>:]<pattern>`
<pattern> マッチするパッケージが <type> の壊れた依存関係を宣言しているパッケージとマッチ	`~RB[<type>:]<pattern>`
他のインストール済みパッケージが依存するパッケージとマッチ	`~R~i`
他のインストール済みパッケージが一切依存しないパッケージとマッチ	`!~R~i`
他のインストール済みパッケージが依存もしくは推薦するパッケージとマッチ	`~R~i|~Rrecommends:~i`
フィルターされたバージョンの <pattern> とマッチ	`~S filter <pattern>`
常に全てのパッケージにマッチ (真)	`~T`
どのパッケージにもマッチしない (偽)	`~F`

Table 2.11: aptitude の regex 式のリスト

- ”~P<term>” == ”~Dprovides:<term>”
- ”~C<term>” == ”~Dconflicts:<term>”
- ”…~W term” == ”(…|term)”

mutt が表現のお手本なので、mutt に慣れているユーザーはすぐ慣れるでしょう。”User’s Manual” (”/usr/share/doc/apti
中の”SEARCHING, LIMITING, AND EXPRESSIONS” を参照下さい。

注意
lenny バージョンの aptitude(8) では、新規の”?broken” のような長形式の regex マッチ形式が、古い”~b” のような短形式のマッチ形式に代えて使えます。そのためチルダ文字”~” に加えてスペース文字” ” も regex の終端文字として扱われます。新規の長形式のマッチ形式については”User’s Manual” を参照下さい。

2.2.8 aptitude による依存関係の解決

aptitude によるパッケージの選択は、”F10 → Options → Preferences → Dependency handling” のメニュー設定に従って、”Depends:” リストに規定されたパッケージばかりでは無く”Recommends:” リストに規定されたパッケージも引き込みます。このような自動的にインストールされたパッケージは不要になると aptitude が自動的に削除します。

aptitude コマンドの” 自動インストール” 挙動を制御するフラグは apt パッケージ中の apt-mark(8) コマンドを用いても操作できます。

2.2.9 パッケージアクティビティーログ

パッケージアクティビティーの履歴はログファイルで確認できます。

ファイル	内容
/var/log/dpkg.log	全パッケージアクティビティの dpkg レベルのアクティビティーのログ
/var/log/apt/term.log	APT アクティビティのログ
/var/log/aptitude	aptitude コマンドアクティビティのログ

Table 2.12: パッケージアクティビティーのログファイル

これらのログから意味のある理解を迅速に得るのは実際には難しいです。より簡単な方法については項9.2.10を参照下さい。

2.3 aptitude 操作例

aptitude(8) 操作例を次に示します。

2.3.1 regex にマッチするパッケージ名のパッケージをリスト

次のコマンドはパッケージの名前が regex にマッチするパッケージをリストします。

```
$ aptitude search '~n(pam|nss).*ldap'
p libnss-ldap - NSS module for using LDAP as a naming service
p libpam-ldap - Pluggable Authentication Module allowing LDAP interfaces
```

これはパッケージの正確な名前を探すときに非常に便利です。

2.3.2 regex マッチをしての閲覧

” 平坦なパッケージリスト” のビューで”l” のプロンプトに regex ”~dipv6” を入れるとその意味にマッチするパッケージにビューが制限され、その情報をインタラクティブに閲覧できます。

2.3.3 パッケージの完全削除

削除したパッケージが残した全ての設定ファイルを次のようにして完全削除できます。

次のコマンドの結果をチェックします。

```
# aptitude search '~c'
```

もしリストされたパッケージが完全削除されても問題ないなら、次のコマンドを実行します。

```
# aptitude purge '~c'
```

同様のことをインタラクティブにすればよりきめの細かい結果が得られます。

” 新規パッケージ画面” のビューで”l” のプロンプトに regex ”~c” を入れると regex にマッチする” 削除されたが完全駆除されていない” パッケージにビューが制限されます。トップレベルの見出しの上で”[” を押すと regex にマッチする全てのパッケージが表示されます。

次に” インストールされていないパッケージ” 等のトップレベルの見出しの上で”_” を押します。その見出しの下の regex にマッチするパッケージだけが完全削除と設定されます。インタラクティブに個々のパッケージの上で”=” を押せばそれらのパッケージを完全削除対象から外せます。

このテクニックは非常に便利で、他の多くのコマンドキーでも使えます。

2.3.4 自動 / 手動インストール状態の整理

(非 aptitude のパッケージインストーラー等を使った後で) パッケージの自動 / 手動インストールの状態を整理する私の方法を次に記します。

1. aptitude を root としてインタラクティブに起動します。
2. ”u” と”U” と”f” と”g” とタイプしてパッケージリストを更新しパッケージをアップグレードします。
3. パッケージ表示制限を”~i(~R~i|~Rrecommends:~i)” と入力するために”l” とタイプし、自動インストールとなるよう”M” と” インストール済みのパッケージ” の上でタイプします。
4. パッケージ表示制限を”~required|~pimportant|~pstandard|~E” と入力するために”l” とタイプし、手動インストールとなるよう”m” と” インストール済みのパッケージ” の上でタイプします。
5. パッケージ表示制限を”~i!~M” と入力するために”l” とタイプし、” インストール済みのパッケージ” の上で”[” とタイプしてパッケージを見えるようにした後で個々のパッケージの上で”-” とタイプして使っていないパッケージを削除します。
6. パッケージ表示制限を”~i” と入力するように”l” とタイプし、そして” タスク” の上で手動インストールとなるよう”m” とタイプします。
7. aptitude を終了します。
8. ”apt-get -s autoremove|less” と root から起動して何が使われていないのか確認します。
9. aptitude とインタラクティブモードで再起動して必要なパッケージを”m” でマークします。
10. ”apt-get -s autoremove|less” と root から再起動して削除対象が期待にかなっていることを再確認します。
11. ”apt-get autoremove|less” と root から起動して使用していないパッケージを自動削除します。

”Tasks” の上で”m” を押すのも一案で、大量ファイル除去となる事態が回避できます。

2.3.5 システム全体のアップグレード

注意
新規リリース等への移行は、Debian では下記のようにアップグレードできるのですが、新たなシステムをクリーンインストールすることを考えるべきです。こうすると溜めてきたゴミの除去ができる上に最新のパッケージの最良の組み合わせも分かります。もちろん安全な場所に完全なシステムのバックアップ (項10.2参照下さい) を事前にしなくてはいけません。異なったパーティションを使ったデュアルブート設定をすることをスムーズな移行をするためにお薦めします。

”/etc/apt/sources.list” ファイルの内容を新規リリースへと向けるように変更し、”apt update; apt dist-upgrade” コマンドを実行することでシステム全体のアップグレードができます。

安定版 stable からテスト版 testing や不安定版 unstable にアップグレードするには、項2.1.4にある”/etc/apt/sour 例の”bullseye” を”bookworm” か”sid” に置き換えます。

一部のパッケージで移行に関して支障をきたすことが実際には起こるかもしれません。これは大体パッケージ依存関係に起因します。アップグレードする差が大きければ大きいほど比較的大きな問題似合う可能性がより大きくなります。以前の安定版 stable からリリース後の新規安定版 stable への移行では新規リリースノートを読んでそこに記載された手続き通りに完全にすれば問題発生を防げます。

安定版 stable からテスト版 testing へ移行すると決めた時には頼りにするリリースノートはありません。前回の安定版 stable のリリースの後で安定版 stable とテスト版 testing の差がかなり大きくなっているかもしれません。そうだとアップグレードをする状況は複雑になっています。

メーリングリストから最新情報を収集するとか常識を使うといった予防措置をしながらフルアップグレードをするべきです。

1. 前回の” リリースノート” を読みます。
2. 全システム (特にデーターや設定情報) をバックアップします。
3. ブートローダーが壊れたときのためにブートできるメディアを確保します。
4. システムを使っているユーザーに十分事前に通告します。
5. script(1) を使ってアップグレード活動を記録します。
6. 削除をされないように”aptitude unmarkauto vim” 等として、”unmarkauto” を重要なパッケージに適用します。
7. デスクトップタスクにあるパッケージ等を削除して、インストールされたパッケージを減らしてパッケージがコンフリクトする可能性を減らします。
8. ”/etc/apt/preferences” ファイルを削除します (apt-pinning を無効化)。
9. 段階的にアップグレードしましょう: 旧安定版 oldstable → 安定版 stable → テスト版 testing → 不安定版 unstable。
10. ”/etc/apt/sources.list” ファイルを更新して新アーカイブ対象に”aptitude update” を実行します。
11. ”aptitude install perl” 等として、先に新規の中核的パッケージを必要に応じてインストールします。
12. ”apt-get -s dist-upgrade” コマンドを実行して影響を確認します。
13. 最後に”apt-get dist-upgrade” コマンドを実行します。

注意
stable リリース間でアップグレードする際に Debian のメジャーリリースを飛ばすのは賢明ではありません。

注意
過去の” リリースノート” ではシステム全体のアップグレードをするのに GCC や Linux カーネルや initrd-tools や Glibc や Perl や APT tool chain 等には特別な配慮が必要でした。

`unstable` での毎日のアップグレードは項2.4.3を参照下さい。

2.4 高度なパッケージ管理操作

2.4.1 コマンドラインによる高度なパッケージ管理操作

`aptitude` ではハイレベル過ぎるとか必要な機能を欠くという他のパッケージ管理操作のリストです。

注意
multi-arch 機能のあるパッケージに関して、一部のコマンドはアーキテクチャー名を必要があるかもしれません。例えば、`amd64` アーキテクチャーの `libglib2.0-0` パッケージの内容をリストするには”`dpkg -L libglib2.0-0:amd64`” を使います。

注意
”`dpkg -i …`” や”`debi …`” といった低いレベルのパッケージツールはシステム管理者によって注意深く使われなければいけません。必要なパッケージ依存関係を自動的に面倒見てくれません。Dpkg の”`--force-all`” や類似のコマンドラインオプション (dpkg(1) 参照下さい) はエキスパートだけが使うようにできています。十分にその影響を理解せずに使うとシステム全体を壊してしまうかもしれません。

以下に注意下さい。

- 全てのシステム設定やインストールコマンドは root から実行なければいけません。
- regex (項1.6.2参照下さい) を使う `aptitude` と異なり、他のパッケージ管理コマンドはシェルグロブ (項1.5.6参照下さい) のようなパターンを使います。
- `apt-file` パッケージに入っている apt-file(1) は事前に”`apt-file update`” を実行する必要があります。
- `configure-debian` パッケージに入っている configure-debian(8) はそのバックエンドとして dpkg-reconfigure(8) を実行します。
- dpkg-reconfigure(8) はそのバックエンドとして debconf(1) を利用するパッケージスクリプトを実行します。
- ”`apt-get build-dep`” や”`apt-get source`” や”`apt-cache showsrc`” コマンドは”`/etc/apt/sources.list`” の中に”`deb-src`” エントリーが必要です。
- dget(1) や debuild(1) や debi(1) は `devscripts` パッケージが必要です。
- ”`apt-get source`” を使った (再) パッケージ化の手続きは項2.7.13を参照下さい。
- `make-kpkg` コマンドは `kernel-package` パッケージが必要です (項9.9参照下さい)。
- 一般的なパッケージ化に関しては項12.11を参照下さい。

コマンド	アクション
COLUMNS=120 dpkg -l < パッケージ名パターン >	バグレポートのためにインストールされたパッケージの状態をリスト
dpkg -L < パッケージ名 >	インストールされたパッケージの内容をリスト
dpkg -L < パッケージ名 > | egrep '/usr/share/man/man.*/.+'	インストールされたパッケージのマンページをリスト
dpkg -S < ファイル名パターン >	マッチするファイル名があるインストールされたパッケージをリスト
apt-file search < ファイル名パターン >	マッチするファイル名があるアーカイブ中のパッケージをリスト
apt-file list < パッケージ名パターン >	アーカイブ中のマッチするパッケージをリスト
dpkg-reconfigure < パッケージ名 >	特定パッケージを再設定
dpkg-reconfigure -p=low < パッケージ名 >	もっとも詳細な質問で特定パッケージを再設定
configure-debian	フルスクリーンメニューからパッケージを再設定
dpkg --audit	部分的にインストールされたパッケージに関してシステムを監査
dpkg --configure -a	全ての部分的にインストールされたパッケージを設定
apt-cache policy < バイナリーパッケージ名 >	バイナリーパッケージに関して利用可能なバージョンやプライオリティーやアーカイブ情報を表示
apt-cache madison < パッケージ名 >	パッケージに関して利用可能なバージョンやアーカイブ情報を表示
apt-cache showsrc < バイナリーパッケージ名 >	バイナリーパッケージに関してソースパッケージの情報を表示
apt-get build-dep < パッケージ名 >	パッケージをビルドするのに必要なパッケージをインストール
aptitude build-dep <package_name>	パッケージをビルドするのに必要なパッケージをインストール
apt-get source < パッケージ名 >	(標準アーカイブから) ソースをダウンロード
dget <dsc ファイルの URL>	(他のアーカイブから) ソースをダウンロード
dpkg-source -x < パッケージ名 >_< バージョン >-<debian バージョン >.dsc	ソースパッケージの組 (”*.tar.gz” と”*.debian.tar.gz” / ”*.diff.gz”) からソースツリーをビルド
debuild binary	ローカルのソースツリーからパッケージをビルド
make-kpkg kernel_image	カーネルソースツリーからカーネルパッケージをビルド
make-kpkg --initrd kernel_image	カーネルソースツリーから initramfs を有効にしてカーネルパッケージをビルド
dpkg -i < パッケージ名 >_< バージョン >-<debian_ バージョン >_< アーキテクチャー名 >.deb	ローカルパッケージをシステムにインストール
apt install /path/to/<package_filename>.deb	自動的に依存関係を解決しながらローカルパッケージをシステムにインストールする
debi < パッケージ名 >_< バージョン >-<debian_ バージョン >_< アーキテクチャー名 >.dsc	ローカルパッケージ (複数) をシステムにインストール
dpkg --get-selections '*' >selection.txt	dpkg レベルのパッケージ選択状態情報を保存
dpkg --set-selections <selection.txt	dpkg レベルのパッケージ選択状態情報を設定
echo <package_name> hold | dpkg --set-selections	特定パッケージの dpkg レベルのパッケージ選択状態を **hold** にする (”aptitude hold <package_name>” と等価)

Table 2.13: 高度なパッケージ管理操作

2.4.2 インストールされたパッケージファイルの検証

`debsums` をインストールすると debsums(1) を使って”`/var/lib/dpkg/info/*.md5sums`” ファイル中の MD5sum 値との比較でインストールされたパッケージファイルを検証できます。MD5sum がどのような仕組かは項10.3.5参照下さい。

注意
侵入者によって MD5sum のデーターベースが改竄されているかもしれないので debsums(1) はセキュリティーツールとしては限定的有用性しかありません。管理者によるローカルの変更や記憶メディアのエラーによる損傷を点検するぐらいには有用です。

2.4.3 パッケージ問題からの防御

多くのユーザーは新規機能やパッケージを求めて Debian システムの非安定版 **unstable** リリースを追いかけることを好みます。こういうことをするとクリティカルなパッケージのバグにシステムが遭遇しやすくなります。

`apt-listbugs` パッケージをインストールすれば、APT システムを使ってアップグレードする時に Debian の BTS を自動的にクリティカルなバグに関して点検することで、クリティカルなバグからあなたのシステムを防御できます。

`apt-listchanges` パッケージをインストールすれば､APT システムを使ってアップグレードする時に `NEWS.Debian` 中の重要ニュースを表示します。

2.4.4 パッケージメタデーターの検索

最近は Debian サイトの https://packages.debian.org/ を訪問するとパッケージメタデーターの検索を簡単に出きるようになっていますが、より伝統的な方法を見てみます。

grep-dctrl(1) や grep-status(1) や grep-available(1) コマンドは Debian のパッケージコントロールファイルの一般的フォーマットに従ういかなるファイルを検索するのにも使えます。

マッチする名前のファイルを含む `dpkg` でインストールされたパッケージ名を探索するのに”`dpkg -S < ファイル名パターン >`” が使えます。しかしメンテナスクリプトで生成されるファイルはこれでは見逃されます。

dpkg のメタデーターに関してより詳細な検索をする必要がある場合、”`/var/lib/dpkg/info/`” ディレクトリーで”`grep -e regex パターン *`” コマンドを実行しないといけません。こうすることでパッケージスクリプトやインストール時の質問テキスト中の言葉まで検索できます。

パッケージ依存関係を再帰的に検索したい際には、apt-rdepends(8) を使います。

2.5 Debian パッケージ管理の内部

Debian のパッケージ管理システムが内部的のどのように機能するのかを学びます。何らかのパッケージ問題が発生した際にあなた自身の解決を見出すのに役立つでしょう。

2.5.1 アーカイブのメタデーター

各ディストリビューションのメタデーターのファイルは例えば”`http://deb.debian.org/debian/`” のような各 Debian ミラーサイトの”`dist/< コード名 >`” の下に保存されています。そのアーカイブ構造はウェッブブラウザーで閲覧できます。6 つのタイプの重要メタデーターがあります。

最近のアーカイブではネットワークトラフィックを減らすべく圧縮された差分ファイルとしてこれらのメタデーターは保存されています。

ファイル	場所	内容
`Release`	ディストリビューションのトップ	アーカイブの説明との整合性情報
`Release.gpg`	ディストリビューションのトップ	アーカイブキーで署名された”`Release`” ファイルに関する署名ファイル
`Contents-< アーキテクチャー >`	ディストリビューションのトップ	該当アーカイブ中全てのパッケージに関する全ファイルリスト
`Release`	各ディストリビューション / エリア / アーキテクチャーの組み合わせのトップ	apt_preferences(5) のルールに利用されるアーカイブの記述。
`Packages`	各ディストリビューション / エリア / バイナリーアーキテクチャーの組み合わせのトップ	バイナリーパッケージに関して `debian/control` を連結
`Sources`	各ディストリビューション / エリア / ソースの組み合わせのトップ	ソースパッケージに関して `debian/control` を連結

Table 2.14: Debian アーカイブのメタデーターの内容

2.5.2 トップレベルの”Release” ファイルと信憑性

ティップ
セキュアー **APT** システムではトップレベルの”Release” ファイルがアーカイブを署名するのに使われています。

Debian アーカイブの各スイーツには例えば次に示すような”`http://deb.debian.org/debian/dists/unstable/Re`
のようなトップレベルの”Release” ファイルがあります。

```
Origin: Debian
Label: Debian
Suite: unstable
Codename: sid
Date: Sat, 14 May 2011 08:20:50 UTC
Valid-Until: Sat, 21 May 2011 08:20:50 UTC
Architectures: alpha amd64 armel hppa hurd-i386 i386 ia64 kfreebsd-amd64 kfreebsd-i386 mips ↩
     mipsel powerpc s390 sparc
Components: main contrib non-free
Description: Debian x.y Unstable - Not Released
MD5Sum:
 bdc8fa4b3f5e4a715dd0d56d176fc789 18876880 Contents-alpha.gz
 9469a03c94b85e010d116aeeab9614c0 19441880 Contents-amd64.gz
 3d68e206d7faa3aded660dc0996054fe 19203165 Contents-armel.gz
...
```

注意
項2.1.4の中で” スイーツ (suite)” や” コード名 (codename)” を使う理由はこれを見れば分かるでしょう。” ディストリビューション” は” スイーツ” と” コード名” との両方を指したい際に用いられます。アーカイブが提供する全アーカイブ” エリア (area)” 名が”Components” の下にリストされます。

トップレベルの”`Release`” ファイルの整合性は セキュアー apt という暗号学手法インフラストラクチャーによって検証されます。

- 暗号手法による署名ファイル”`Release.gpg`” は真正のトップレベルの”`Release`” ファイルと秘密の Debian アーカイブキーから作成されます。

- 公開の Debian アーカイブキーは”/etc/apt/trusted.gpg” に取り込むには次のようにします。
  - 最新の base-files パッケージを用いてキーリングをインストールすることで自動で取り込む。
  - ftp-master.debian.org に掲示された最新の公開アーカイブキーを gpg か apt-key ツールを用いて手動で取り込む。
- セキュアー **APT** システムはこの”Release.gpg” ファイルと”/etc/apt/trusted.gpg” 中の公開アーカイブキーを用いてダウンロードされたトップレベルの”Release” ファイルの整合性を暗号学手法を用いて検証します。

” 全ての Packages” と”Sources ファイルの整合性はそのトップレベルの”Release” ファイル中の MD5sum 値を用いて検証します。” パッケージファイルの整合性は”Packages” や”Sources” ファイル中の MD5sum 値を用いて検証します。debsums(1) と項2.4.2を参照下さい。

暗号学手法を用いた署名の検証は MD5sum 値の計算よりも非常に CPU を使うプロセスなので、トップレベルの”Release” ファイルには暗号学手法を用いた署名を使いつつ各パッケージには MD5sum 値を用いることでパーフォーマンスを保ったまま良好なセキュリティーが確保できます (項10.3参照下さい)。

2.5.3 アーカイブレベルの”Release” ファイル

ティップ
アーカイブレベルの”Release” ファイルが apt_preferences(5) のルールに使われます。

”http://deb.debian.org/debian/dists/unstable/main/binary-amd64/Release” や”http://deb.debian 等の”/etc/apt/sources.list” 中の”deb” 行で特定される全てのアーカイブロケーションにはアーカイブレベルの次に示すような”Release” ファイルがあります。

```
Archive: unstable
Origin: Debian
Label: Debian
Component: main
Architecture: amd64
```

注意
”Archive:” スタンザには、Debian アーカイブではスイート名 (”stable” や”testing” や”unstable” 等) が使われますが、Ubuntu アーカイブではコード名 (”trusty” や”xenial” や”artful” 等) が使われます。

---

experimental や bullseye-backports のような自動的にインストールされるべきでないパッケージを含むような一部アーカイブでは次に示す”http://deb.debian.org/debian/dists/experimental/main/binary-amd64 のような追加の行があります。

```
Archive: experimental
Origin: Debian
Label: Debian
NotAutomatic: yes
Component: main
Architecture: amd64
```

”NotAutomatic: yes” となっていない通常のアーカイブではデフォールトの Pin-Priority 値は 500 ですが、”NotAutomatic: yes” となっている特別なアーカイブではデフォールトの Pin-Priority 値は 1 です (apt_preferences(5) と項2.7.3参照下さい)。

2.5.4 パッケージメタデーターの取得

aptitude や apt-get や synaptic や apt-file や auto-apt 等の APT ツールが使われる際には Debian アーカイブ情報を含むメタデーターのローカルコピーを更新する必要があります。この様なローカルのコピーは”/etc/apt/sources.list” 中のディストリビューション (distribution) とエリア (area) とアーキテクチャー (architecture) の名前に対応する次のファイル名です (項2.1.4参照下さい)。

- ”/var/lib/apt/lists/deb.debian.org_debian_dists_< ディストリビューション >_Release”
- ”/var/lib/apt/lists/deb.debian.org_debian_dists_< ディストリビューション >_Release.gpg”
- ”/var/lib/apt/lists/deb.debian.org_debian_dists_< ディストリビューション >_< エリア >_source_S
- ”/var/lib/apt/lists/deb.debian.org_debian_dists_< ディストリビューション >_< エリア >_source_S
- ”/var/cache/apt/apt-file/deb.debian.org_debian_dists_< ディストリビューション >_Contents-< アーキテクチャー >.gz” (apt-file 用)

最初の 4 つのタイプのファイルは全ての適切な APT コマンド間で共有されておりコマンドラインから”apt-get update” や”aptitude update” によって更新されます。もし”/etc/apt/sources.list” 中に”deb” 行があれば”Packages” メタデーターが更新されます。もし”/etc/apt/sources.list” 中に”deb-src” 行があれば”Sources” メタデーターが更新されます。

”Packages” や”Sources” メタデーターはバイナリーやソースパッケージのファイルの場所を指している”Filename:” スタンザを含んでいます。現在、それらのパッケージはリリース間の移行を滞り無くするために”pool/” ディレクトリーツリーの下に置かれています。

”Packages” メタデーターのローカルコピーは aptitude を使ってインタラクティブに検索できます。grep-dctrl(1) という専用の検索コマンドを使うと”Packages” と”Sources” メタデーターのローカルコピーを検索できます。

”Contents-< アーキテクチャー >” メタデーターのローカルコピーは”apt-file update” で更新でき、他の 4 つと異なるところにあります。apt-file(1) を参照下さい。(auto-apt では”Contents-< アーキテクチャー >.gz” のローカルコピーがデフォールトでは異なるところにあります。)

2.5.5 APT に関するパッケージ状態

lenny 以降の APT ツールではリモートから取得したメタデーターに追加でローカルで生成されるインストール状態情報を”/var/lib/apt/extended_states” に保存して、自動インストールされた全パッケージを全ての APT ツールで追跡するのに用います。

2.5.6 aptitude に関するパッケージ状態

aptitude コマンドではリモートから取得したメタデーターに追加でローカルで生成されるインストール状態情報を”/var/lib/aptitude/pkgstates” に保存して用いています。

2.5.7 取得したパッケージのローカルコピー

APT メカニズムでリモートから取得されたパッケージは消去されるまでは”/var/cache/apt/archives” に貯蔵されます。

aptitude では、このキャッシュファイルのクリーニングポリシーは”Options” → ”Preferences” の下で設定でき、”Actions” の下の”Clean package cache” か”Clean obsolete files” メニューによって強制実行できる。

パッケージタイプ	名前の構造
バイナリーパッケージ (所謂 deb)	<package-name>_<upstream-version>-<debian-version>_<arc
Debian インストーラー用のバイナリーパッケージ (所謂 udeb)	<package-name>_<upstream-version>-<debian-version>_<arc
ソースパッケージ (アップストリームのソース)	<package-name>_<upstream-version>-<debian-version>.orig
1.0 ソースパッケージ (Debian の変更部分)	<package-name>_<upstream-version>-<debian-version>.diff
3.0 (quilt) ソースパッケージ (Debian の変更部分)	<package-name>_<upstream-version>-<debian-version>.debi
ソースパッケージ (内容記述)	<package-name>_<upstream-version>-<debian-version>.dsc

Table 2.15: Debian パッケージの名前の構造

2.5.8 Debian パッケージファイル名

Debian のパッケージファイルには特定の名前の構造があります。

ティップ
ここでは基本的なパッケージフォーマットのみが記述されています。詳細は dpkg-source(1) を参照下さい。

名前の部分	使用可能文字 **(regex)**	存在
< パッケージ名 >	[a-z,A-Z,0-9,.,+,-]+	必須
< エポック >:	[0-9]+:	任意
< アップストリームのバージョン >	[a-z,A-Z,0-9,.,+,-,:]+	必須
<debian のバージョン >	[a-z,A-Z,0-9,.,+,~]+	任意

Table 2.16: Debian パッケージ名の各部分に使用可能な文字

注意
パッケージバージョンの順位は dpkg(1) を使って、例えば”dpkg --compare-versions 7.0 gt 7.~pre1 ; echo $?” とすると確認できます。

注意
Debian インストーラー (d-i) のバイナリーパッケージには、通常の deb ではなく udeb をファイル拡張子として使われます。udeb パッケージはポリシー条件を緩和しドキュメントのように必須でない内容を削除した減量 deb パッケージです。deb と udeb パッケージは同一のパッケージ構造を共有しています。”u” はマイクロと言う意味で使っています。

2.5.9 dpkg コマンド

dpkg(1) は Debian パッケージ管理の最も低レベルのツールです。非常に強力ですから気をつけて使う必要があります。

”< パッケージ名 >” というパッケージをインストールする際に、dpkg は次に記す順番でパッケージを処理します。

1. deb ファイルを解凍 (”ar -x” と等価)

2. debconf(1) を使い”<package_name>.preinst” を実行
3. システムにパッケージ内容をインストール (”tar -x” と等価)
4. debconf(1) を使い”<package_name>.postinst” を実行

debconf システムによって I18N と L10N (第8章) のサポートのある標準化されたユーザーとの対話が実現できます。

ファイル	内容の説明
/var/lib/dpkg/info/< パッケージ名 >.conffiles	設定ファイルのリスト。(ユーザー変更可能)
/var/lib/dpkg/info/< パッケージ名 >.list	パッケージによりインストールされるファイルやディレクトリーのリスト
/var/lib/dpkg/info/< パッケージ名 >.md5sums	パッケージによりインストールされるファイルの MD5 ハッシュ値のリスト
/var/lib/dpkg/info/< パッケージ名 >.preinst	パッケージインストールの前に実行するパッケージスクリプト
/var/lib/dpkg/info/< パッケージ名 >.postinst	パッケージインストールの後に実行するパッケージスクリプト
/var/lib/dpkg/info/< パッケージ名 >.prerm	パッケージ削除の前に実行するパッケージスクリプト
/var/lib/dpkg/info/< パッケージ名 >.prerm	パッケージ削除の前に実行するパッケージスクリプト
/var/lib/dpkg/info/< パッケージ名 >.conffiles	debconf システムのためのパッケージスクリプト
/var/lib/dpkg/alternatives/< パッケージ名 >	update-alternatives コマンドが用いる代替情報
/var/lib/dpkg/available	すべてのパッケージの入手可能性情報
/var/lib/dpkg/diversions	dpkg(1) が利用し、dpkg-divert(8) が設定するする迂回情報
/var/lib/dpkg/statoverride	dpkg(1) が利用し、dpkg-statoverride(8) が設定する状態オーバーライド情報
/var/lib/dpkg/status	全パッケージに関する状態情報
/var/lib/dpkg/status-old	”var/lib/dpkg/status” ファイルの第一世代のバックアップ
/var/backups/dpkg.status*	”var/lib/dpkg/status” ファイルの第二世代以前のバックアップ

Table 2.17: dpkg が作成する特記すべきファイル

”status” ファイルは dpkg(1) や”dselect update” や”apt-get -u dselect-upgrade” のようなツールによって使われます。

grep-dctrl(1) という専用の検索コマンドを使うと”status” と”available” メタデーターのローカルコピーを検索できます。

ティップ
デビアンインストーラー 環境下では、udpkg コマンドが udeb パッケージを開けるのに用いられます。udpkg コマンドはストリップダウンされたバージョンの dpkg コマンドです。

2.5.10 update-alternative コマンド

Debian システムには update-alternatives(8) を用いて何らかの重畳するプログラムを平和裏にインストールするメカニズムがあります。例えば vim と nvi の両方のパッケージがインストールされた状況下で vi コマンドが vim を選択して実行するようにできます。

```
$ ls -l $(type -p vi)
lrwxrwxrwx 1 root root 20 2007-03-24 19:05 /usr/bin/vi -> /etc/alternatives/vi
$ sudo update-alternatives --display vi
...
$ sudo update-alternatives --config vi
  Selection    Command
 -----------------------------------------------
      1        /usr/bin/vim
*+    2        /usr/bin/nvi

Enter to keep the default[*], or type selection number: 1
```

Debian の代替 (alternatives) システムは、その選択を”`/etc/alternatives/`” の中のシムリンクとして保持します。選択プロセスには”`/var/lib/dpkg/alternatives/`” の中の対応するファイルが使われます。

2.5.11 dpkg-statoverride コマンド

dpkg-statoverride(8) コマンドで提供される状態の上書きは、パッケージをインストールする際にファイルに関して異なる所有者やモードを使うよう dpkg(1) に指示する方法です。もし”`--update`” が指定されファイルが存在すれば、即座に新たな所有者やモードに設定されます。

注意
パッケージが所有するファイルの所有者やモードをシステム管理者が `chmod` や `chown` コマンドを用いて直接変更しても次のパッケージアップグレードがリセットします。

注意
ここでファイルと言いましたが、実際には `dpkg` が扱うディレクトリーやデバイス等のいかなるファイルシステムオブジェクトであってもいいです。

2.5.12 dpkg-divert コマンド

dpkg-divert(8) コマンドによって提供されるファイル迂回は、ファイルをデフォールトの場所ではなく迂回した場所にインストールするように dpkg(1) にさせます。`dpkg-divert` は本来パッケージメインテナンススクリプトのためのものです。システム管理者がこれを軽々に使うのはお薦めできません。

2.6 壊れたシステムからの復元

非安定 (`unstable`) システムを動かす時には、管理者には壊れたパッケージ管理状況から復元できることが望まれます。

注意
ここで説明するいくつかの方法は非常にリスクが高いアクションです。警告しましたよ!

2.6.1 古いユーザーの設定との非互換性

もしデスクトップ GUI プログラムが上流の大きなバージョンアップグレードの後に不安定性を経験した際には、そのプログラムが作った古いローカル設定ファイルとの干渉を疑うべきです。もし新規作成したユーザーアカウントでそのプログラムが安定なら、この仮説が裏付けられます。(これはパッケージングのバグで、通常パッケージャーによって回避されます。)

安定性を復元するには、対応するローカル設定ファイルを移動し GUI プログラムを再スタートします。後日設定情報を回復するために古い設定ファイルの内容を読む必要があるかもしれません。(あまり慌てて消去しないようにしましょう。)

2.6.2 重複するファイルを持つ相異なるパッケージ

aptitude(8) や apt-get(1) 等の、アーカイブレベルのパッケージ管理システムはパッケージの依存関係を使って重複するファイルを持つファイルのインストールしようとさえしません (項2.1.6参照下さい)。

パッケージメインテナによるエラーや、システム管理者による不整合な混合ソースのアーカイブの採用 (項2.7.2参照下さい) があった場合には、パッケージ依存関係が誤って定義される事態が発生するかもしれません。そういう状況下で重複するファイルを持つパッケージを aptitude(8) や apt-get(1) を使ってインストールしようとすると、パッケージを展開する dpkg(1) は既存ファイルを上書きすることなく呼ばれたプログラムにエラーを確実に返します。

注意
第三者が作成したパッケージを使うと、root 権限で実行されるシステムに関して何でもできるメンテナスクリプトが実行されるので、システムが重大なリスクにさらされます。dpkg(1) はパッケージを展開するするさいに上書きする事を防止するだけです。

そのような壊れたインストール状況は、まず古い問題原因となっているパッケージ <old-package> を削除すれば回避できます。

```
$ sudo dpkg -P <old-package>
```

2.6.3 壊れたパッケージスクリプトの修正

パッケージスクリプト内のコマンドが何らかの理由でエラーを返しスクリプトがエラーで終了した場合には、パッケージ管理システムは動作を途中終了するので部分的にインストールされたパッケージのある状況が生まれます。パッケージがその削除スクリプト内にバグを持つ場合には、パッケージが削除不能になりうるので大変厄介です。

”< パッケージ名 >” のパッケージスクリプトの問題に関しては、次のパッケージスクリプトの内容を確認するべきです。

- ”/var/lib/dpkg/info/< パッケージ名 >.preinst”
- ”/var/lib/dpkg/info/< パッケージ名 >.postinst”
- /var/lib/dpkg/info/< パッケージ名 >.prerm
- ”/var/lib/dpkg/info/< パッケージ名 >.prerm”

スクリプトの問題原因部分を次のようなテクニックを使い root から編集します。

- 行頭に”#” を挿入し問題行を無効にする
- 行末に”|| true” を挿入し成功を強制的に返さす

全ての部分的にインストールされたパッケージを次のコマンドで設定します。

```
# dpkg --configure -a
```

2.6.4 dpkg コマンドを使っての救済

dpkg は非常に低レベルのパッケージツールなのでネットワーク接続もないブート不能な非常に劣悪な状況下でも機能します。foo パッケージが壊れていて置き換える必要があると仮定します。

バグの無い古いバージョンの foo パッケージが"/var/cache/apt/archives/" にあるパッケージキャッシュの中に見つかるかもしれません。(ここにみつからなければ、https://snapshot.debian.org/ アーカイブからダウンロードしたり、機能している機器のパッケージキャッシュからコピーできます。)

もしブート不可能な場合には、次のコマンドを使ってインストールすることもできます。

```
# dpkg -i /path/to/foo_<old_version>_<arch>.deb
```

ティップ
システムがそれほど壊れていないなら、項2.7.10に書かれているようにして、より高レベルの APT システムを通じてシステム全体をダウングレードする手もあります。

ハードディスクからブートできない場合は、他の方法でのブート方法を考えるべきです。

1. Debian インストーラー (debian-installer) の CD を使ってレスキューモードでブートします。
2. ブートできないハードディスク上のシステムを"/target" にマウントします。
3. 古いバージョンの foo パッケージを次のようにしてインストールします。

```
# dpkg --root /target -i /path/to/foo_<old_version>_<arch>.deb
```

この例は、たとえハードディスク上の dpkg コマンドが壊れていても機能します。

ティップ
ハードディスク上の別のシステムであれ、GNU/Linux のライブ CD であれ、ブート可能な USB キードライブであれ、ネットブートであれ、どのように起動された GNU/Linux システムでも同様にして壊れたシステムを救済するのに使えます。

もしこの方法でパッケージをインストールしようとして何らかの依存関係違反のためにうまくいかなくてどうしようもなくなった場合には、dpkg の"--ignore-depends" や"--force-depends" や他のオプションを使って依存関係をオーバーライドすることができます。こうした場合には、後で適正な依存関係を修復するように真剣に取り組む必要があります。詳細は dpkg(8) を参照下さい。

注意
システムがひどく壊れた場合には、システムを安全な場所に完全バックアップし (項10.2参照下さい)、クリーンインストールを実行するべきです。こうすることは時間の節約でもあり最終的に良い結果に結びつきます。

2.6.5 パッケージセレクションの復元

もし何らかの理由で"/var/lib/dpkg/status" の内容が腐った場合には、Debian システムはパッケージ選択データーが失われ大きな打撃を被ります。古い"/var/lib/dpkg/status" ファイルは、"/var/lib/dpkg/status-old" や"/var/backups/dpkg.status.*" としてあるので探します。

"/var/backups/" は多くの重要な情報を保持しているので、これを別のパーティション上に置くのも良い考えです。

ひどく壊れた場合には、システムのバックアップをした後フレッシュに再インストールすることをお薦めします。たとえ"/var/" ディレクトリーの中が完全に消去されても、"/usr/share/doc/" ディレクトリー中から新規インストールのガイドとなる情報を復元できます。

最低限の (デスクトップ) システムを再インストールします。

```
# mkdir -p /path/to/old/system
```

”/path/to/old/system/” に古いシステムをマウントします。

```
# cd /path/to/old/system/usr/share/doc
# ls -1 >~/ls1.txt
# cd /usr/share/doc
# ls -1 >>~/ls1.txt
# cd
# sort ls1.txt | uniq | less
```

こうすると、インストールすべきパッケージ名が表示されます。(”texmf” のようなパッケージ名以外が一部あるかもしれません。)

2.7 パッケージ管理のヒント

2.7.1 Debian パッケージの選択方法

パッケージの説明や”Tasks” の下のリストを使ってあなたが必要なパッケージを aptitude で見つけることができます。

2 つ以上の似たパッケージに出会い” 試行錯誤” の努力無しにどのパッケージをインストールするか迷った際には、常識を使って下さい。次に示す点は好ましいパッケージの良い指標と考えます。

- 必須 (essential): yes > no
- エリア (area): メイン (main) > contrib > non-free
- 優先度 (priority): 必須 (required) > 重要 (important) > 標準 (standard) > 任意 (optional) > 特別 (extra)
- タスク (tasks): ” デスクトップ環境” のようなタスクにリストされたパッケージ
- 依存パッケージにより選ばれたパッケージ (例えば、python による python2.4)
- ポプコン: 投票やインストールの数が多い
- changelog: メンテナによる定期的アップデート
- BTS: RC bug が無いこと (critical も grave も serious もいずれのバグも無い)
- BTS: バグレポートに反応の良いメンテナ
- BTS: 最近修正されたバグの数が多い
- BTS: wishlist 以外のバグが少ない

Debian は分散型の開発モデルのボランティアプロジェクトですので、そのアーカイブには目指すところや品質の異なる多くのパッケージがあります。これらをどうするかは自己判断をして下さい。

2.7.2 混合したアーカイブソースからのパッケージ

注意
安定版 (stable) と security updates と bullseye-updates のような公式にサポートされた特定の組み合わせ以外は、混合したアーカイブソースからのパッケージをインストールすることを、公式には Debian ディストリビューションとしてサポートしていません。

testing を追跡しながら、unstable にある特定の新規アップストリームバージョンのパッケージを 1 回だけ取り入れる操作例を次に示します。

1. ”/etc/apt/sources.list” ファイルを変更し、単一の”unstable” エントリーのみにします。
2. ”aptitude update” を実行します。
3. ”aptitude install < パッケージ名 >” の実行します。
4. testing のためのオリジナルの”/etc/apt/sources.list” ファイルを復元します。
5. ”aptitude update” を実行します。

この様な手動のアプローチをすると”/etc/apt/preferences” ファイルを作ることもないし、また apt-pinning について悩むこともありません。でもこれではとても面倒です。

注意
混合したアーカイブソースを使うことを Debian が保証していないので、その場合にはパッケージ間の互換性は自分自身で確保しなければいけません。もしパッケージに互換性がないと、システムを壊すことになるかもしれません。この様な技術的要件を判断できる必要があります。ランダムな混合したアーカイブソースを使うことは全く任意の操作ですが、私としてはこの操作はお薦めできません。

異なるアーカイブからパッケージをインストールするための一般ルールは以下です。

- 非バイナリーパッケージのインストールは比較的安全です。
  - 文書パッケージ: 特段の要件無し
  - インタープリタプログラムパッケージ: 互換性あるインタープリタ環境が利用可能
- バイナリーパッケージ (非”Architecture: all”) のインストールは、通常多くの障害があり、安全ではありません。
  - ライブラリー (”libc” 等) のバージョン互換性
  - 関連ユーティリティープログラムのバージョン互換性
  - カーネル ABI 互換性
  - C++ の ABI 互換性
  - …

注意
パッケージを比較的安全にインストールできるようにするために、一部の商用 non-free バイナリープログラムパッケージは完全に静的にリンクされたライブラリーとともに提供される事があります。そんなパッケージに関しても ABI 互換性等の問題は確認するべきです。

注意
壊れたパッケージを短期的に避ける場合以外では、公式にサポートされていないアーカイブからバイナリーパッケージをインストールするのは一般的には賢明ではありません。たとえ apt-pinning (項2.7.3参照下さい) を使った場合にもこれは当てはまります。chroot や類似のテクニック (項9.10参照下さい) 使って、他のアーカイブからのプログラムを実行するよう検討するべきです。

2.7.3 候補バージョンの調整

”/etc/apt/preferences” ファイル無しだと、APT システムはバージョン文字列を用いて、最新利用可能バージョンを候補バージョンとします。これが通常状態で APT システムの最も推薦される使い方です。全ての公式にサポートされたアーカイブの組み合わせは、自動的にアップグレードするソースとすべきでないアーカイブは **NotAutomatic** とマークされ適正な扱いを受けるので、”/etc/apt/preferences” ファイルを必要としません。

ティップ
バージョン文字列比較ルールは、例えば”dpkg --compare-versions ver1.1 gt ver1.1~1; echo $?” とすれば確認できます (dpkg(1) 参照下さい)。

パッケージを混合したアーカイブからのソース (項2.7.2参照下さい) から定常的にインストールする場合には、apt_preferences(5) に書かれたように適正な項目のある”/etc/apt/preferences” ファイルを作り候補バージョンに関するパッケージ選択ルールを操作することによってこういった複雑な操作を自動化できます。これを **apt-pinning** と呼びます。

警告
初心者のユーザーによる apt-pinning の利用は大トラブル発生を間違いなく起こします。本当に必要な時以外は apt-pinning の利用は避けなければいけません。

注意
apt-pinning を利用する際には、Debian はパッケージの互換性を保証しないので、ユーザー自身がパッケージの互換性を確保しなければいけません。apt-pinning は全く任意の操作で、著者が使うようにと勧めているわけではありません。

注意
アーカイブレベルの Release ファイル (項2.5.3参照下さい) が apt_preferences(5) のルールに使われます。だから、apt-pinning は normal Debian archives や security Debian archives ではスイート (”suite”) 名を使って機能します。(これは Ubuntu アーカイブとは異なります)。例えば”/etc/apt/preferences” ファイル中で、”Pin: release a=unstable” とはできますが、”Pin: release a=sid” とはできません。

注意
非 Debian アーカイブを apt-pinning の一部に使う場合には、それが提供されている対象の確認とその信頼性の確認をします。例えば、Ubuntu と Debian は混合して使うようにはなっていません。

注意
”/etc/apt/preferences” ファイルを作成することなしでも、かなり複雑なシステム操作 (項2.6.4と項2.7.2参照下さい) が apt-pinning を使わずにできます。

単純化した **apt-pinning** テクニックの説明を次にします。

APT システムは”/etc/apt/sources.list” ファイル中に規定された利用可能なパッケージソースから最高の Pin-Priority でアップグレードするパッケージを候補バージョンパッケージとして選択します。パッケージの Pin-Priority が 1000 より大きい場合には、このアップグレードするというバージョン制約が外れるのでダウングレードできるようになります (項2.7.10参照下さい)。

各パッケージの Pin-Priority 値は”/etc/apt/preferences” ファイル中の”Pin-Priority” 項目にて規定されるか、そのデフォールト値が使われます。

ターゲットのリリースアーカイブは次のようにして設定できます。

Pin-Priority	パッケージに関する **apt-pinning** 効果
1001	パッケージのダウングレードになる場合でもパッケージをインストールする
990	ターゲットのリリースアーカイブのデフォルトとして使用
500	ノーマルアーカイブのデフォルトとして使用
100	**NotAutomatic** かつ **ButAutomaticUpgrades** アーカイブのデフォルトとして使用
100	インストール済みパッケージに使用
1	**NotAutomatic** アーカイブのデフォルトとして使用
-1	たとえ推奨 (Recommend) されても、パッケージを絶対にインストールしない

Table 2.18: **apt-pinning** テクニックに関する特記すべき Pin-Priority 値をリストします。

- "`APT::Default-Release "stable";`" 行を使う"`/etc/apt/apt.conf`" ファイル
- "`apt-get install -t testing some-package`" 等の"`-t`" オプションの引数

アーカイブ中のアーカイブレベルの Release ファイル (項2.5.3参照下さい) に"`NotAutomatic: yes`" や"`ButAutomaticU
yes`" が含まれると **NotAutomatic** かつ **ButAutomaticUpgrades** アーカイブと設定されます。

複数アーカイブソースの <package> に関する Pin-Priority 値は"`apt-cache policy <package>`" の出力で表示されます。

- "`Package pin:`" で始まる行は、<package> のみとの関連付けが"`Package pin: 0.190`" 等と定義されている場合に、**pin** のパッケージバージョンを示します。
- <package> とのみの関連付けが定義されていない場合には、"`Package pin:`" という行はありません。
- <package> とのみの関連付けが定義されている場合の Pin-Priority 値は、全バージョン文字列の右側に"`0.181 700`" 等としてリストされます。
- <package> とのみの関連付けが定義されていない場合には、全バージョン文字列の右側に"`0`" が"`0.181 0`" 等としてリストされます。
- アーカイブの Pin-Priority 値 ("`/etc/apt/preferences`" ファイル中に"`Package: *`" として定義) はアーカイブへのパスの左側に、"`100 http://deb.debian.org/debian/ bullseye-backports/main Packages`" 等としてリストされます。

2.7.4 Updates と Backports

`stable` のためのアップグレードパッケージを提供する、bullseye-updates と backports.debian.org アーカイブがあります。

これらのアーカイブを使うには、以下に示すように"`/etc/apt/preferences`" ファイル中に全ての必要なアーカイブをリストします。

```
deb http://deb.debian.org/debian/ bullseye main contrib non-free
deb http://security.debian.org/ bullseye/updates main contrib
deb http://deb.debian.org/debian/ bullseye-updates main contrib non-free
deb http://deb.debian.org/debian/ bullseye-backports main contrib non-free
```

"`/etc/apt/preferences`" ファイル中に Pin-Priority 値を明示的に設定する必要はありません。より新しいパッケージが利用可能となった場合はいつも、デフォルトの設定によりもっとも合理的なアップグレードがなされます (項2.5.3 参照下さい)。

- 全てのインストールされている古いパッケージが `bullseye-updates` からのより新しいパッケージにアップグレードされます。
- `bullseye-backports` からインストールしされた古いパッケージのみが `bullseye-backports` からのより新しいパッケージにアップグレードされます。

”<package-name>” という名前のパッケージをその依存関係ともども bullseye-backports アーカイブからインストールしたい時には、”-t” オプションでターゲットリリースを切り替えながら次のコマンドを使います。

```
$ sudo apt-get install -t bullseye-backports <package-name>
```

2.7.5 ” 推奨 (Recommends)” によりパッケージがインストールされるのを阻止

たとえ” 推奨 (Recommends)” されていても自動的に特定のパッケージが引きこまれ無くしたいときには、”/etc/apt/pref
ファイルを作成しその中に全てのパッケージを次のように明示的にリストしなければいけません。

```
Package: <package-1>
Pin: version *
Pin-Priority: -1

Package: <package-2>
Pin: version *
Pin-Priority: -1
```

2.7.6 unstable からのパッケージと共に、testing を追いかける

testing を追跡しながら、unstable にある特定の新規アップストリームバージョンのパッケージが定常的にアップグレードされる、**apt-pinning** テクニックの例を次に示します。全ての必要なアーカイブを”/etc/apt/sources.list” ファイル中に次のようにリストします。

```
deb http://deb.debian.org/debian/ testing main contrib non-free
deb http://deb.debian.org/debian/ unstable main contrib non-free
deb http://security.debian.org/ testing/updates main contrib
```

”/etc/apt/preferences” を次のように設定します。

```
Package: *
Pin: release a=unstable
Pin-Priority: 100
```

”<package-name>” という名前のパッケージとその依存ファイルを unstable アーカイブからこの設定の下でインストールしたい場合、”-t” オプションを使ってターゲットのリリースを切り替える (unstable の Pin-Priority が 990 になる) 次のコマンドを実行します。

```
$ sudo apt-get install -t unstable <package-name>
```

この設定では、通常の”apt-get upgrade” や”apt-get dist-upgrade”(”aptitude safe-upgrade” や”aptitude full-upgrade”) の実行は、testing アーカイブからインストールされたパッケージは最新の testing アーカイブを使ってアップグレードし、unstable アーカイブからインストールされたパッケージは最新の unstable アーカイブを使ってアップグレードします。

注意
”/etc/apt/sources.list” ファイルから”testing” の項目を削除しないように注意します。”testing” 項目がその中にないと、APT システムは最新の unstable アーカイブを使ってアップグレードします。

ティップ
著者は上記操作のすぐ後に”/etc/apt/sources.list” ファイルを編集して”unstable” アーカイブ項目をコメントアウトします。こうすることで、最新の unstable アーカイブによって unstable からインストールされたパッケージをアップグレードしなくなりますが、”/etc/apt/sources.list” ファイル中に項目が多すぎてアップデートのプロセスが遅くなることをさけられます。

ティップ
もし”/etc/apt/preferences” ファイル中で”Pin-Priority: 100” の代わりに”Pin-Priority: 1” が用いられた場合は、”/etc/apt/sources.list” ファイルの中の”testing” 項目が削除されようと、Pin-Priority 値は 100 のインストール済みパッケージは unstable アーカイブによってアップグレードされる事はありません。

最初の”-t unstable” によるインストール無しに、unstable の特定パッケージを自動的に追跡したい場合、”/etc/apt/preferences” ファイルを作りそのトップにこれらパッケージを明示的に次のようにリストします。

```
Package: <package-1>
Pin: release a=unstable
Pin-Priority: 700

Package: <package-2>
Pin: release a=unstable
Pin-Priority: 700
```

以上で、各特定パッケージに関して Pin-Priority 値が設定されます。例えば最新の unstable バージョンのこの”Debian リファレンス” を英語版で追跡するためには、”/etc/apt/preferences” ファイルに次の項目を設定します。

```
Package: debian-reference-en
Pin: release a=unstable
Pin-Priority: 700

Package: debian-reference-common
Pin: release a=unstable
Pin-Priority: 700
```

ティップ
この apt-pinning テクニックは stable アーカイブを追跡している際にも有効です。著者の経験では、文書パッケージは unstable アーカイブからインストールしても今までいつも安全でした。

2.7.7 **experimental** からのパッケージと共に、**unstable** を追いかける

次に unstable を追跡しながら experimental にある特定の新規アップストリームバージョンのパッケージを取り込む **apt-pinning** テクニックの例を示します。すべての必要なアーカイブを”/etc/apt/sources.list” ファイルに次のようにリストします。

```
deb http://deb.debian.org/debian/ unstable main contrib non-free
deb http://deb.debian.org/debian/ experimental main contrib non-free
deb http://security.debian.org/ testing/updates main contrib
```

experimental アーカイブのデフォールトの Pin-Priority 値は、**NotAutomatic** アーカイブ (項2.5.3参照下さい) なので、常に 1 (<<100) です。experimental アーカイブにある特定パッケージを次回アップグレード時に自動的に追跡しようとしない限り、”/etc/apt/preferences” ファイル中で experimental アーカイブを使うために Pin-Priority 値を明示的に設定する必要はありません。

2.7.8 パッケージの自動ダウンロードとアップグレード

apt パッケージには、パッケージの自動ダウンロードのサポートする専用の cron スクリプト”/etc/cron.daily/apt” が同梱されています。このスクリプトは unattended-upgrades パッケージをインストールすることで自動アップグレード実行の機能拡張をします。これらは、”/usr/share/doc/unattended-upgrades/README” に記述され

ているように、”/etc/apt/apt.conf.d/02backup” と”/etc/apt/apt.conf.d/50unattended-upgrades” の中のパラメーターでカスタム化できます。

unattended-upgrades パッケージは基本的に stable システムのセキュリティーアップグレードのためです。既存の stable システムが、自動アップグレードで壊される危険性が、セキュリティーアップグレードがすでに閉じたセキュリティーホールからの侵入者によりシステムが壊わされる危険性より小さいなら、パラメーターを次のように設定して自動アップグレードをするのも一計です。

```
APT::Periodic::Update-Package-Lists ”1”;
APT::Periodic::Download-Upgradeable-Packages ”1”;
APT::Periodic::Unattended-Upgrade ”1”;
```

unstable システムを使っている場合には、自動アップグレードするとシステムはいつの日か確実に壊われるので、それはしたくないでしょう。そんな unstable の場合でも、次に記すような事前にパッケージをダウンロードするパラメーターを設定でインタラクティブなアップグレードをするための時間を節約したいでしょう。

```
APT::Periodic::Update-Package-Lists ”1”;
APT::Periodic::Download-Upgradeable-Packages ”1”;
APT::Periodic::Unattended-Upgrade ”0”;
```

2.7.9 APT のよるダウンロードバンド幅の制限

APT によるダウンロードのバンド幅を例えば 800Kib/sec (=100kiB/sec) に制限したい場合には、APT のパラメーターを次のように設定します。

```
APT::Acquire::http::Dl-Limit ”800”;
```

2.7.10 緊急ダウングレード

注意
Debian では設計としてはダウングレードを正式にサポートしません。緊急の復元処置の一部としてのみ実行されるべきです。こういう状況であるにもかかわらず、多くの場合にうまく機能することが知られています。重要なシステムでは回復処置の後に全ての重要データーをバックアップし、最初から新規システムを再インストールします。

壊れたシステムアップグレードからの復元するために、候補バージョンを操作して新しいアーカイブから古いアーカイブにダウングレードすることがうまくいくかもしれません (項2.7.3参照下さい)。これは、何度も”dpkg -i <broken-package>_<old-version>.deb” コマンドを実行する退屈な作業をしないでよくする方法です (項2.6.4参照下さい)。

次に記すような”unstable” を追跡する”/etc/apt/sources.list” ファイル中の行を探します。

```
deb http://deb.debian.org/debian/ sid main contrib non-free
```

それを testing を追いかけるように次と交換します。

```
deb http://deb.debian.org/debian/ bookworm main contrib non-free
```

”/etc/apt/preferences” を次のように設定します。

```
Package: *
Pin: release a=testing
Pin-Priority: 1010
```

”apt-get dist-upgrade” を実行して、システム全体にわたってパッケージのダウングレードを強制します。

この緊急ダウングレードの後でこの特別の”/etc/apt/preferences” ファイルを削除します。

ティップ
依存関係の問題を最小限とすべく、できるだけ多くのパッケージを削除 (remove で、完全削除 purge ではありません!) します。システムのダウングレードのためには手動でいくつかのパッケージを削除とインストールしなければいけないかも知れません。Linux カーネルやブートローダーや udev や PAM や APT やネットワーク関係のパッケージやそれらの設定ファイルには特に注意が必要です。

2.7.11 誰がパッケージをアップロードしたのか?

"/var/lib/dpkg/available" や"/usr/share/doc/package_name/changelog" の中にリストされたメンテナの名前は" 誰がパッケージ化活動の背後にいるのか" に関していくばくかの情報を提供しますが、パッケージを実際にアップロードをした人がはっきりしません。devscripts パッケージ中の who-uploads(1) は Debian のソースパッケージを実際にアップロードした人を確定します。

2.7.12 equivs パッケージ

ソースからプログラムをコンパイルして Debian パッケージを置換えたい際には、それを実際にローカルで Debian 化してパッケージ (*.deb) して、私的アーカイブを使うのが好ましいです。

しかし、プログラムをソースからコンパイルして"/usr/local" にインストールすることを選んだ際には、パッケージ依存関係を満足させるための最後の手段として equivs を使う必要があるかもしれません。

```
Package: equivs
Priority: optional
Section: admin
Description: Circumventing Debian package dependencies
 This package provides a tool to create trivial Debian packages.
 Typically these packages contain only dependency information, but they
 can also include normal installed files like other packages do.
 .
 One use for this is to create a metapackage: a package whose sole
 purpose is to declare dependencies and conflicts on other packages so
 that these will be automatically installed, upgraded, or removed.
 .
 Another use is to circumvent dependency checking: by letting dpkg
 think a particular package name and version is installed when it
 isn't, you can work around bugs in other packages' dependencies.
 (Please do still file such bugs, though.)
```

2.7.13 安定版システムへのパッケージ移植

stable システムの部分アップグレードのためには、その環境内でソースパッケージを使ってパッケージをリビルドするのが好ましいです。こうすることでパッケージ依存関係による大掛かりなアップグレードをしないで済みます。

stable システムのための"/etc/apt/sources.list" ファイルに次のエントリーを追加します。

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

コンパイルするのに必要なパッケージをインストールしソースパッケージをダウンロードをします。

```
# apt-get update
# apt-get dist-upgrade
# apt-get install fakeroot devscripts build-essential
# apt-get build-dep foo
$ apt-get source foo
$ cd foo*
```

バックポートに必要な際には、dpkg や debhelper 等のツールチェインパッケージをバックポートパッケージを用いてアップデートします。

次を実行します。

```
$ dch -i
```

”+bp1” を後ろに付けるなどして、”debian/changelog” 中でパッケージバージョンを先に進める

次のようにしてパッケージをビルドしシステムにインストールします。

```
$ debuild
$ cd ..
# debi foo*.changes
```

2.7.14 APT のためのプロキシサーバー

Debian アーカイブの特定サブセクション全てをミラーするとディスク空間とネットワークのバンド幅の大いなる無駄遣いですので、LAN 上に多くのシステムを管理している際には APT のためのローカルのプロキシサーバーを設置することを考えるのは良いことです。APT は、apt.conf(5) とか”/usr/share/doc/apt/examples/configure-index.gz に説明されたようにして、汎用の squid のようなウェッブ (http) プロキシサーバー (項6.10参照下さい) を使うように設定できます。”$http_proxy” 環境変数による設定は、”/etc/apt/apt.conf” ファイル中の設定より優先します。

Debian アーカイブ専用のプロキシツールがあります。実際に使う前に BTS をチェック下さい。

パッケージ	ポプコン	サイズ	説明
approx	V:0, I:0	6317	Debian アーカイブファイルのキャッシュプロキシサーバー (コンパイルされた OCaml プログラム)
apt-cacher	V:0, I:0	289	Debian パッケージとソースファイルのキャッシュプロキシ (Perl プログラム)
apt-cacher-ng	V:5, I:5	1488	ソフトウエアーパッケージの頒布ためのキャッシュプロキシ (コンパイルされた C++ プログラム)

Table 2.19: Debian アーカイブ専用のプロキシツールのリスト

注意
Debian がそのアーカイブ構造を再編した際に、このような専用のプロキシツールはパッケージメンテナによるコードの修正が必要で、一定期間使えなくなることがあります。一方、汎用のウェッブ (http) プロキシは比較的堅牢ですしそのような変化に合わすのも簡単です。

2.7.15 小さな公開パッケージアーカイブ

ティップ
手動でパッケージアーカイブを設定するのは複雑です。いくつかのレポジトリー管理ツールが有ります。総括リスト がオンラインであります。

近代的なセキュアー **APT** システム (項2.5.2参照下さい) と互換性のある小規模のパブリックアーカイブを作る例を次に示します。まず、いくつかの仮定をします。

- アカウント名: ”foo”

- ホスト名: ”www.example.com”
- 必要なパッケージ: apt-utils や gnupg 等のパッケージ
- URL: ”http://www.example.com/~foo/” (→ ”/home/foo/public_html/index.html”)
- パッケージのアーキテクチャ: ”amd64”

サーバーシステム上で Foo のアーカイブキーを作成します。

```
$ ssh foo@www.example.com
$ gpg --gen-key
...
$ gpg -K
...
sec   1024D/3A3CB5A6 2008-08-14
uid                  Foo (ARCHIVE KEY) <foo@www.example.com>
ssb   2048g/6856F4A7 2008-08-14
$ gpg --export -a 3A3CB5A6 >foo.public.key
```

Foo に関するキー ID”3A3CB5A6” のアーカイブキーファイル”foo.public.key” を公開

”Origin: Foo” というアーカイブツリーを作成します。

```
$ umask 022
$ mkdir -p ~/public_html/debian/pool/main
$ mkdir -p ~/public_html/debian/dists/unstable/main/binary-amd64
$ mkdir -p ~/public_html/debian/dists/unstable/main/source
$ cd ~/public_html/debian
$ cat > dists/unstable/main/binary-amd64/Release << EOF
Archive: unstable
Version: 4.0
Component: main
Origin: Foo
Label: Foo
Architecture: amd64
EOF
$ cat > dists/unstable/main/source/Release << EOF
Archive: unstable
Version: 4.0
Component: main
Origin: Foo
Label: Foo
Architecture: source
EOF
$ cat >aptftp.conf <<EOF
APT::FTPArchive::Release {
  Origin ”Foo”;
  Label ”Foo”;
  Suite ”unstable”;
  Codename ”sid”;
  Architectures ”amd64”;
  Components ”main”;
  Description ”Public archive for Foo”;
};
EOF
$ cat >aptgenerate.conf <<EOF
Dir::ArchiveDir ”.”;
Dir::CacheDir ”.”;
TreeDefault::Directory ”pool/”;
TreeDefault::SrcDirectory ”pool/”;
Default::Packages::Extensions ”.deb”;
Default::Packages::Compress ”. gzip bzip2”;
```

```
Default::Sources::Compress ”gzip bzip2”;
Default::Contents::Compress ”gzip bzip2”;

BinDirectory ”dists/unstable/main/binary-amd64” {
  Packages ”dists/unstable/main/binary-amd64/Packages”;
  Contents ”dists/unstable/Contents-amd64”;
  SrcPackages ”dists/unstable/main/source/Sources”;
};

Tree ”dists/unstable” {
  Sections ”main”;
  Architectures ”amd64 source”;
};
EOF
```

あなたのサーバーシステム上の APT アーカイブ内容の繰り返しアップデートは dupload を設定することで自動化できます。

次に示す内容の”~/.dupload.conf” を設定したクライアントで”dupload -t foo changes_file” を実行して、全てのパッケージファイルを”~foo/public_html/debian/pool/main/” に設置します。

```
$cfg{’foo’} = {
  fqdn => ”www.example.com”,
  method => ”scpb”,
  incoming => ”/home/foo/public_html/debian/pool/main”,
  # The dinstall on ftp-master sends emails itself
  dinstall_runs => 1,
};

$cfg{’foo’}{postupload}{’changes’} = ”
  echo ’cd public_html/debian ;
  apt-ftparchive generate -c=aptftp.conf aptgenerate.conf;
  apt-ftparchive release -c=aptftp.conf dists/unstable >dists/unstable/Release ;
  rm -f dists/unstable/Release.gpg ;
  gpg -u 3A3CB5A6 -bao dists/unstable/Release.gpg dists/unstable/Release’|
  ssh foo@www.example.com  2>/dev/null ;
  echo ’Package archive created!’”;
```

dupload(1) が起動する **postupload** フックスクリプトがアップロード毎に更新されたアーカイブファイルを作成します。

この小規模のパブリックアーカイブをクライアントシステムの apt 行に追加できます。

```
$ sudo bash
# echo ”deb http://www.example.com/~foo/debian/ unstable main” \
   >> /etc/apt/sources.list
# apt-key add foo.public.key
```

ティップ
もしローカルファイルシステム上にアーカイブがある場合には、上記の代わりに”deb file:///home/foo/debian/ …” が使えます。

2.7.16 システム設定の記録とコピー

パッケージと debconf の選択状態のローカルコピーは次に記すようにして作成できます。

```
# dpkg --get-selections ’*’ > selection.dpkg
# debconf-get-selections    > selection.debconf
```

ここで、”*” は”selection.dpkg” が”purge” に関するパッケージ項目も含めるようにします。

これら 2 ファイルを他のコンピューターに移動し、次のようにしてインストールします。

```
# dselect update
# debconf-set-selections < myselection.debconf
# dpkg --set-selections  < myselection.dpkg
# apt-get -u dselect-upgrade    # or dselect install
```

実質的に同じ設定でクラスターとなった多くのサーバーを管理することをお考えの場合には、専用パッケージである fai 等を使って全システムを管理することを考えます。

2.7.17 外来のバイナリーパッケージの変換やインストール

alien(1) を使うと、Red Hat の rpm や Stampede の slp や Slackware の tgz や Solaris の pkg ファイルフォーマットを Debian の deb パッケージに変換できます。あなたのシステムにインストールしたパッケージに替えて他の Linux ディストリビューション由来のパッケージを使いたい際には、alien を使って変換しインストールします。alien は LSB パッケージをサポートします。

警告
alien(1) は sysvinit や libc6 や libpam-modules 等の必須のシステムパッケージを置き換えるために使うべきではありません。実質的には alien(1) は、LSB 準拠か静的にリンクされた **non-free** のバイナリーのみで提供されるパッケージにのみ使われるべきです。フリーソフトウエアーの場合は、ソースパッケージを使い本物の Debian パッケージを作るべきです。

2.7.18 dpkg を使わないパッケージの開梱

”dpkg*.deb” パッケージの内容は、どんな Unix 的環境でも標準の ar(1) と tar(1) を使うことで、dpkg(1) を使うこと無く開梱できます。

```
# ar x /path/to/dpkg_<version>_<arch>.deb
# ls
total 24
-rw-r--r-- 1 bozo bozo  1320 2007-05-07 00:11 control.tar.gz
-rw-r--r-- 1 bozo bozo 12837 2007-05-07 00:11 data.tar.gz
-rw-r--r-- 1 bozo bozo     4 2007-05-07 00:11 debian-binary
# mkdir control
# mkdir data
# tar xvzf control.tar.gz -C control
# tar xvzf data.tar.gz -C data
```

他の”*.deb” パッケージの内容は、上記のようにして”dpkg*.deb” から取り出した dpkg-deb(1) コマンドにより開梱できるし、また標準の ar(1) と tar(1) を xz(1) 解凍サポートともに使うことで上記同様に開梱できます。

パッケージの内容は mc コマンドを使っても閲覧できます。

2.7.19 パッケージ管理の追加参考文書

パッケージ管理に関しては次の文書からさらに学習できます。

- パッケージ管理の一義的文書:
  - aptitude(8) と dpkg(1) と tasksel(8) と apt(8) と apt-get(8) と apt-config(8) と apt-key(8) と sources.list(5) と apt.conf(5) と apt_preferences(5);

  - ”/usr/share/doc/apt-doc/guide.html/index.html”と”/usr/share/doc/apt-doc/offline.html/ind
    from the apt-doc package;
  - aptitude-doc-en パッケージに入っている、”/usr/share/doc/aptitude/html/en/index.html”。

- 正規で詳細な Debian アーカイブに関する文書:
  - Debian アーカイブの正式のポリシーは Debian ポリシーマニュアル、第 2 章 - Debian アーカイブに規定されています。
  - ”Debian 開発者リファレンス、第 4 章 Debian 開発者が利用可能なリソース 4.6 Debian アーカイブ”と、
  - ”The Debian GNU/Linux FAQ, Chapter 6 - The Debian FTP archives”.

- Debian ユーザー向けの Debian パッケージ作成の入門書:
  - ”Debian メンテナ入門” (廃止対象).
  - ”Debian Maintainer 向け案内書”.

Chapter 3

システムの初期化

Debian システムが以下に起動され設定されるかの知っていることはシステム管理者として賢明です。正確で詳細な情報がインストールされたパッケージのソースや文書中にあるとは言え、我々の大部分にとってはちょっと大変過ぎます。

著者などの過去の知見に基づき Debian システムの要点とそれらの設定の簡単な参考となる概論を提供するように勤めました。Debian システムは動く標的なので、システムの状況が変わっているかもしれません。システムに変更を加える前に、各パッケージの最新文書を参照下さい。

ティップ
`systemd` に準拠するシステムのブートアッププロセスは bootup(7) に詳述されている。(最新の Debian)

ティップ
UNIX System V Release 4 に準拠するシステムのブートアッププロセスは boot(7) に詳述されている。(過去の Debian)

3.1 ブートストラッププロセスの概要

コンピューターシステムは、電源投入イベントからユーザーに機能の完備したオペレーティングシステム (OS) を提供するまでブートストラッププロセスを数段通過します。

単純化のため、デフォールトのインストールをした典型的な PC プラットフォームに限定し議論します。

典型的なブートストラッププロセスは 4 段ロケットのようです。各段のロケットは次の段のロケットにシステムのコントロールを引き継ぎます。

- 項3.1.1
- 項3.1.2
- 項3.1.3
- 項3.1.4

もちろん、これらに関して異なる設定をすることはできます。例えば、自分自身で専用カーネルをコンパイルした場合、ミニ Debian システムのステップをスキップできます。自分自身で確認するまでは、あなたのシステムがこの様になっていると決めつけないで下さい。

注意
非伝統的 PC プラットフォームの SUN とか Macintosh システム等では、ROM 上の BIOS やディスク上のパーティション (項9.5.2) が非常に異なっているかもしれません。そのような場合にはプラットフォーム特定の文書をどこかで求めて下さい。

3.1.1 1 段目: BIOS

BIOS は電源投入イベントが引き起こすブートプロセスの 1 段目です。CPU のプログラムカウンターが電源投入イベントで初期化され、読出し専用メモリー (ROM) 上にあるBIOS が特定のメモリーアドレスから実行されます。

BIOS はハードウエアーの基本的な初期化 (POST: 電源投入時自己診断) を行い、システムのコントロールをあなたが提供する次のステップにシステムのコントロールを引き継ぎます。BIOS は通常ハードウエアーによって供給されます。

BIOS 初期画面はどのキーを押すと BIOS 設定画面に入って BIOS の挙動を設定できるかを通常表示しています。よく使われるキーは F1 や F2 や F10 や Esc や Ins や Del です。もし BIOS 初期画面が洒落た画像表示で隠されている場合、Esc 等の何らかのキーをおすとこれを無効にできます。こういったキーはハードウエアーに大いに依存します。

BIOS が起動するコードのハードウエアー上の場所や優先順位は BIOS 設定画面から選択できます。典型的には最初に見つかった選択されたデバイス (ハードディスクやフロッピーディスクや CD-ROM 等) の最初の数セクターがメモリー上にロードされこの初期コードが実行されます。この初期コードは次のいずれでもよろしい。

- ブートローダーコード
- FreeDOS のような踏み石 OS のカーネルコード
- この小さな空間に収まればターゲット OS のカーネルコード

典型的にはプライマリハードディスクの指定されたパーティションからシステムが起動されます。伝統的 PC のハードディスクの最初の 2 セクターにマスターブートレコード (MBR) が含まれます。ブート選択に含まれるディスクのパーティション情報はこの MBR の最後に記録されています。BIOS から実行される最初のブートローダーコードは残りの部分を占めます。

3.1.2 2 段目: ブートローダー

ブートローダーは BIOS によって起動されるブートプロセスの 2 段目です。それはシステムのカーネルイメージと initrd イメージをメモリーにロードし、それらにコントロールを引き継ぎます。この initrd イメージはルートファイルシステムイメージで、そのサポートは使われるブートローダー次第です。

Debian システムは通常 Linux カーネルをデフォールトのシステムカーネルとして使っています。現在の 2.6/3.x カーネルにとっての initrd イメージは技術的に言うなら initramfs (初期 RAM ファイルシステム) イメージです。基本的な initrd イメージはルートファイルシステム中の圧縮 cpio アーカイブです。カーネルはこの基本的な initrd イメージをロードする前の非常に初期のブート中に microcode をアップデートすることができます。非圧縮 cpio フォーマット中の microcode のバイナリーブロッブと基本的な initrd イメージよりなる組み合わせ initrd イメージを作成することを可能にします。

ティップ
`initramfs-tools-core` パッケージ中の lsinitramfs(8) や unmkinitramfs(8) を用いると initrd イメージファイルの内容を検査できます。詳細は https://wiki.debian.org/initramfs を参照ください。

Debian システムのデフォールトインストールでは、GRUB ブートローダーの 1 段目のコードを PC プラットホームの MBR の中に置きます。多くのブートローダーと設定の選択肢が利用可能です。

パッケージ	ポプコン	サイズ	**initrd**	ブートローダー	説明
grub-legacy	V:0, I:2	735	サポート	GRUB Legacy	ディスクパーティションや vfat や ext3 等のファイルシステムを理解するぐらいスマートです。
grub-pc	V:28, I:774	533	サポート	GRUB 2	ディスクパーティションや vfat や ext4 等のファイルシステムを理解するぐらいスマートです。(デフォルト)
grub-rescue-pc	V:0, I:1	6367	サポート	GRUB 2	GRUB 2 のブート可能なレスキューイメージ (CD とフロッピー) (PC/BIOS バージョン)
lilo	V:0, I:2	697	サポート	Lilo	ハードディスク上のセクター位置に依存します。(旧式)
syslinux	V:4, I:48	343	サポート	Isolinux	ISO9660 ファイルシステムを理解します。ブート CD に使われています。
syslinux	V:4, I:48	343	サポート	Syslinux	MSDOS ファイルシステム (FAT) 理解します。ブートフロッピーで使われます。
loadlin	V:0, I:1	90	サポート	Loadlin	新しいシステムが FreeDOS/MSDOS システムから起動されます。
mbr	V:0, I:7	50	非サポート	Neil Turton の MBR	MSDOS の MBR を代替するフリーソフトウエアーです。ディスクパーティションを理解するだけです。

Table 3.1: ブートローダーのリスト

警告
`grub-rescue-pc` パッケージのイメージから作ったブート可能なレスキューメディア (CD かフロッピー) 無しにブートローダーを試してはいけません。これさえあると、ハードディスク上に機能するブートローダーが無くともシステムの起動ができます。

GRUB Legacy のメニューの設定は”`/boot/grub/menu.lst`” にあります。例えば、次のような内容です。

```
title           Debian GNU/Linux
root            (hd0,2)
kernel          /vmlinuz root=/dev/hda3 ro
initrd          /initrd.img
```

GRUB 2 のメニューの設定は”`/boot/grub/grub.cfg`” にあります。”`/etc/grub.d/*`” の雛形と”`/etc/default/grub`” の設定から”`/usr/sbin/update-grub`” を使って自動的に作られます。例えば、次のような内容です。

```
menuentry ”Debian GNU/Linux” {
        set root=(hd0,3)
        linux /vmlinuz root=/dev/hda3
        initrd /initrd.img
}
```

これらの例で、これらの GRUB パラメーターは次の意味です。

注意
GRUB legacy プログラムが使うパーティション値は Linux カーネルやユーティリティーツールが使う値より 1 つ少ない数字です。GRUB 2 プログラムはこの問題を修正します。

GRUB パラメーター	意味
`root`	GRUB legacy では"`(hd0,2)`" また GRUB 2 では"`(hd0,3)`" と設定することでプライマリディスクの 3 つ目のパーティションを使用
`kernel`	カーネルパラメーター"`root=/dev/hda3 ro`" とともに"`/vmlinuz`" にあるカーネルを使用
`initrd`	"`/initrd.img`" にある initrd/initramfs イメージを使用

Table 3.2: GRUB パラメーターの意味

ティップ
UUID (項9.5.3参照下さい) は、"`/dev/hda3`" のようなファイル名に代わるブロックの特定デバイスを確定するのに使え、例えば"`root=UUID=81b289d5-4341-4003-9602-e254a17ac232 ro`" です。

ティップ
もし GRUB が使われている場合には、カーネルブートパラメーターは `/boot/grub/grub.cfg` 中に設定されます。Debian システム上では、`/boot/grub/grub.cfg` を直接編集するべきではありません。`/etc/default/grub` 中の `GRUB_CMDLINE_LINUX_DEFAULT` の値を編集するべきで、update-grub(8) を実行することで `/boot/grub/grub.cfg` を更新すべきです。

ティップ
チェインロード (連鎖導入) とよばれる技術を使うと、あるブートローダーから他のブートローダーを起動できます。

"`info grub`" と grub-install(8) を参照下さい。

3.1.3 3 段目: ミニ Debian システム

ミニ Debian システムはブートローダーによって起動されるブートプロセスの 3 段目です。メモリー上でルートファイルシステムとともにシステムカーネルを実行します。これはオプションの起動プロセスの準備段階です。

注意
" ミニ Debian システム" は著者がこの 3 段目のブートプロセスを本文書中で記述するために作った言葉です。このシステムは一般に initrd とか initramfs システムと呼ばれています。類似のメモリー上のシステムは Debian インストーラーでも使われています。

"`/init`" スクリプトはこのメモリー上のルートファイルシステムで最初に実行されるプログラムです。それはユーザー空間でカーネルを初期化し次の段階にコントロールを引き継ぐプログラムです。このミニ Debian システムは、メインのブートプロセスが始まる前にカーネルモジュールを追加したり、ルートファイルシステムを暗号化されたファイルシステムとしてマウントする等のブートプロセスの柔軟性を提供します。

- `initramfs-tools` で initramfs が作成された場合には"`/init`" プログラムはシェルプログラムです。
  - "`break=init`" 等をカーネルブートパラメーターとして与えると、本部分のブートプロセスに割り込み root シェルを獲得できます。この他の割り込み条件は"`/init`" スクリプトを参照下さい。このシェル環境はあなたの機器のハードウエアーを詳細に検査できるだけ十分洗練されています。
  - このミニ Debian システムで利用可能なコマンドは機能を削ったコマンドで、主に busybox(1) という GNU ツールで提供されます。

- `dracut` で initramfs が作成された場合には”`/init`” プログラムはバイナリーの `systemd` プログラムです。
  - このミニ Debian システムで利用可能なコマンドは機能を削った systemd(1) 環境です。

注意
読出しのみのルートファイルシステム上では、`mount` コマンドには”`-n`” オプションを使う必要があります。

3.1.4 4 段目: 通常の Debian システム

通常の Debian システムはミニ Debian システムによって起動されるブートプロセスの 4 段目です。ミニ Debian システムのシステムカーネルはこの環境ででも実行され続けます。ルートファイルシステムはメモリー上から本当にハードディスク上にあるファイルシステムに切り替えられます。

多くのプログラムを起動する主ブートプロセスを行う init プログラムは、PID=1 で最初のプログラムとして実行されます。init プログラムのデフォールトのファイルパスは”`/sbin/init`” ですが、”`init=/path/to/init_program`” のようなカーネルブートパラメーターにより変更できます。

デフォルトの init プログラムは変化してきています:

- `squeeze` 以前の Debian は、単純な SysV-スタイル init を使用します。
- `wheezy` の Debian は、LSB ヘッダーを用いブート順決めブートスクリプトを並列起動をする改良された SysV-スタイル init を使用します。
- `jessie` の Debian は、イベントドリブンで並列初期化のために systemd にデフォルトの init を切り替えました。

ティップ
あなたのシステム上の実際の init コマンドは”`ps --pid 1 -f`” コマンドで確認できます。

ティップ
Debian `jessie` 以降”`/sbin/init`” は”`/lib/systemd/systemd`” にシムリンクされています。

ティップ
ブートプロセスを高速化する最新のティップは Debian wiki: BootProcessSpeedup を参照下さい。

3.2 Systemd init

このセクションは `PID=1` (詰まり、init プロセス) の systemd(1) プログラムがどのようにシステムを起動するのかを説明します。

`systemd` の init プロセスは、SysV 的な手続き定義スタイルではなく宣言定義スタイルで書かれた unit 設定ファイル (systemd.unit(5) 参照) に従い並列で複数プロセスを起動します。これらは以下に記すような複数のパス (systemd-system.conf(5) 参照) から読み込まれます:

- ”`/lib/systemd/system`”: OS のデフォルトの設定ファイル
- ”`/etc/systemd/system`”: OS デフォルト設定ファイルをオーバーライドするシステム管理者設定ファイル

パッケージ	ポプコン	サイズ	説明
systemd	V:810, I:916	15998	並行処理のためのイベント依存の init(8) デーモン (sysvinit 代替)
systemd-sysv	V:802, I:914	138	systemd で sysvinit を置換するのに必要な、マニュアルページとリンク
systemd-cron	V:1, I:1	143	cron デーモンと anacron 機能を提供する systemd の unit。
init-system-helpers	V:675, I:930	131	sysvinit と systemd 間を切り替える補助ツール
initscripts	V:91, I:323	176	システムの始動と停止のためのスクリプト
sysvinit-core	V:7, I:9	276	System-V 的な init(8) ユーティリティー
sysv-rc	V:183, I:335	81	System-V 的なランレベル変更メカニズム
sysvinit-utils	V:494, I:999	79	System-V 的なユーティリティー (startpar(8)、bootlogd(8)、…)
lsb-base	V:881, I:999	49	Linux Standard Base 3.2 の init スクリプト機能
insserv	V:210, I:330	150	LSB init.d スクリプト依存関係を使いブート順序を整理するツール
uswsusp	V:3, I:8	714	Linux が提供するユーザースペースソフトウエアーによるサスペンドを使うためのツール
kexec-tools	V:1, I:8	278	kexec(8) リブートのための kexec ツール (ワームリブート)
systemd-bootchart	V:0, I:1	128	ブートプロセスのパーフォンマンスアナライザー
bootchart2	V:0, I:0	94	ブートプロセスのパーフォンマンスアナライザー
pybootchartgui	V:0, I:0	177	ブートプロセスのアナライザー (可視化)
mingetty	V:0, I:3	38	コンソール専用 getty(8)
mgetty	V:0, I:1	315	インテリジェントモデム用の代替 getty(8)

Table 3.3: Debian システムののブートユーティリティーのリスト

- "/run/systemd/system": OS デフォルト設定ファイルをオーバーライドする実行時生成される設定ファイル

相互依存関係は"Wants="、"Requires="、"Before="、"After="、…("MAPPING OF UNIT PROPERTIES TO THEIR INVERSES" in systemd.unit(5) 参照) 等の指示定義によって規定される。リソースのコントロールは (systemd.resource-control(5) 参照) によっても定義される。

unit 設定ファイルのサッフィックスにそのタイプを折込みます:

- ***.service** は systemd がコントロールしたりスーパーバイズするプロセスを記述します。systemd.service(5) 参照ください。
- ***.device** は sysfs(5) 中に udev(7) デバイスツリーとして暴露されるデバイスを記述します。See systemd.device(5) を参照ください。
- ***.mount** は systemd がコントロールしたりスーパーバイズするファイルシステムのマウントポイントを記述します。systemd.mount(5) を参照ください。
- ***.automount** は systemd がコントロールしたりスーパーバイズするファイルシステムの自動マウントポイントを記述します。systemd.automount(5) を参照ください。
- ***.swap** は systemd がコントロールやスーパーバイズするスワップデバイスやファイルを記述します。systemd.swap(5) を参照ください。
- ***.path** は systemd がパス基準で起動するために監視するパスを記述します。systemd.path(5) を参照ください。
- ***.socket** は systemd がソケット基準で起動するためにコントロールしたりスーパーバイズするソケットを記述します。systemd.socket(5) を参照ください。
- ***.timer** は systemd がタイマー基準で起動するためにコントロールしたりスーパーバイズするタイマーを記述します。systemd.timer(5) を参照ください。

- ***.slice** は cgroups(7) でリソースを管理します。systemd.slice(5) を参照ください。
- ***.scope** はシステムプロセスの集合を `systemd` のバスインターフェースを用いて管理するためにプログラムで作られます。systemd.scope(5) を参照ください。
- ***.target** は他の unit 設定ファイルを組み合わせて始動時同期点を作ります。systemd.target(5) を参照ください。

システムの始動 (init) されると `systemd` プロセスは (通常"`graphical.target`" にシムリンクされている) "`/lib/systemd/system/default.target` を起動しようとします。. 最初に、"`local-fs.target`" や"`swap.target`" や"`cryptsetup.target`" 等のいくつかの特殊ターゲット unit (systemd.special(7) 参照) が引き込まれファイルシステムをマウントします。そして、他のターゲット unit が、ターゲット unit の依存関係で引き込まれます。詳細に関しては bootup(7) を読んで下さい。

`systemd` はバックワードコンパティビリティー機能を提供します。"`/etc/init.d/rc[0123456S].d/[KS]<name>`" 中の、SysV-スタイルのブートスクリプトは依然として読み込まれ処理されますし、telinit(8) は systemd の unit 有効化要求に変換されます。

注意
擬似実装された runlevel の 2 から 4 は、すべて同じ"`multi-user.target`" にシムリンクされます。

3.2.1 ホスト名

カーネルがシステムのホスト名を維持管理します。`systemd-hostnamed.service` により起動されたシステム unit が"`/etc/hostname`" に保存された名前を使ってブート時にホスト名を設定します。このファイルには、完全修飾ドメイン名ではなく、システムのホスト名のみが含まれているべきです。

現在のホスト名を確認するには、hostname(1) を引数無しで実行します。

3.2.2 ファイルシステム

通常のディスクやネットワークのファイルシテムのマウントオプションは"`/etc/fstab`" で設定されます。fstab(5) と項9.5.7を参照下さい。

暗号化されたファイルシステムの設定は"`/etc/crypttab`" で設定されます。crypttab(5) を参照ください。

mdadm(8) を用いるソフトウエアー RAID は"`/etc/mdadm/mdadm.conf`" で設定されます。mdadm.conf(5) を参照ください。

警告
各ブートアップごとに、全てのファイルシステムをマウントした後で、"`/tmp`" と"`/var/lock`" と"`/var/run`" 中の一時ファイルはクリーンされます。

3.2.3 ネットワークインターフェースの初期化

最近の `systemd` 下の Debian デスクトップ環境では、ネットワークインターフェースは、`lo` が"`networking.service`" で、他のインターフェースが"`NetworkManager.service`" で通常初期化されます。

どのように設定するのかは第5章を参照下さい。.

エラーレベル値	エラーレベル名	意味
0	KERN_EMERG	システムは不安定
1	KERN_ALERT	直ぐアクションが必要
2	KERN_CRIT	クリチカルなコンディション
3	KERN_ERR	エラーコンディション
4	KERN_WARNING	警告コンディション
5	KERN_NOTICE	ノーマルだが重要なコンディション
6	KERN_INFO	情報
7	KERN_DEBUG	デバグレベルのメッセージ

Table 3.4: カーネルエラーレベルのリスト

3.2.4 カーネルメッセージ

コンソールに表示されるカーネルのエラーメッセージは、その閾値て設定できる。

```
# dmesg -n3
```

3.2.5 システムメッセージ

systemd の下では､カーネルとシステムの両方のメッセージがジャーナルサービス systemd-journald.service (所謂 journald) で、”/var/log/journal” の下の恒久的なバイナリーデーターか”/run/log/journal/” 下の非恒久的なバイナリーデーターとして記録されます。このようなバイナリー記録データーには journalctl(1) コマンドを用いてアクセスできます。.

systemd の下ではロギングのユーティリティーである rsyslogd(8) は、(systemd 以前のデフォルトの”/dev/log” を読むのではなく) 揮発性のバイナリーログデーターを読み伝統的な恒久的な ASCII システムログデーターを作成するように挙動が変わります。

ログファイルとスクリーン上の両方のシステムメッセージに関して､”/etc/default/rsyslogd” と”/etc/rsyslog.co によってカスタム化できます。rsyslogd(8) と rsyslog.conf(5) を参照下さい。さらに項9.2.2を参照下さい。

3.2.6 systemd の下でのシステム管理

systemd は init システムを提供するのみならず、ジャーナルのロギングや、ログイン管理や、時間管理や、ネットワーク管理などの汎用システム管理機能を提供します。

systemd(1) はいくつかのコマンドで管理されます:

- systemctl(1) コマンドは systemd システムとサービス管理をコントロールします (CLI)､
- systemsdm(1) コマンドは systemd システムとサービス管理をコントロールします (GLI)､
- journalctl(1) コマンドは systemd ジャーナルの内容照会をします､
- loginctl(1) コマンドは systemd のログイン管理をコントロールします､
- systemd-analyze(1) はシステムのブートアップの性能を解析します。

以下は典型的 systemd 管理コマンドの断片です。正確な意味は該当するマンページを参照ください。

ここで、上記の例の中の”$unit” は単一の unit 名 (.service や .target といったサフィックスは任意) とか、多くの場合、現在メモリー中の全 unit の主名称に対して fnmatch(3) を用いて”*” や”?” や”[]” 等のシェルスタイルのグロブによる複数 unit 指定であっても良い。

上記例中のシステムの状態を変えるコマンドは必要な管理特権を獲得させるべく”sudo” を通常前置する。

”systemctl status $unit|$PID|$device” の出力は色付きドット (”●”) を使い unit の状態が一目瞭然とされる。

操作	タイプ	コマンド断片
GUI のサービスマネージャー	GUI	"systemadm" (systemd-ui パッケージ)
全ターゲットユニット設定をリスト	Unit	"systemctl list-units --type=target"
全サービスユニット設定をリスト	Unit	"systemctl list-units --type=service"
全ユニット設定タイプをリスト	Unit	"systemctl list-units --type=help"
メモリー中の全ソケット unit のリスト	Unit	"systemctl list-sockets"
メモリー中の全タイマー unit のリスト	Unit	"systemctl list-timers"
"$unit" 始動	Unit	"systemctl start $unit"
"$unit" 停止	Unit	"systemctl stop $unit"
サービス特定の設定の再ロード	Unit	"systemctl reload $unit"
"$unit" 停止と始動	Unit	"systemctl restart $unit"
"$unit" 始動と、他全ての停止	Unit	"systemctl isolate $unit"
"graphical" に切り替え (GUI システム)	Unit	"systemctl isolate graphical"
"multi-user" に切り替え (CLI システム)	Unit	"systemctl isolate multi-user"
"rescue" に切り替え (シングルユーザー CLI システム)	Unit	"systemctl isolate rescue"
"$unit" に kill 信号を送る	Unit	"systemctl kill $unit"
"$unit" サービスがアクティブかを確認	Unit	"systemctl is-active $unit"
"$unit" サービスが失敗かを確認	Unit	"systemctl is-failed $unit"
"$unit\|$PID|device" の状態を確認	Unit	"systemctl status $unit\|$PID|$device"
"$unit\|$job" の属性を表示	Unit	"systemctl show $unit\|$job"
失敗した"$unit" をリセット	Unit	"systemctl reset-failed $unit"
全ての unit サービスの依存関係をリスト	Unit	"systemctl list-dependencies --all"
システムにインストールされた unit ファイルをリスト	Unit ファイル	"systemctl list-unit-files"
"$unit" を有効にする (symlink 追加)	Unit ファイル	"systemctl enable $unit"
"$unit" を無効にする (symlink 削除)	Unit ファイル	"systemctl disable $unit"
"$unit" のマスクを外す ("/dev/null" への symlink を削除)	Unit ファイル	"systemctl unmask $unit"
"$unit" にマスクをかける ("/dev/null" への symlink を追加)	Unit ファイル	"systemctl mask $unit"
デフォールトのターゲット設定を取得	Unit ファイル	"systemctl get-default"
"graphical" にデフォールトのターゲットを設定 (GUI システム)	Unit ファイル	"systemctl set-default graphical"
"multi-user" にデフォールトのターゲットを設定 (CLI システム)	Unit ファイル	"systemctl set-default multi-user"
ジョブ環境の表示	環境	"systemctl show-environment"
ジョブ環境"variable"(変数)を"value(値) に設定する"	環境	"systemctl set-environment variable=value"
ジョブ環境"variable" (変数) の設定を解除する	環境	"systemctl unset-environment variable"
全 unit ファイルとデーモンを再起動	ライフサイクル	"systemctl daemon-reload"
システムをシャットダウンする	システム	"systemctl poweroff"

- 白い”●” は” 活動停止” や” 停止済み” の状態を示す。
- 赤い”●” は” 失敗発生” や” エラー発生” の状態を示す。
- 緑の”●” は” 活動中” や” 再起動中” や” 起動中” の状態を示す。

3.2.7 systemd のカスタム化

デフォルトのインストールでは、多くのネットワークサービス (第6章を参照) はブート時に `systemd` によってブート時に `network.target` の後に起動される。”`sshd`” も例外ではありません。カスタム化の例としてオンデマンド起動に”`sshd`” をかえましょう。

最初に、システムがインストールしたサービスの unit を無効化しましょう。

```
$ sudo systemctl stop sshd.service
$ sudo systemctl mask sshd.service
```

古典的 Unix サービスでは `indetd` スーパーサーバーによるによりオンデマンドでソケット有効化をしていました。`systemd` では、***.socket** や ***.service** unit 設定ファイルを等してこれと等価のことができます。

聞くソケットを指定するには `sshd.socket`

```
[Unit]
Description=SSH Socket for Per-Connection Servers

[Socket]
ListenStream=22
Accept=yes

[Install]
WantedBy=sockets.target
```

`sshd.socket` に対応するサービスファイルの `sshd@.service`

```
[Unit]
Description=SSH Per-Connection Server

[Service]
ExecStart=-/usr/sbin/sshd -i
StandardInput=socket
```

そして、再ロードします。

```
$ sudo systemctl daemon-reload
```

3.3 udev システム

Linux カーネル 2.6 以降では、udev システムがハードウエアーの自動検出と初期化のメカニズムを提供します (udev(7) 参照下さい)。カーネルが各デバイスを発見すると、udev システムは sysfs ファイルシステム (項1.2.12参照下さい) からの情報を使いユーザープロセスを起動し、modprobe(8) プログラム (項3.3.1参照下さい) を使ってそれをサポートする必要なカーネルモジュールをロードし、対応するデバイスノードを作成します。

ティップ
もし”`/lib/modules/<kernel-version>/modules.dep`” が何らかの理由で depmod(8) によって適正に生成されていなかった場合には、モジュールは udev システムによる期待にそってロードされないかもしれません。これを修正するには、”`depmod -a`” を実行します。

デバイスノード名は”/etc/udev/rules.d/” の中のファイルによって設定できます。現在のデフォールトのルールは、cd とネットワークデバイス以外は非静的なデバイス名となる動的生成名を作る傾向があります。cd やネットワークデバイスと同様のカスタムルールを追加することで USB メモリースティック等の他のデバイスにも静的なデバイス名を生成出来ます。”Writing udev rules” か”/usr/share/doc/udev/writing_udev_rules/index.html” を参照下さい。

udev システムは少々動くターゲットなので、詳細は他のドキュメントに譲り、ここでは最小限の記述に止めます。

ティップ
”/etc/fstab” 中のマウントルールでは、デバイス名が静的なデバイス名である必要がありません。”/dev/sda” 等のデバイス名ではなく UUID を使ってデバイスをマウントできます。項9.5.3を参照下さい。

3.3.1 カーネルモジュール初期化

modprobe(8) プログラムは、ユーザープロセスからカーネルモジュールを追加や削除することで実行中の Linux カーネルの設定を可能にします。udev システム (項3.3参照下さい) は、その起動を自動化しカーネルモジュールの初期化を補助します。

”/etc/modules” ファイル中にリストしてプリロードする必要のある (modules(5) 参照下さい) 次に記すような非ハードウエアーや特殊ハードウエアーのドライバーモジュールがあります。

- ポイント間ネットワークデバイス (TUN) と仮想 Ethernet ネットワークデバイス (TAP) を提供する、TUN/TAP モジュール
- netfilter ファイアーウォール機能 (iptables(8) と項5.10) を提供する netfilter モジュール
- ウォッチドッグタイマードライバーのモジュール

modprobe(8) プログラムのための設定ファイルは、modprobe.conf(5) で説明されているように”/etc/modprobes.d/” ディレクトリーの下にあります。(あるカーネルモジュールが自動ロードされるのを避けるには、”/etc/modprobes.d/bl
ファイル中にブラックリストします。)

depmod(8) プログラムによって生成される”/lib/modules/<version>/modules.dep” ファイルは、modprobe(8) プログラムによって使われるモジュール依存関係を記述します。

注意
ブート時に modprobe(8) を使ってのモジュールロードの問題に出会った場合には、”depmod -a” として”modules.dep” を再構築をするとこの様な問題が解消できるかもしれません。

---

modinfo(8) プログラムは Linux カーネルモジュールに関する情報を表示します。

lsmod(8) プログラムは”/proc/modules” の内容を読みやすい形式にして、どのカーネルモジュールが現在ロードされているかを表示します。

ティップ
あなたのシステム上の正確なハードウエアーを特定します。項9.4.3を参照下さい。

ティップ
ブート時に期待されるハードウエアー機能を有効となるように設定もできます。項9.4.4を参照下さい。

ティップ
あなたのデバイスのサポートは、カーネルを再コンパイルすれば追加できます。項9.9を参照下さい。

Chapter 4

認証

人 (またはプログラム) がシステムへのアクセスの要求をした際に、認証はその正体が信頼できるものだと確認します。

警告
PAM の設定のエラーはあなたをあなた自身のシステムから締め出すかも知れません。レスキュー CD を手元に置くか代替ブートパーティション設定を必ずします。復元するには、それらを使ってシステムをブートしそこから修正します。

警告
本章は、2013 年にリリースされた Debian 7.0 (Wheezy) に基づいているため、内容が陳腐化しつつあります。

4.1 通常の Unix 認証

通常の Unix 認証は PAM (プラグ可能な認証モジュール) のもとで pam_unix.so(8) モジュールによって提供される。":" で分離されたエントリーを持つその 3 つの重要な設定ファイルは次です。

ファイル	パーミッション (許可)	ユーザー	グループ	説明
/etc/passwd	-rw-r--r--	root	root	(浄化された) ユーザーアカウント情報
/etc/shadow	-rw-r-----	root	shadow	保護されたユーザーアカウント情報
/etc/group	-rw-r--r--	root	root	グループ情報

Table 4.1: 3 つの pam_unix(8) に関する重要な設定ファイル

"/etc/passwd" ファイルは次の内容です。

```
 ...
user1:x:1000:1000:User1 Name,,,:/home/user1:/bin/bash
user2:x:1001:1001:User2 Name,,,:/home/user2:/bin/bash
 ...
```

passwd(5) に説明されているように、このファイルの":" で分離されたエントリーそれぞれは次の意味です。

- ログイン名
- パスワード規定エントリー
- 数値のユーザー ID
- 数値のグループ ID
- ユーザー名またはコメント領域
- ユーザーのホームディレクトリー
- ユーザーのコマンドインタープリター (無いこともある)

”/etc/passwd” の 2 番目のエントリーは暗号化したパスワードのエントリーとして使われていました。”/etc/shadow” が導入された後は、このエントリーはパスワード規定エントリーとして使われています。

内容	意味
(空白)	パスワード無しアカウント
x	暗号化したパスワードは”/etc/shadow” ファイルの中にあります。
*	このアカウントへのログイン不可
!	このアカウントへのログイン不可

Table 4.2: ”/etc/passwd” の 2 番目のエントリーの内容

”/etc/shadow” の内容は次です。

```
 ...
user1:$1$Xop0FYH9$IfxyQwBe9b8tiyIkt2P4F/:13262:0:99999:7:::
user2:$1$vXGZLVbS$ElyErNf/agUDsm1DehJMS/:13261:0:99999:7:::
 ...
```

shadow(5) で説明されているように、このファイルの”:” で分離されたエントリーそれぞれは次の意味です。

- ログイン名
- 暗号化されたパスワード (最初が”1” で始まっているのは MD5 暗号化が使われていることを示します。”*” はログイン不可を示します。)
- 1970 年 1 月 1 日から、最後にパスワードが変更された日までの日数
- パスワードが変更可能となるまでの日数
- パスワードを変更しなくてはならなくなる日までの日数
- パスワード有効期限が来る前に、ユーザが警告を受ける日数
- パスワード有効期限が過ぎてからアカウントが使用不能になるまでの日数
- 1970 年 1 月 1 日からアカウントが使用不能になる日までの日数
- …

”/etc/group” のファイル内容は次です。

```
group1:x:20:user1,user2
```

group(5) に説明されているように、このファイルの”:” で分離されたエントリーそれぞれは次の意味です。

- グループ名

- 暗号化されたパスワード (実際は使われていない)
- 数値のグループ ID
- ”,” で分離されたユーザー名のリスト

注意
”/etc/gshadow” ファイルは”/etc/shadow” ファイルが”/etc/group” ファイルに対する機能と同様の機能がありますが、実際には使われていません。

注意
も し”authoptionalpam_group.so” 行 が”/etc/pam.d/common-auth” に 書 き 加 え れ、”/etc/security/group.conf” に対応する設定がされていれば、実際のユーザーのグループメンバーシップは動的に割り当てられます。pam_group(8) を参照下さい。

注意
base-passwd パッケージはユーザーとグループに関する権威のあるリストが含まれます: ”/usr/share/doc/base-passwd/users-and-groups.html”。

4.2 アカウントとパスワードの情報管理

アカウント情報管理のための重要コマンドを記します。

コマンド	機能
getent passwd <user_name>	”<user_name>” のアカウント情報の閲覧
getent shadow <user_name>	”<user_name>” のシャドーされたアカウント情報の閲覧
getent group <group_name>	”<group_name>” のグループ情報の閲覧
passwd	アカウントのパスワード管理
passwd -e	アカウント開設のための一回だけ使えるパスワードの設定
chage	パスワードのエージング情報管理

Table 4.3: アカウント情報を管理するコマンドのリスト

一部機能が機能するには root 権限が必要な場合があります。パスワードとデーターの暗号化は crypt(3) を参照下さい。

注意
Debian が提供する salsa 機器と同様な PAM と NSS の設定をされたシステム上では、ローカルの”/etc/passwd” や”/etc/group” や”/etc/shadow” の内容がシステムにアクティブに利用されていないことがあります。そういった環境下でも上記コマンドは有効です。

4.3 良好なパスワード

passwd(1) によるとシステムインストール時や passwd(1) コマンドによってアカウント作成する際には、次に記すようなセットからなる少なくとも 6 から 8 文字の良好なパスワードを選択するべきです。

- 小文字のアルファベット

- 数字の 0 から 9
- 句読点

警告
容易に推測できるパスワードを選んではいけません。アカウント名、社会保険番号、電話番号、住所、誕生日、家族員やペットの名前、辞書にある単語、”12345” や”qwerty” のような単純な文字列…、これらすべてパスワードに選んではいけません。

4.4 暗号化されたパスワード作成

ソルトを使って暗号化されたパスワードを生成する独立のツールがあります。

パッケージ	ポプコン	サイズ	コマンド	機能
whois	V:35, I:393	364	mkpasswd	crypt(3) ライブラリーの充実しすぎたフロントエンド
openssl	V:794, I:993	1465	openssl passwd	パスワードハッシュの計算 (OpenSSL)。passwd(1ssl)

Table 4.4: パスワード生成ツールのリスト

4.5 PAM と NSS

Debian システムのような最新の Unix 的システムは PAM (プラグ可能な認証モジュール: Pluggable Authentication Modules) と NSS (ネームサービススイッチ: Name Service Switch) メカニズムをローカルのシステム管理者がそのシステム管理用に提供します。それらの役割をまとめると次の様になります。

- PAM は、アプリケーションソフトウエアーが使う柔軟な認証メカニズムを提供し、パスワードデーターの交換に関与します。
- NSS は、ls(1)andid(1) 等のプログラムがユーザーやグループの名前を得ために C 標準ライブラリー経由で頻用する柔軟なネームサービスメカニズムを提供します。

これらの PAM と NSS システムは一貫した設定が必要です。

PAM と NSS システムに関する注目のパッケージは次です。

- libpam-doc 中の”The Linux-PAM System Administrators’ Guide” は PAM 設定を学ぶ上で必須です。
- glibc-doc-reference の中の”System Databases and Name Service Switch” セクションは NSS 設定を学ぶ上で必須です。

注意
より大規模かつ最新のリストは”aptitude search ’libpam-|libnss-’” コマンドを実行すると得られます。NSS という頭字語は” ネームサービススイッチ: Name Service Switch” と異なる” ネットワークセキュリティーサービス: Network Security Service” を指すこともあります。

注意
PAM は個別プログラムに関する環境変数をシステム全体のデフォールト値に初期化する最も基本的な手段です。

パッケージ	ポプコン	サイズ	説明
libpam-modules	V:807, I:999	1032	差し替え可能な認証モジュール (基本サービス)
libpam-ldap	I:12	249	LDAP インターフェースを可能にする差し替え可能な認証モジュール
libpam-cracklib	I:16	115	cracklib のサポートを可能にする差し替え可能な認証モジュール
libpam-systemd	V:474, I:853	573	logind のために登録ユーザーセッションを登録するプラガブルオーセンティケーション (PAM)
libpam-doc	I:1	1044	差し替え可能な認証モジュール (html と text の文書)
libc6	V:935, I:999	12771	GNU C ライブラリー: ” ネームサービススイッチ” も提供する共有ライブラリー
glibc-doc	I:11	3161	GNU C ライブラリー: マンページ
glibc-doc-reference	I:5	12740	GNU C ライブラリー: info と pdf と html フォーマットでのリファレンスマニュアル (non-free)
libnss-mdns	I:526	150	マルチキャスト DNS を使った名前解決のための NSS モジュール
libnss-ldap	I:11	265	LDAP をネームサービスとして使う NSS モジュール
libnss-ldapd	I:14	153	LDAP をネームサービスとして使う NSS モジュール (libnss-ldap の新たなフォーク)

Table 4.5: 特記すべき PAM と NSS システムのリスト

systemd の下では、logind のために systemd のコントロールグループ階層中にユーザーセッションを登録することでユーザーのログインを管理すべく libpam-systemd パッケージがインストールされている。systemd-logind(8) や logind.conf(5) や pam_systemd(8) を参照ください。

4.5.1 PAM と NSS によってアクセスされる設定ファイル

PAM と NSS がアクセスする注目すべき設定ファイルを次に記します。

パスワード選択の制限は pam_unix(8) と pam_cracklib(8) モジュールで実装されています。それらは引数を使って設定します。

ティップ
PAM モジュールはファイル名のサフィクスとして”.so” を使います。

4.5.2 最新の集中システム管理

集中化された軽量ディレクトリーアクセスプロトコル (LDAP) を採用することで多くのネットワーク上の Unix 的や非 Unix 的システムを最新の集中システム管理が実現できます。軽量ディレクトリーアクセスプロトコルのオープンソース実装は OpenLDAP ソフトウエアーです。

LDAP サーバーは、libpam-ldap と libnss-ldap パッケージで提供される PAM と NSS を使うことで Debian システムにアカウント情報を提供します。この実現ためにはいくつかの設定が必要です (著者は本設定を使っていないため、次の情報は完全に二次情報です。ご理解の上お読み下さい。)。

- スタンドアローンの LDAP デーモンである slapd(8) 等のプログラムを走らせることで集中化された LDAP サーバーを設置します。
- デフォールトの”pam_unix.so” に代えて”pam_ldap.so” を使うには”/etc/pam.d/” ディレクトリー中の PAM 設定ファイルを変更します。

設定ファイル	機能
/etc/pam.d/< プログラム名 >	"<program_name>" に関する PAM 設定の設定; pam(7) と pam.d(5) 参照下さい
/etc/nsswitch.conf	各サービスに関するエントリーによる NSS 設定の設定; nsswitch.conf(5) 参照下さい
/etc/nologin	ユーザーのログイン制限のために pam_nologin(8) モジュールがアクセス
/etc/securetty	pam_securetty(8) モジュールにより root アクセスに使う tty を制限
/etc/security/access.conf	pam_access(8) モジュールによりアクセス制限を設定
/etc/security/group.conf	pam_group(8) モジュールによりグループに基づく制約を設定
/etc/security/pam_env.conf	pam_env(8) モジュールにより環境変数を設定
/etc/environment	"readenv=1" 引数を付きの pam_env(8) モジュールによって追加での環境変数を設定
/etc/default/locale	"readenv=1envfile=/etc/default/locale" 引数を付きの pam_env(8) モジュールによって追加でロケールを設定します (Debian)
/etc/security/limits.conf	pam_linits(8) モジュールによってリソース制限 (ulimit, core, ⋯) を設定
/etc/security/time.conf	pam_time(8) モジュールによって時間制限を設定
/etc/systemd/logind.conf	systemd ログイン管理設定の設定 (logind.conf(5) と systemd-logind.service(8) を参照)

Table 4.6: PAM NSS によりアクセスされる設定ファイルのリスト

  - Debian では、"/etc/pam_ldap.conf" を libpam-ldap の設定ファイル、"/etc/pam_ldap.secret" を root のパスワードを保存するファイルとして使っています。

- デフォールト ("compat" または"file") に代えて"ldap" を使うには"/etc/nsswitch.conf" ファイル中の NSS 設定を変更します。

  - Debian では、"/etc/libnss-ldap.conf" を libnss-ldap の設定ファイルとして使っています。

- パスワードのセキュリティー確保のために libpam-ldap が SSL (もしくは TLS) 接続を使うよう設定しなければいけません。
- LDAP のネットワークオバーヘッドの犠牲はあるとはいえ、データーの整合性確保のために libnss-ldap が SSL (もしくは TLS) 接続を使うように設定できます。
- LDAP のネットワークトラフィックを減少させるために LDAP サーチ結果を一時保持するための nscd(8) をローカルで走らせるべきです。

libpam-doc パッケージで提供される pam_ldap.conf(5) や"/usr/share/doc/libpam-doc/html/" や glibc-doc パッケージで提供される"info libc 'NameServiceSwitch'" といった文書を参照下さい。

同様に、これに代わる集中化されたシステムは他の方法を使っても設定できます。

- Windows システムとのユーザーとグループの統合
  - winbind と libpam_winbind パッケージを使って Windows ドメインサービスにアクセスします。
  - winbind(8) と SAMBA による MS Windows Networks への統合 を参照下さい。
- 旧来の Unix 的なシステムとのユーザーとグループの統合
  - nis パッケージにより NIS (当初 YP と呼ばれた) または NIS+ にアクセス
  - The Linux NIS(YP)/NYS/NIS+ HOWTO 参照下さい。

4.5.3 「どうして **GNU** の **su** は **wheel** グループをサポートしないのか」

これは Richard M. Stallman が書いた昔の”`info su`” の最後に書かれていた有名な文言です。ご心配は無用です。現在 Debian にある su は PAM を使っているので”`/etc/pam.d/su`” の中の”`pam_wheel.so`” の行をエネーブルすることで su を使えるのを `root` グループに限定できます。

4.5.4 パスワード規則強化

`libpam-cracklib` パッケージをインストールすると、例えば”`/etc/pam.d/common-password`” に次のような行があれば、パスワード規則を強化できます。

`squeeze` の場合:

```
password required pam_cracklib.so retry=3 minlen=9 difok=3
password [success=1 default=ignore] pam_unix.so use_authtok nullok md5
password requisite pam_deny.so
password required pam_permit.so
```

4.6 他のアクセスコントロール

注意
カーネルのセキュアーアテンションキー (SAK) 機能の制限は項9.3.15を参照下さい。

4.6.1 sudo

`sudo` はシステム管理者がユーザーに制限付きの root 権限を与え、その root 活動を記録するように設計されたプログラムです。`sudo` はユーザーの通常パスワードだけが必要です。`sudo` パッケージをインストールし、”`/etc/sudoers`” の中のオプションを設定することによりアクティベートして下さい。”`/usr/share/doc/sudo/examp` や項1.1.12 の設定例を参照下さい。

単一ユーザーシステムにおける私の `sudo` の使い方 (項1.1.12参照下さい) は自分自身の失敗からの防衛を目指しています。`sudo` を使うことは、常に root アカウントからシステムを使うよりは良い方法だと個人的には考えます。例えば、次は”`<some_file>`” の所有者を”`<my_name>`” に変更します。

```
$ sudo chown <my_name> <some_file>
```

root のパスワード (自分でシステムインストールをした Debian ユーザーなら当然知っています) を知っていれば、どのユーザーアカウントからいかなるコマンドも”`su -c`” とすれば root もとで実行できます。

4.6.2 PolicyKit

PolicyKit は Unix 系オペレーティングシステムにおけるシステム全体の特権を制御するオペレーティングシステム構成要素です。

新しい GUI アプリケーションは、特権プロセスとして実行するように設計されていません。それらは、PolicyKit を経由し管理操作を実行する特権プロセスに話しかけます。

PolicyKit は、このような操作を Debian システム上の `sudo` グループ所属のユーザーアカウントに限定します。

polkit(8) を参照下さい。

4.6.3 SELinux

セキュリティー強化した Linux (SELinux) は強制アクセス制御 (MAC) ポリシーのある通常の Unix 的な権限モデルより厳格な枠組みです。ある条件下では root の権限すら制限を受けます。

4.6.4 サーバーのサービスへのアクセスの制限

システムのセキュリティーのためにできるだけ多くのサーバープログラムを無効とするのは良い考えです。このことはネットワークサーバーの場合は決定的です。直接デーモンとしてであれスーパーサーバープログラム経由であれ有効にされている使っていないサーバーがあることはセキュリティーリスクと考えられます。

sshd(8) 等の多くのプログラムが PAM を使ったアクセスコントロールを使っています。サーバーサービスへのアクセスを制限するには多くの方法があります。

- 設定ファイル: "`/etc/default/<` プログラム名 `>`"
- デーモンに関するサービス unit 設定
- PAM (プラグ可能な認証モジュール: Pluggable Authentication Modules)
- スーパーサーバーに関する"`/etc/inetd.conf`"
- TCP ラッパーに関する"`/etc/hosts.deny`" と"`/etc/hosts.allow`"、tcpd(8)
- Sun RPC に関する"`/etc/rpc.conf`"
- atd(8) に関する"`/etc/at.allow`" と"`/etc/at.deny`"
- atd(8) に関する"`/etc/at.allow`" と"`/etc/at.deny`"
- netfilter インフラのネットワークファイアーウォール

項3.2.6と項4.5.1と項5.10 を参照下さい。

ティップ
NFS 他の RPC を使うプログラムためには Sun RPC サービスはアクティブにする必要があります。

ティップ
もし現代的な Debian システムでリモートアクセスで問題に会った場合には、"`/etc/hosts.deny`" 中に"`ALL:PARANOID`" 等の問題となっている設定があればコメントアウトします。(ただしこの種の行為に関するセキュリティーリスクに注意を払わなければいけません。)

4.7 認証のセキュリティー

注意
ここに書かれている情報はあなたのセキュリティーのニーズに充分ではないかもしれませんが、良いスタートです。

インセキュアーなサービス名	ポート	セキュアーなサービス名	ポート
www (http)	80	https	443
smtp (mail)	25	ssmtp (smtps)	465
ftp-data	20	ftps-data	989
ftp	21	ftps	990
telnet	23	telnets	992
imap2	143	imaps	993
pop3	110	pop3s	995
ldap	389	ldaps	636

Table 4.7: インセキュアーとセキュアーのサービスとポートのリスト

4.7.1 インターネット上でセキュアーなパスワード

多くのトランスポーテーションレイヤーサービスはパスワード認証も含めて暗号化せずにメッセージをプレーンテキストで通信します。途中で傍受されかねないインターネットの荒野を経由して暗号化せずパスワードを送ることは非常によくない考えです。これらに関しては、”トランスポーテーションレイヤーセキュリティー”(TLS) もしくはその前身の” セキュアーソケットレイヤー” (SSL) で暗号化することでパスワードを含むすべての通信をセキュアーにしてサービスができます。

暗号化には CPU タイムがかかります。CPU に友好的な代替方法として、POP には” パスワードを認証されたポストオフィスプロトコル”(APOP) や SMTP や IMAP には” チャレンジレスポンス認証メカニズム MD5” (CRAM-MD5) といったセキュアーな認証プロトコルでパスワードのみを保護しつつ通信はプレーンテキストですることもできます。(最近メールクライアントからメールサーバーにインターネット経由でメールメッセージを送る際には、CRAM-MD5 で認証をしたのちネットワークプロバイダーによるポート 25 ブロッキングを避けて従来の SMTP ポート 25 の代わりにメッセージサブミッションポート 587 を使うことがよく行われます。)

4.7.2 セキュアーシェル

セキュアーシェル (SSH) プログラムはセキュアーな認証とともにインセキュアーなネットワークを通過したお互いに信頼し合っていないホスト間のセキュアーで暗号化された通信を可能にします。OpenSSH クライアント ssh(1) と OpenSSH デーモン sshd(8) から成り立っています。SSH はポートフォーワーディング機能を使い POP や X のようなインセキュアープロトコルの通信をインターネット経由でトンネルするのに使えます。

クライアントは、ホストベースド認証、公開鍵認証、チャレンジレスポンス認証、パスワード認証を使って認証をとろうとします。公開鍵認証を利用すると、リモートからのパスワード無しログインができるようになります。項6.9を参照下さい。

4.7.3 インターネットのためのセキュリティー強化策

たとえ、セキュアーシェル (SSH) やポイントツーポイントトンネリングプロトコル (PPTP) サーバーのようなセキュアーサービスを走らせる場合でも、ブルートフォースのパスワード推測等による侵入の可能性は残っています。次のようなセキュリティーのためのツールとともに、ファイアーウォールポリシー (項5.10参照下さい) を使うのはセキュリティー状況を向上させることが期待できます。

4.7.4 root パスワードのセキュリティー確保

あなたの機器に他人が root 権限を持ってアクセスするのを阻止するには、次のアクションが必要です。

- ハードディスクへの物理的アクセスを阻止
- BIOS に鍵を書け、リムーバブルメディアからの起動を阻止

パッケージ	ポプコン	サイズ	説明
knockd	V:0, I:3	102	小さなポートノックのデーモン knockd(1) とクライアント konck
fail2ban	V:112, I:123	2092	複数回の認証エラーを発生させる IP を使用禁止にします
libpam-shield	V:0, I:0	115	パスワード推測によるリモートからの攻撃者を締め出す

Table 4.8: 追加セキュリティー策を提供するツールのリスト

- GRUB のインタラクティブセッションのパスワードを設定
- GRUB のメニュー項目編集に施錠

ハードディスクへの物理的アクセスがあれば、パスワードをリセットすることは次の手順を使うと比較的簡単です。

1. ハードディスクを CD から起動可能な BIOS のついた PC に移動します。
2. レスキューメディア (Debian ブートディスク、Knoppix CD、GRUB CD、…) でシステムを起動します。
3. ルートパーティションを読出し / 書込みアクセスでマウントします。
4. ルートパーティションの”`/etc/passwd`” を編集し、`root` アカウントの 2 番目の項目を空にします。

`grub-rescue-pc` の起動時に GRUB のメニュー項目を編集可能 (項3.1.2参照下さい) なら、次の手順を使えてさらに簡単です。

1. カーネルパラメーターを”`root=/dev/hda6 rw init=/bin/sh`” のような感じに変更してシステムを起動します。
2. ”`/etc/passwd`” を編集し、`root` アカウントの 2 番目の項目を空にします。
3. システム再起動します。

これで、システムの root シェルにパスワード無しに入れるようになりました。

注意
root シェルにアクセスさえできれば、システム上の全てにアクセスできシステム上のどのパスワードでもリセットできます。さらに。`john` とか `crack` パッケージ (項9.4.11参照下さい) のようなブルートフォースのパスワードクラッキングツールを使ってすべてのユーザーアカウントのパスワードが破られるかもしれません。こうして破られたパスワードは他のシステムへの侵入を引き起こしかねません。

この様な懸念を回避できる唯一の合理的なソフトウエアー的解決法は、dm-crypt と initramfs (項9.8 参照下さい) をつかう、ソフトウエアー暗号化されたルートパーティション (もしくは”`/etc`” パーティション) を使うことです。でも、パスワードがシステム起動毎に必要になってしまいます。

Chapter 5

ネットワークの設定

ティップ
GNU/Linux のネットワーク設定の一般的ガイドは Linux Network Administrators Guide を参照下さい。

ティップ
最近の Debian に特化したネットワーク設定のガイドは The Debian Administrator's Handbook —Configuring the Network を参照下さい。

警告
伝統的なインターフェースの命名法 ("eth0", "eth1", "wlan0", ⋯) を使う代わりに、新しい systemd は"enp0s25" のような"予測可能なネットワークインターフェース名" を用います。

警告
本章は、2013 年にリリースされた Debian 7.0 (Wheezy) に基づいているため、内容が陳腐化しつつあります。

ティップ
本ドキュメントはネットワーク設定例に IPv4 を用いて古い ifconfig(8) を使っていますが、Debian は wheezy リリースから IPv4+IPv6 を用いて ip(8) を使うように移行中です。本ドキュメント更新のパッチは歓迎です。

ティップ
systemd の下では、networkd がネットワーク管理に使えます。systemd-networkd(8)y> を参照ください。

5.1 基本的ネットワークインフラ

現代的な Debian システムの基本的ネットワークインフラをレビューします。

パッケージ	ポプコン	サイズ	タイプ	説明
ifupdown	V:587, I:991	217	設定::ifupdown	ネットワークを接続したり切断したりする標準化されたツール (Debian 特定)
ifplugd	V:3, I:18	217	, ,	有線ネットワークを自動的に管理
ifupdown-extra	V:0, I:1	106	, ,	"ifupdown" パッケージを強化するネットワークテストスクリプト
ifmetric	V:0, I:1	37	, ,	ネットワークインターフェースの経路メトリック設定
guessnet	V:0, I:0	422	, ,	"/etc/network/interfaces" ファイル経由で"ifupdown" パッケージを補強する mapping スクリプト
ifscheme	V:0, I:0	59	, ,	"ifupdown" パッケージを補強する mapping スクリプト
network-manager	V:358, I:440	14957	設定::NM	NetworkManager (デーモン): ネットワークを自動管理
network-manager-gnome	V:132, I:372	5540	, ,	NetworkManager (GNOME フロントエンド)
wicd	I:24	36(*)	設定::wicd	有線と無線のネットワークマネージャー (メタパッケージ)
wicd-cli	V:0, I:1	60(*)	, ,	有線と無線のネットワークマネージャー (コマンドラインクライアント)
wicd-curses	V:0, I:3	176(*)	, ,	有線と無線のネットワークマネージャー (Curses クライアント)
wicd-daemon	V:19, I:26	992(*)	, ,	有線と無線のネットワークマネージャー (デーモン)
wicd-gtk	V:15, I:25	576(*)	, ,	有線と無線のネットワークマネージャー (GTK+ クライアント)
iptables	V:300, I:993	2520	設定::Netfilter	パケットフィルターと NAT のための管理ツール (Netfilter)
iproute2	V:672, I:926	2867	設定::iproute2	iproute2、IPv6 や他の上級ネットワーク設定: ip(8) や tc(8) 等
ifrename	V:0, I:3	125	, ,	各種の静的クライテリアに基づきネットワークインターフェースを改名します: ifrename(8)
ethtool	V:102, I:261	597	, ,	Ethernet デバイス設定の表示と変更
iputils-ping	V:234, I:997	113	test::iproute2	ホスト名か IP アドレスによってリモートホストのネットワークからの到達性をテスト (旧来、GNU)
iputils-arping	V:8, I:127	55	, ,	ARP アドレスによって特定されるリモートホストのネットワークからの到達性をテスト
iputils-tracepath	V:4, I:60	72	, ,	リモートホストへのネットワークパスを追跡
net-tools	V:234, I:634	991	設定::net-tools	NET-3 ネットワーキングツールキット (net-tools、IPv4 ネットワーク設定): ifconfig(8) 等
inetutils-ping	V:0, I:1	359	テスト::net-tools	ホスト名か IP アドレスによってリモートホストのネットワークからの到達性をテスト (旧来、GNU)
arping	V:2, I:29	77	, ,	ARP アドレスによって特定されるリモートホストのネットワークからの到達性をテスト (旧来)
traceroute	V:63, I:936	159	, ,	リモートホストへのネットワークパス追跡するツール (旧来、コンソール)
isc-dhcp-client	V:231, I:979	686	設定:: 低レベル	DHCP クライアント
wpasupplicant	V:332, I:507	3436	, ,	WPA と WPA2 (IEEE 802.11i) のためのクライアントサポート
wpagui	V:0, I:2	781	, ,	wpa_supplicant の Qt GUI クライアント
wireless-tools	V:188, I:254	297	, ,	Linux のワイアレス拡張を操作するツール
ppp	V:206, I:474	1054	, ,	chat による PPP/PPPoE コネクション
pppoeconf	V:0, I:8	192	設定:: ヘルパー	PPPoE コネクションの設定ヘルパー
pppconfig	V:1, I:2	801	, ,	chat による PPP コネクションの設定ヘルパー
wvdial	V:0, I:5	249	, ,	wvdial と ppp による PPP コネクションの設定ヘルパー
			テスト:: 低	リモートホストへのネットワークパスを追跡する

5.1.1 ホスト名の解決

ホスト名の解決もまた、現在 NSS (ネームサービススイッチ、Name Service Switch) メカニズムによってサポートされています。この解決の流れは次です。

1. "`hosts: files dns`" のようなスタンザのある"`/etc/nsswitch.conf`" ファイルがホスト名の解消の順序を規定します。(これは、"`/etc/host.conf`" ファイル中の"`order`" スタンザの機能を置換します。)
2. `files` メソッドが最初に発動されます。ホスト名が"`/etc/hosts`" ファイルに見つかると、それに対応する全ての有効アドレスを返し終了します。("`/etc/host.conf`" ファイルは"`multi on`" を含みます。)
3. `dns` メソッドが発動されます。"`/etc/resolv.conf`" ファイルで識別されるインターネットドメイン名システム (DNS) への問い合わせでホスト名が見つかれば、それに関する全ての有効アドレスを返します。

例えば、"`/etc/hosts`" は以下の内容です。

```
127.0.0.1 localhost
127.0.1.1 <host_name>

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

各行は IP アドレス で始まり、関連する ホスト名がそれに続きます。

本例の 2 行目の IP アドレス `127.0.1.1` は他の Unix 系システムでは見かけないかもしれません。bug #719621 に記録されているように、Debian インストーラー は恒久的 IP アドレスのないシステムのために一部ソフトウエアー (GNOME 等) のための回避策としてこの項目を作成します。

<host_name> は、"`/etc/hostname`" の中に定義されたホスト名と一致します。

恒久的 IP アドレスを持つシステムでは `127.0.1.1` の代えてその恒久的 IP アドレスがここにあるべきです。

恒久的 IP アドレスと Domain 名システム (DNS)が提供する完全修飾ドメイン名 (FQDN) を持つシステムでは、その標準的な <host_name(ホスト名)>.<domain_name(ドメイン名)> が <host_name(ホスト名)> のみに代えて使われるべきです。

`resolvconf` パッケージがインストールされなかったら、"`/etc/resolv.conf`" は静的なファイルです。インストールされると、それはシンボリックリンクになります。いずれにせよ、解決機構を初期化する情報を含んでいます。もし DNS が IP="`192.168.11.1`" に見つかるなら、それは次の内容です。

```
nameserver 192.168.11.1
```

`resolvconf` パッケージはこの"`/etc/resolv.conf`" をシンボリックリンクにし、フックスクリプトで自動的にその内容を管理します。

典型的 adhoc な LAN 環境にある PC ワークステーションの場合、基本的な `files` や `dns` 法に加えて Multicast DNS (mDNS, Zeroconf) 経由でホスト名を解決する事ができます。

- Avahi は Debian で Multicast DNS サービスの探索の枠組みを提供します。
- Apple Bonjour / Apple Rendezvous と同等です。
- `libnss-mdns` プラグインパッケージが GNU C ライブラリー (glibc) の GNU Name Service Switch (NSS) 機能に mDNS 経由のホスト名解決を提供します。
- "`/etc/nsswitch.conf`" ファイルには"`hosts: files mdns4_minimal [NOTFOUND=return] dns mdns4`" のようなスタンザがあるべきです。

- ホスト名が ".local" で終わる 擬似-top-level domain (TLD) が解決されます。
- mDNS の IPv4 リンク-ローカルのマルチキャストアドレス"224.0.0.251" とか IPv6 でそれに相当する"FF02::FB" が".local"　で終わる名前の DNS クエリーに用いられます。

非推奨である NETBios over TCP/IP を使うホスト名解決は、winbind パッケージをインストールすると提供できます。このような機能を有効にするには、"/etc/nsswitch.conf" ファイル中に"hosts: files mdns4_minimal [NOTFOUND=return] dns mdns4 wins" のようなスタンザが必要です。(最近の Windows システムは通常 dns メソッドをホスト名の解決に使います。)

注意
ドメイン名システムにおける ジェネリックトップレベルドメイン (gTLD) の拡張が進行中です。LAN 内のみで使うドメイン名を選ぶ際に名前衝突に注意が必要です。

5.1.2　ネットワークインターフェース名

Linux カーネル中の各ハードウエアーは、それが見つかり次第ユーザー空間の設定メカニズム udev (項3.3参照下さい) を通じて、例えば eth0 のようなネットワークインターフェース名が付与されます。ネットワークインターフェース名は、ifup(8) と interfaces(5) の中で物理インターフェースと呼ばれています。

MAC アドレス等を使って各リブート毎に永続性をもって各ネットワークインターフェースが名付けられるようにするルールファイル"/etc/udev/rules.d/70-persistent-net.rules" があります。このファイルは"persistent-net-generator.rules" ルールファイルによって実行されているような"/lib/udev/write_net_r プログラムによって自動生成されます。そのファイルを変更することで命名ルールを変更できます。

注意
"/etc/udev/rules.d/70-persistent-net.rules" ルールファイルを編集する時は、各ルールを 1 行に納めて MAC アドレスに小文字を使わなければいけません。例えば、あなたがこのファイル中に"FireWire device" と"PCI device" を見つけたら、きっと"PCI device" を eth0 として第 1 番目のネットワークインターフェースとして設定したいでしょう。

5.1.3　LAN のためのネットワークアドレス範囲

rfc1918 によってローカルエリアネットワーク (LAN) での使用に予約されている各クラス毎の IPv4 32 ビットアドレス範囲を確認します。これらのアドレスは本来のインターネット上のアドレスとかち合う事が無いことが保証されています。

クラス	ネットワークアドレス	ネットマスク	ネットマスク / ビット	サブネットの数
A	10.x.x.x	255.0.0.0	/8	1
B	172.16.x.x —172.31.x.x	255.255.0.0	/16	16
C	192.168.0.x —192.168.255.x	255.255.255.0	/24	256

Table 5.2: ネットワークアドレス範囲のリスト

注意
これらのアドレス内の 1 つがホストに付与されている場合、そのホストはインターネットに直接アクセスせず、各サービスのプロキシとなるかネットワークアドレス変換 (NAT) をするゲートウエーを通してアクセスしなければいけません。ブロードバンドルーターは消費者 LAN 環境のために通常 NAT を行います。

5.1.4 ネットワークデバイスサポート

Debian システムによってほとんどのハードウエアーデバイスはサポートされていますが、一部のネットワークデバイスはそのサポートのために DFSG non-free のファームウエアーが必要です。項9.9.6を参照下さい。

5.2 デスクトップのためのモダンネットワーク設定

最近の systemd 下の Debian デスクトップ環境では、ネットワークインターフェースは、lo が"networking.service"で、他のインターフェースが"NetworkManager.service" で通常初期化されます。

Debian の squeeze 以降のシステム上では、NetworkManager (NM) (network-manager と関連パッケージ) や Wicd (wicd と関連パッケージ) 等の管理デーモン経由でネットワーク接続の管理ができます。

- それらには洒落た GUI やコマンドラインのユーザーインターフェースとともに提供されます。
- それらのバックエンドシステムとして、自前のデーモンとともに提供されます。
- それらによりあなたのシステムをインターネットへ容易に接続できます。
- それらによりインターネットへの有線や無線のネットワークの管理が容易にできます。
- それらにより旧来の"ifupdown" パッケージと独立にネットワークを設定できます。

注意
サーバーにはこの様な自動ネットワーク設定を使わないで下さい。これらはラップトップ上のモービルデスクトップを主対象としています。

これらの現代的なネットワーク設定ツールは旧来の"ifupdown" パッケージやその"/etc/network/interfaces" 設定ファイルとの競合を避けるように適正に設定する必要があります。

注意
これらの自動ネットワーク設定ツール機能の一部は、リグレッションにあっているかもしれません。これらは旧来の ifupdown パッケージほどは堅牢ではありません。最新の問題や制約条件に関しては network-manager の BTS や wicd の BTS を参照下さい。

5.2.1 GUI のネットワーク設定ツール

Debian 上での NM や Wicd に関する正式のドキュメンテーションは、"/usr/share/doc/network-manager/README.De
や"/usr/share/doc/wicd/README.Debian" によりそれぞれ提供されます。

デスクトップのための現代的ネットワーク設定の要点は以下です。

1. 次のようにして、例えば foo というデスクトップユーザーを"netdev" グループに属するようにします。(GNOME や KDE のような現代的デスクトップ環境の下では D-bus 経由でそれを自動的にするのも一つの方法です。)

```
$ sudo adduser foo netdev
```

2. "/etc/network/interfaces" の設定を次のようにできるだけ簡単にします。

```
auto lo
iface lo inet loopback
```

3. 次のようにして NM や Wicd を再起動します。

```
$ sudo /etc/init.d/network-manager restart
```

```
$ sudo /etc/init.d/wicd restart
```

4. GUI 経由でネットワークを設定します。

注意
ifupdown との干渉を避けるために、"/etc/network/interfaces" にリストされてないインターフェースのみが NM もしくは Wicd によって管理されます。

ティップ
NM のネットワーク設定能力を拡張したい場合には、network-manager-openconnect や network-manager-openvpn-gnome や network-manager-pptp-gnome や mobile-broadband-provider-info や gnome-bluetooth 等の適正なプラグインモジュールや補足パッケージを探してください。Wicd の場合もまったく同様の方法で対処します。

注意
こういった自動ネットワーク設定は、項5.6や項5.7のような"/etc/network/interfaces" への凝った旧来の ifupdown 設定と互換性がないかも知れません。最新の問題や制約条件に関しては network-manager の BTS や wicd の BTS を参照下さい。

5.3 GUI 無しのモダンネットワーク設定

上記とは異なり、systemd の下では、ネットワークは /etc/systemd/network/ を使って設定されているかもしれません。systemd-resolved(8) や resolved.conf(5) や systemd-networkd(8) を参照ください。

これにより GUI 無しのモダンネットワーク設定ができます。

DHCP クライアントの設定は"/etc/systemd/network/dhcp.network" を作成することで設定できます。例えば:

```
[Match]
Name=en*

[Network]
DHCP=yes
```

静的ネットワーク設定は"/etc/systemd/network/static.network" を作成することで設定できます。例えば:

```
[Match]
Name=en*

[Network]
Address=192.168.0.15/24
Gateway=192.168.0.1
```

5.4 旧来のネットワーク接続や設定

項5.2に記載された手法ではあなたの目的にとって不十分な場合には、多くの簡単なツールを組み合わせる旧来のネットワーク接続や設定を使うべきです。

旧来のネットワーク接続は手法毎に特定です (項5.5参照下さい)。

Linux の低レベルネットワーク設定には 2 タイプのプログラムがあります (項5.8.1参照下さい)。

- 旧式の net-tools プログラム (ifconfig(8)、…) は Linux NET-3 ネットワークシステム由来です。これらの多くはすでに型遅れです。
- 新規の Linux iproute2 プログラム (ip(8)、…) は現行の Linux のネットワークシステムです。

これらの低レベルのネットワークプログラムは強力ですが、使うのが面倒です。そこで高レベルのネットワーク設定プログラムが作られました。

`ifupdown` パッケージは Debian 上のそのような高レベルネットワーク設定のデファクトスタンダードです。それを使うと、”`ifup eth0`” とするだけでネットワークを立ち上げることができます。その設定ファイルは、”`/etc/network/interfaces`” ファイルで、その典型的内容は次です。

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp
```

`resolvconf` パッケージは、ネットワークアドレス解決設定ファイル”`/etc/resolv.conf`” の書き変えを自動化してネットワークアドレス解決の円滑な再構成をサポートするために `ifupdown` システムを補完するために作られました。現在、ほとんどの Debian のネットワーク設定パッケージは `resolvconf` パッケージを使うように変更されています (”`/usr/share/doc/resolvconf/README.Debian`” 参照下さい)。

`ifplugd` や `guessnet` や `ifscheme` 等の `ifupdown` パッケージの補助スクリプトが有線 LAN 上の可動 PC のためのネットワーク環境設定のようなネットワーク環境の動的設定を自動化するために作られました。これらはちょっと使い難いですが、既存の `ifupdown` システムとは反りが合います。

詳細に例示とともに説明します (項5.6と項5.7参照下さい)。

5.5 ネットワーク接続方法 (旧来)

注意
本セクションで説明されている接続テスト方法はあくまでテスト目的のためのものです。毎日のネットワーク接続のために使うためではありません NM か Wicd か `ifupdown` パッケージを通して使うことをお勧めします (項5.2と項5.6参照下さい)。

典型的なネットワーク接続方法と PC までの接続経路は次のようにまとめられます。

各接続方法のための設定スクリプトのまとめて次に記します。

ネットワーク接続の省略語は次の意味です。

注意
ケーブル TV 経由の WAN 接続サービスは大体 DHCP か PPPoE でサービスを受けます。ADSL と FTTP の接続サービスは大体 PPPoE でサービスを受けます。WAN 接続の正確な設定要件はあなたの ISP にご確認下さい。

PC	接続方法	接続経路
シリアルポート (ppp0)	PPP	⇔ モデム ⇔ POTS ⇔ ダイヤルアップアクセスポイント ⇔ ISP
イーサーネットポート (eth0)	PPPoE/DHCP/静的	⇔ BB モデム ⇔ BB サービス ⇔ BB アクセスポイント ⇔ ISP
イーサーネットポート (eth0)	DHCP/静的	⇔ LAN ⇔ ネットワークアドレス変換 (NAT) 機能のある BB ルーター (⇔ BB モデム…)

Table 5.3: 典型的なネットワーク接続方法と接続経路のリスト

接続方法	設定	バックエンドパッケージ
PPP	pppconfig 決定論的 chat の作成	pppconfig と ppp
PPP (代替)	wvdialconf はヒューリスティックのある chat の作成	ppp と wvdial
PPPoE	pppoeconf 決定論的 chat の作成	pppoeconf と ppp
DHCP	"/etc/dhcp/dhclient.conf" 中に記述されています	isc-dhcp-client
静的 IP (IPv4)	"/etc/network/interfaces" 中に記述されています	iproute もしくくは net-tools (型遅れ)
静的 IP (IPv6)	"/etc/network/interfaces" 中に記述されています	iproute

Table 5.4: ネットワーク接続設定のリスト

省略語	意味
POTS	旧来型電話サービス
BB	ブロードバンド
BB サービス	例: デジタルサブスクラーバーライン (DSL) やケーブルテレビや家庭向け光ファイバー (FTTP)
BB モデム	例: DSL モデムやケーブルモデムや光ファイバー端末 (ONT)
LAN	ローカルエリアネットワーク
WAN	広域ネットワーク
DHCP	動的ホスト設定プロトコル
PPP	ポイント間接続プロトコル
PPPoE	Ethernet 経由のポイント間接続プロトコル
ISP	インターネットサービス供給者

Table 5.5: ネットワーク接続の省略語のリスト

注意
BB ルータが家庭内 LAN 環境を作るのに使われる時には、LAN 上の PC はネットワークアドレス変換 (NAT) をして BB ルーター経由で WAN に接続されます。そのような場合、LAN 上の PC のネットワークインターフェースは静的 IP か BB ルーターからの DHCP でサービスを受けます。BB ルーターはあなたの ISP による指示にしたがって WAN に接続されるよう設定しなければいけません。

5.5.1 イーサーネットを使っての DHCP 接続

典型的な現代的な家庭内や小規模ビジネスネットワークの LAN は WAN (インターネット) に何らかの消費者向けブロードバンドルーターを使って接続されています。このルーターの後ろの LAN は通常ルーター上で稼働する動的ホスト設定プロトコル (DHCP) サーバーによりサービスを受けています。

動的ホスト設定プロトコル (DHCP) サービスを受けるイーサーネットでは、`isc-dhcp-client` パッケージをインストールするだけです。

dhclient.conf(5) を参照下さい。

5.5.2 イーサーネットを使っての静的 IP 接続

静的 IP サービスを受けるイーサーネットでは、特段何をする必要もありません。

5.5.3 pppconfig を使っての PPP 接続

設定スクリプト `pppconfig` はPPP 接続を次の選択をすることで対話式で設定します。

- 電話番号
- ISP でのユーザー名
- ISP のパスワード
- ポートの速度
- モデム通信のポート
- 認証方法

ファイル	機能
`/etc/ppp/peers/<isp_name>`	`pppconfig` によって生成された <isp_name> に特定な `pppd` のための設定ファイル
`/etc/chatscripts/<isp_name>`	`pppconfig` によって生成された <isp_name> に特定な `chat` のための設定ファイル
`/etc/ppp/options`	`pppd` のための一般的な実行パラメーター
`/etc/ppp/pap-secret`	PAP のための認証データー (安全上問題あり)
`/etc/ppp/chap-secret`	CHAP のための認証データー (比較的安全)

Table 5.6: pppconfig を使っての PPP 接続のための設定ファイルのリスト

注意
もし `pon` と `poff` コマンドが引数無しに起動された場合には、”<isp_name>” の値として”`provider`” が使われます。

低レベルのネットワーク設定ツールを使って次に記すように設定の確認ができます。

```
$ sudo pon <isp_name>
...
$ sudo poff <isp_name>
```

”/usr/share/doc/ppp/README.Debian.gz” を参照下さい。

5.5.4 wvdialconf を使った代替 PPP 接続

pppd(8) を使う異なるアプローチは、wvdial パッケージが提供する wvdial(1) からそれを実行することです。電話を掛け接続の交渉をするために pppd が chat(8) を実行するのではなく、wvdial が電話を掛け接続の交渉をした後に pppd を始動し後を任せます。

設定スクリプト wvdialconf は PPP 接続を次の選択をすることで対話式で設定します。

- 電話番号
- ISP でのユーザー名
- ISP のパスワード

ほとんどの場合 wvdial は接続をすることに成功し、自動的に認証データーのリストを管理します。

ファイル	機能
/etc/ppp/peers/wvdial	wvdialconf が wvdial に合わせて生成した pppd の設定ファイル
/etc/wvdial.conf	wvdialconf が生成した設定ファイル
/etc/ppp/options	pppd のための一般的な実行パラメーター
/etc/ppp/pap-secret	PAP のための認証データー (安全上問題あり)
/etc/ppp/chap-secret	CHAP のための認証データー (比較的安全)

Table 5.7: wvdialconf で PPP 接続する際の設定ファイルリスト

低レベルのネットワーク設定ツールを使って次に記すように設定の確認ができます。

```
$ sudo wvdial
...
$ sudo killall wvdial
```

wvdial(1) と wvdial.conf(5) を参照下さい。

5.5.5 pppoeconf を使った PPPoE 接続

あなたの ISP が PPPoE 接続を提供し、あなたの PC を直接 WAN に接続すると決めた際には、あなたの PC のネットワークは PPPoE で設定しないといけません。PPPoE とはイーサーネット経由の PPP の意味です。設定スクリプト pppoeconf は PPPoE 接続を対話式で設定します。

設定ファイルは以下。

低レベルのネットワーク設定ツールを使って次に記すように設定の確認ができます。

```
$ sudo /sbin/ifconfig eth0 up
$ sudo pon dsl-provider
...
$ sudo poff dsl-provider
$ sudo /sbin/ifconfig eth0 down
```

”/usr/share/doc/pppoeconf/README.Debian” を参照下さい。

ファイル	機能
/etc/ppp/peers/dsl-provider	pppoeconf が pppoe に合わせて生成した pppd の設定ファイル
/etc/ppp/options	pppd のための一般的な実行パラメーター
/etc/ppp/pap-secret	PAP のための認証データー (安全上問題あり)
/etc/ppp/chap-secret	CHAP のための認証データー (比較的安全)

Table 5.8: pppoeconf を用いて PPPoE 接続する際の設定ファイルのリスト

5.6 ifupdown を使った基本的なネットワーク設定 (旧来)

Debian システム上の伝統的な TCP/IP ネットワークのセットアップは ifupdown パッケージをより高レベルのツールとして使います。2 つの場合があります。

- 可動 PC のような動的 **IP** システムの場合、TCP/IP ネットワークを resolvconf パッケージを使い設定し、容易にネットワーク設定を切り替えられるようにしましょう (項5.6.4参照下さい)。
- サーバーのような静的 **IP** システムの場合、TCP/IP ネットワークを resolvconf パッケージを使うことなく設定し、システムを単純にしましょう (項5.6.5参照下さい)。

このような伝統的設定方法は上級設定をしたい際に非常に有用です。以下を参照下さい。

ifupdown パッケージは Debian システムでの高レベルネットワーク設定の標準化された枠組みを提供します。このセクションでは、簡略化された紹介と多くの典型例で ifupdown を使った基本的なネットワーク設定を学びます。

5.6.1 簡略化されたコマンドシンタックス

ifupdown パッケージには 2 つのコマンドがあります: ifup(8) と ifdown(8)。設定ファイル”/etc/network/interfaces” により規定される高レベルのネットワーク設定を提供します。

コマンド	アクション
ifup eth0	”iface eth0” スタンザがあれば、ネットワークインターフェース eth0 をネットワーク設定 eth0 で起動
ifdown eth0	”iface eth0” スタンザがあれば、ネットワークインターフェース eth0 に関するネットワーク設定 eth0 を終了し停止

Table 5.9: ifupdown を使う基本的なネットワーク設定コマンドのリスト

警告
up 状態にあるインターフェースを、ifconfig(8) や ip(8) コマンドのような低レベル設定ツールを使って設定してはいけません。

注意
ifupdown というコマンドはありません。

5.6.2 ”/etc/network/interfaces” の基本的なシンタックス

”/etc/network/interfaces” のシンタックスは interfaces(5) に説明されていて、要点を次に記します。

iface スタンザで始まる行は次のシンタクスです。

スタンザ	意味
”auto <interface_name>”	システム起動時にインターフェース <interface_name> を起動
”allow-auto <interface_name>”	,,
”allow-hotplug <interface_name>”	カーネルがインターフェースからのホットプラグイベントを認知した際にインターフェース <interface_name> を起動
”iface <config_name> …” によって始まる行	ネットワーク設定 <config_name> を規定
”mapping <interface_name_glob>” によって始まる行	<interface_name> に対応する <config_name> の mapping 値を規定
ハッシュ”#” で始まる行	コメントして無視 (行末コメントは非サポート)
バックスラシュ”\” で終わる行	設定を次行に継続

Table 5.10: ”/etc/network/interfaces” のスタンザのリスト

```
iface <config_name> <address_family> <method_name>
 <option1> <value1>
 <option2> <value2>
 ...
```

基本設定に関しては、**mapping** スタンザは使われませんし、ネットワークインターフェース名をネットワーク設定名として使います (項5.7.5参照下さい)。

警告
”/etc/network/interfaces” のネットワークインターフェースに関する”iface” スタンザの重複定義をしてはいけません。

5.6.3 ループバックネットワークインターフェース

”/etc/network/interfaces” ファイルに次の設定をすることでシステムブート時にループバックネットワークインターフェース lo が起動されます (**auto** スタンザ経由)。

```
auto lo
iface lo inet loopback
```

この設定は、”/etc/network/interfaces” ファイル中にいつも存在します。

5.6.4 DHCP サービスを受けるネットワークインターフェース

項5.5.1によりシステムの準備をした後、DHCP によってサービスされるネットワークインターフェースは”/etc/network/
ファイルの中に次のような設定エントリーを設定します。

```
allow-hotplug eth0
iface eth0 inet dhcp
```

Linux カーネルが物理インターフェース eth0 を認識した場合、**allow-hotplug** スタンザは ifup によりそのインターフェースが起動させられるようにし、**iface** スタンザが ifup がインターフェースが DHCP を使うように設定します。

5.6.5 静的 IP を使うネットワークインターフェース

静的 IP によってサービスされるネットワークインターフェースは”/etc/network/interfaces” ファイル中に例えば次の設定エントリーを作ることで設定されます。

```
allow-hotplug eth0
iface eth0 inet static
 address 192.168.11.100
 netmask 255.255.255.0
 gateway 192.168.11.1
 dns-domain example.com
 dns-nameservers 192.168.11.1
```

Linux カーネルが物理インターフェース eth0 を認識した場合、**allow-hotplug** スタンザは ifup によりそのインターフェースが起動させられるようにし、**iface** スタンザが ifup がインターフェースが静的 IP を使うように設定します。

ここでは次を仮定しています。

- LAN ネットワークの IP アドレス範囲: 192.168.11.0 - 192.168.11.255
- ゲートウエーの IP アドレス: 192.168.11.1
- PC の IP アドレス: 192.168.11.100
- resolvconf パッケージ: インストール済み
- ドメイン名: ”example.com”
- DNS サーバーの IP アドレス: 192.168.11.1

resolvconf パッケージがインストールされていないと、DNS 関係の設定は手動で”/etc/resolv.conf” を次のように編集する必要があります。

```
nameserver 192.168.11.1
domain example.com
```

注意
上記例で用いた IP アドレスはそのままコピーされるべきものではありません。IP 番号はあなたの実際のネットワーク設定に合わせなければいけません。

5.6.6 ワイアレス LAN インターフェースの基本

ワイアレス LAN (省略すると WLAN) は IEEE 802.11 という標準群に基づく特別な免許なく使える無線を使う帯域拡散通信を経由の高速ワイアレス接続を提供します。

WLAN インターフェースは通常の Ethernet インターフェースと非常に似ていますが、始動時にネットワーク ID と暗号キーデーターを必要とします。インターフェース名が使われるカーネルドライバーによって eth1 や wlan0 や ath0 や wifi0 等とインターフェース名が少々異なる以外は、それらに使われる高レベルのネットワークツールは Ethernet インターフェースのものとまったく同じです。

ティップ
wmaster0 デバイスは、新規の Linux の mac80211 API による SoftMAC によってのみ使われる内部デバイスのマスターデバイスです。

WLAN に関して留意すべきキーワードは次です。

実際のプロトコルの選択肢は採用しているワイアレスルーターによって普通制約されます。

省略語	元の言葉	意味
NWID	ネットワーク ID	802.11 以前の WaveLAN ネットワークによって使われる 16 ビットネットワーク ID (非常に非推奨)
(E)SSID	(拡張) サービスセットアイデンティファイアー	802.11 ワイアレス LAN が統合されて連結されてできるワイアレスアクセスポイント (APs) のネットワーク名、ドメイン ID
WEP, (WEP2)	有線等価プライバシー	第 1 世代の、40 ビットのキーを使う 64 ビット (128 ビット) のワイアレス暗号化標準 (非推奨)
WPA	Wi-Fi 保護アクセス (Protected Access)	第 2 世代の、WEP との互換性のあるワイアレス暗号化標準 (802.11i の殆ど)
WPA2	Wi-Fi 保護アクセス 2 (Protected Access 2)	第 3 世代の、WEP との互換性のないワイアレス暗号化標準 (完全に 802.11i)

Table 5.11: WLAN の略語のリスト

5.6.7 WPA/WPA2 を使うワイアレス LAN インターフェース

新規の WPA/WPA2 を使う WLAN をサポートするには、`wpasupplicant` パッケージをインストールする必要があります。

WLAN 接続上の DHCP によってサービスされている IP の場合には、”`/etc/network/interfaces`” ファイルのエントリーは次のようです。

```
allow-hotplug ath0
iface ath0 inet dhcp
 wpa-ssid homezone
 # hexadecimal psk is encoded from a plaintext passphrase
 wpa-psk 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
```

”`/usr/share/doc/wpasupplicant/README.modes.gz`” を参照下さい。

5.6.8 WEP を使うワイアレス LAN インターフェース

旧式の WEP を使う WLAN をサポートするには、`wireless-tools` パッケージをインストールする必要があります。(あなたの消費者用のルーターは今だにセキュアーでないインフラを使っているかもしれませんが、無いよりはましです。)

注意
WEP を使う WLAN 上のネットワークトラフィックは他人に覗かれているかも知れないことを覚えておいて下さい。

WLAN 接続上の DHCP によってサービスされている IP の場合には、”`/etc/network/interfaces`” ファイルのエントリーは次のようです。

```
allow-hotplug eth0
iface eth0 inet dhcp
 wireless-essid Home
 wireless-key1 0123-4567-89ab-cdef
 wireless-key2 12345678
 wireless-key3 s:password
 wireless-defaultkey 2
 wireless-keymode open
```

”`/usr/share/doc/wireless-tools/README.Debian`” を参照下さい。

5.6.9 PPP 接続

以前説明したように PPP 接続を設定する必要があります (項5.5.3参照下さい)。さらに、第 1 番目の PPP デバイス `ppp0` のための”`/etc/network/interfaces`” ファイルのエントリーを次のように追加します。

```
iface ppp0 inet ppp
 provider <isp_name>
```

5.6.10 代替の PPP 接続

以前説明したように `wvdial` を使う代替 PPP 接続をまず設定する必要があります (項5.5.4参照下さい)。さらに、第 1 番目の PPP デバイス `ppp0` のための”`/etc/network/interfaces`” ファイルのエントリーを次のように追加します。

```
iface ppp0 inet wvdial
```

5.6.11 PPPoE 接続

PPPoE によってサービスされる直接 WAN に接続した PC の場合、以前説明したように PPPoE 接続を設定する必要があります (項5.5.5参照下さい)。さらに、第 1 番目の PPPoE デバイス `eth0` のための”`/etc/network/interfaces`” ファイルのエントリーを次のように追加します。

```
allow-hotplug eth0
iface eth0 inet manual
 pre-up /sbin/ifconfig eth0 up
 up ifup ppp0=dsl
 down ifdown ppp0=dsl
 post-down /sbin/ifconfig eth0 down
# The following is used internally only
iface dsl inet ppp
 provider dsl-provider
```

5.6.12 ifupdown のネットワーク設定状態

”`/etc/network/run/ifstate`” ファイルは、`ifupdown` パッケージが管理する全ての有効なネットワークインターフェースの意図したネットワーク設定状態を記録します。残念ながら `ifupdown` が意図したにもかかわらずインターフェースを起動できなかった場合でも、”`/etc/network/run/ifstate`” ファイルはインターフェースが起動されたと表示します。

あるインターフェースに関する ifconfig(8) コマンド出力が次の例のような行を欠いている場合、IPV4 ネットワークの一部としては使えません。

```
  inet addr:192.168.11.2  Bcast:192.168.11.255  Mask:255.255.255.0
```

注意
PPPoE に接続されているイーサーネットデバイスの場合、ifconfig(8) コマンドのアウトプットは上記例のような行を欠いています。

5.6.13 基本ネットワーク設定

例えば eth0 というインターフェースを再設定したい際には、まず”**sudo ifdown eth0**” コマンドを実行してそれを無効にしなければいけません。こうすることで eth0 のエントリーが”/etc/network/run/ifstate” ファイルから削除されます。(もし eth0 が有効でなかったりそれにたいして過去に間違った設定がされている際には、こうするとエラーメッセージを発することになるかもしれません。シンプルな単一ユーザーのワークステーションではいつも問題なことは知る限り起こらないようです。)

こうすることで、必要に応じてネットワークインターフェース eth0 を自由に再設定するために”/etc/network/interfa
の内容を自由に書き換えられます。

こうした後で、eth0 を”**sudo ifup eth0**” コマンドを使って再起動できます。

ティップ
ネットワークインターフェースは、”**sudo ifdown eth0;sudo ifup eth0**” とすることで (再) 初期化できます。

5.6.14 ifupdown-extra パッケージ

ifupdown-extra パッケージは、ifupdown とともに使う使いやすいネットワーク接続テストを提供します。

- network-test(1) コマンドはシェルから実行できます。
- 自動スクリプトは各 ifup コマンド実行毎に実行されます。

network-test コマンドをつかうことで面倒な低レベルコマンドを使ってネットワーク問題を分析する手間を省けます。

自動スクリプトは”/etc/network/*/” にインストールされ、次の機能があります。

- ネットワークケーブルの接続の確認
- IP アドレスの重複使用の確認
- ”/etc/network/routes” の定義に従った静的ルートの設定
- ネットワークのゲートウエーが到達可能かの確認
- 結果を”/var/log/syslog” ファイルに記録

この syslog 記録はリモートシステムのネットワーク問題を管理する上で非常に有用です。

ティップ
ifupdown-extra パッケージの自動的な挙動は”/etc/default/network-test” によって設定可能です。これらの自動テストの一部は ARP からの返答を聞くのでシステムのブートプロセスを少々遅らせます。

5.7 ifupdown を使う上級ネットワーク設定 (旧来)

ifupdown パッケージの機能は、上級知識を使うと項5.6に書かれているよりも向上します。

ここに記述されている機能は全く任意のものです。著者自身、怠け者で面倒な事が嫌いなために、ここに書かれたことを使うことは滅多にありません。

注意
項5.6に書かれた情報でネットワーク接続をうまく設定できないのに、次の情報を使うと状況は更に悪くなります。

5.7.1 ifplugd パッケージ

ifplugd パッケージはイーサーネット接続のみを管理する旧式の自動ネットワーク設定ツールです。これによって可動 PC 等のイーサーネットケーブルの脱着問題を解決します。もし NetworkManager か Wicd (項5.2参照下さい) がインストールされている場合は、このパッケージは必要ありません。

このパッケージはデーモンとして実行され、**auto** とか **allow-hotplug** という機能 (表 5.10) を置き換え、インターフェースがネットワークに繋がれるとインターフェースを起動します。

例えば eth0 という内部イーサーネットポートに対する ifplugd パッケージの利用方法は次です。

1. "/etc/network/interfaces" の中のスタンザを削除しましょう: "auto eth0" または"allow-hotplug eth0"、
2. "/etc/network/interfaces" の中のスタンザを残しましょう: "iface eth0 inet …" と"mapping …"、
3. ifplugd パッケージをインストールします。
4. "sudo dpkg-reconfigure ifplugd" の実行します。
5. eth0 を"ifplugd により監視される静的インターフェース" とします。

こうするとお望みどおりのネットワーク設定が機能します。

- 電源投入もしくはハードウエアーの発見時に、インターフェースは自動的に起動されます。
  - 長い DHCP のタイムアウトを待つことのない迅速なブートプロセス。
  - 適正な IPv4 アドレス無いまま起動された変なインターフェースが無いこと (項5.6.12参照下さい)。
- イーサーネットケーブルを発見時にインターフェースが起動されます。
- イーサーネットケーブルを外して少し経った時点でインターフェースが自動的に停止されます。
- イーサーネットケーブルを接続した時点でインターフェースが新規ネットワーク環境下で起動されます。

ティップ
ifplugd(8) コマンドの引数はインターフェースの再設定の遅延時間などの挙動を設定します。

5.7.2 ifmetric パッケージ

ifmetric パッケージを使うと、DHCP でもルートのメトリクスを事後操作できます。

次のようにすると eth0 インターフェースを wlan0 インターフェースより優先するように設定できます。

1. ifmetric パッケージをインストールします。
2. "/etc/network/interfaces" 中の"iface eth0 inet dhcp" 行のすぐ下に"metric 0" というオプション行を追加します。
3. "/etc/network/interfaces" 中の"iface wlan0 inet dhcp" 行のすぐ下に"metric 1" というオプション行を追加します。

メトリック 0 とは最高優先順位のルートでデフォールトのルートということを意味します。大きなメトリック値は低い優先順位を意味します。最低のメトリック値をもつ有効なインターフェースの IP アドレスが発信源となるインターフェースになります。ifmetric(8) を参照下さい。

5.7.3 仮想インターフェース

物理的には単一のイーサーネットインターフェースは異なる IP アドレスをもつ複数の仮想インターフェースとして設定できます。いくつかの IP サブネットワークにインターフェースを繋ぐのが通常こうする目的です。例えば、単一ネットワークインターフェースを使った IP アドレスに基づく仮想ウェッブホスティングがその適用例です。

例えば、次を仮定します。

- あなたのホスト上の単一のイーサーネットインターフェースが (ブロードバンドルーターではなく) イーサーネットハブに接続されています。
- イーサーネットハブはインターネットと LAN ネットワークの両方に接続されています。
- LAN ネットワークはサブネット `192.168.0.x/24` を使います。
- あなたのホストはインターネットに関しては物理インターフェース `eth0` を DHCP が提供する IP アドレスで使います。
- あなたのホストは LAN に関しては仮想インターフェース `eth0:0` を `192.168.0.1` で使います。

このとき”`/etc/network/interfaces`” 中の次のスタンザがあなたのネットワークを設定します。

```
iface eth0 inet dhcp
 metric 0
iface eth0:0 inet static
 address 192.168.0.1
 netmask 255.255.255.0
 network 192.168.0.0
 metric 1
```

注意
netfilter/iptables (項5.10参照下さい) を使ってネットワークアドレス変換 (NAT) を使う上記設定例は単一インターフェースを使って LAN に対して安価なルーターを提供しますが、そのような設定を使ったのでは真のファイアーウォール能力はありません。2 つの物理インターフェースと NAT を使ってインターネットからローカルネットワークをセキュアーするべきです。

5.7.4 上級コマンドシンタックス

`ifupdown` パッケージはネットワーク設定名とネットワークインターフェース名を使って上級ネットワーク設定をできるようにします。わたしは ifup(8) や interfaces(5) とは少々異なる用語法をここでは使っています。

マンページの用語法	著者の用語法	この後の文中の用例	説明
物理インターフェース名	ネットワークインターフェース名	`lo`, `eth0`, `<interface_name>`	Linux カーネルが (udev メカニズムを利用して) 与えた名前
論理インターフェース名	ネットワーク設定名	`config1`, `config2`, `<config_name>`	”`/etc/network/interfaces`” 中で **`iface`** に続く名前のトークン

Table 5.12: ネットワークデバイスの用語法のリスト

項5.6.1中の基本的なネットワーク設定コマンドは、**`iface`** スタンザ中のネットワーク設定名のトークンと、”`/etc/network/interfaces`” 中のネットワークインターフェース名が一致している必要があります。

上級ネットワーク設定コマンドは次のような”`/etc/network/interfaces`” の中で、ネットワーク設定名とネットワークインターフェース名を区別を可能にします。

コマンド	アクション
`ifup eth0=config1`	ネットワーク設定 `config1` を使うネットワークインターフェース `eth0` を始動
`ifdown eth0=config1`	ネットワーク設定 `config1` を使うネットワークインターフェース `eth0` を停止
`ifup eth0`	**`mapping`** スタンザによって選ばれる設定を使ってネットワークインターフェース `eth0` を始動
`ifdown eth0`	**`mapping`** スタンザによって選ばれる設定を使ってネットワークインターフェース `eth0` を停止

Table 5.13: ifupdown を使う上級ネットワーク設定コマンドのリスト

5.7.5 mapping スタンザ

項5.6.2では複雑になるのを避けるために”`/etc/network/interfaces`” 中の **`mapping`** スタンザを説明しませんでした。このスタンザには次のシンタクスがあります。

```
mapping <interface_name_glob>
 script <script_name>
 map <script_input1>
 map <script_input2>
 map ...
```

上記は、`<script_name>` で指定される mapping スクリプトで設定の選択を自動化することで”`/etc/network/interfac`
に上級機能を付与します。

次の実行を追いかけましょう。

```
$ sudo ifup eth0
```

”`<interface_name_glob>`” が”`eth0`” と一致する時に、この実行は自動的に `eth0` を設定する次のコマンドの実行を引き起こします。

```
$ sudo ifup eth0=$(echo -e '<script_input1> \n <script_input2> \n ...' | <script_name> eth0 ↩
    )
```

ここで、”`map`” を含む行は任意で反復可です。

注意
`mapping` スタンザのグロッブはシェルのファイル名グロブのように機能します (項1.5.6参照下さい)。

5.7.6 手動切り替え可能なネットワーク設定

項5.6.13でしたように”`/etc/network/interfaces`” ファイルを書き直すことなくいくつかのネットワーク設定間を手動で切り替える方法を次に示します。

アクセスする必要のある全てのネットワーク設定のついて、”`/etc/network/interfaces`” ファイル中に次に示すような個別のスタンザを作ります。

```
auto lo
iface lo inet loopback

iface config1 inet dhcp

iface config2 inet static
 address 192.168.11.100
```

```
 netmask 255.255.255.0
 gateway 192.168.11.1
 dns-domain example.com
 dns-nameservers 192.168.11.1

iface pppoe inet manual
 pre-up /sbin/ifconfig eth0 up
 up ifup ppp0=dsl
 down ifdown ppp0=dsl
 post-down /sbin/ifconfig eth0 down

# The following is used internally only
iface dsl inet ppp
 provider dsl-provider

iface pots inet ppp
 provider provider
```

iface の後にあるトークンのネットワーク設定名に、ネットワークインターフェース名のトークンを使っていないことに注目下さい。また、何らかのイベントの際にネットワークインターフェース eth0 を自動的に起動する **auto** スタンザも **allow-hotplug** スタンザもありません。

さあ、ネットワーク設定を切り替える準備完了です。

あなたの PC を DHCP が提供される LAN に移動します。次のようにしてネットワークインターフェース (物理インターフェース) eth0 にネットワーク設定名 (論理インターフェース名) config1 を付与してそれを起動します。

```
$ sudo ifup eth0=config1
Password:
...
```

インターフェース eth0 が起動され、DHCP で設定され、LAN に接続されます。

```
$ sudo ifdown eth0=config1
...
```

インターフェース eth0 が停止され、LAN から切断されます。

あなたの PC を静的 IP が提供される LAN に移動します。次のようにしてネットワークインターフェース (物理インターフェース) eth0 にネットワーク設定名 (論理インターフェース名) config2 を付与してそれを起動します。

```
$ sudo ifup eth0=config2
...
```

インターフェース eth0 が起動され、静的 IP で設定され、LAN に接続されます。dns-* で与えられる追加パラメーターが”/etc/resolv.conf” の内容を設定します。この”/etc/resolv.conf” は resolvconf パッケージがインストールされている方がうまく管理されます。

```
$ sudo ifdown eth0=config2
...
```

インターフェース eth0 が停止され、LAN から再度切断されます。

あなたの PC を PPPoE が提供されているサービスに繋がっている BB モデムのポートに移動します。次のようにしてネットワークインターフェース eth0 にネットワーク設定名 pppoe を付与してそれを起動します。

```
$ sudo ifup eth0=pppoe
...
```

インターフェース eth0 が起動され、ISP に直接接続された PPPoE で設定されます。

```
$ sudo ifdown eth0=pppoe
...
```

インターフェース eth0 が停止され再切断されます。

あなたの PC を LAN や BB モデムのない POTS とモデムを使っている場所に移動します。次のようにしてネットワークインターフェース ppp0 にネットワーク設定名 pots を付与してそれを起動します。

```
$ sudo ifup ppp0=pots
...
```

インターフェース ppp0 が起動され、PPP を使ってインターネットに接続されます。

```
$ sudo ifdown ppp0=pots
...
```

インターフェース ppp0 が停止され再切断されます。

ifupdown システムのネットワーク設定状態の現状は”/etc/network/run/ifstate” ファイルの内容で確認します。

警告
複数のネットワークインターフェースがある場合には、eth* や ppp* 等の最後の数字を調整する必要があります。

5.7.7 ifupdown システムを使うスクリプト

ifupdown システムはスクリプトに環境変数を引き渡して”/etc/network/*/” 中にインストールされたスクリプトを自動実行します。

環境変数	引き渡す変数値
”$IFACE”	処理対象のインターフェースの物理名 (インターフェース名)
”$LOGICAL”	処理対象のインターフェースの論理名 (設定名)
”$ADDRFAM”	インターフェースの <address_family>
”$METHOD”	インターフェースの <method_name> (例えば”static”)
”$MODE”	ifup から実行されると”start”、ifdown から実行されると”stop”
”$PHASE”	”$MODE” と同じ、ただし pre-up と post-up と pre-down と post-down 段階を詳細に区別
”$VERBOSITY”	”--verbose” 使用の指標; 使用されたら 1、使用されなかったら 0
”$PATH”	コマンドサーチパス: ”/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbi
”$IF_<OPTION>”	**iface** スタンザ下の対応するオプションの値

Table 5.14: ifupdown システムが引き渡す環境変数のリスト

各環境変数”$IF_<OPTION>” は、対応する <option1> や <option2> オプションの名前に”$IF_” を付け、大文字に変換し、ハイホンを下線に変換し、英数字文字以外を捨てて作成されます。

ティップ
<address_family> や <method_name> や <option1> や <option2> に関しては項5.6.2を参照下さい。

ifupdown-extra パッケージ (項5.6.14参照下さい) はこのような環境変数を使って ifupdown パッケージの機能拡張をします。ifmetric パッケージ (項5.7.2参照下さい) は”$IF_METRIC” 変数を通してメトリック値を設定する”/etc/network/if-up.d/ifmetric” スクリプトをインストールします。ネットワーク設定の自動選択の簡単で強力な枠組みを提供している guessnet パッケージ (項5.7.8参照下さい) もまたこれらを使います。

注意
これらの環境変数を使うネットワーク設定スクリプトのより具体的な例に関しては、”/usr/share/doc/ifupdown/examples/*” 中の例示スクリプトや ifscheme と ifupdown-scripts-zg2 パッケージで使われているスクリプトを確認して下さい。このような追加スクリプトは基本的な ifupdown-extra や guessnet パッケージと機能の重複があります。このような追加スクリプトをインストールしたら、干渉しないようにこのようなスクリプトをカスタム化するべきです。

5.7.8 guessnet を使う mapping

項5.7.6に記述されたように手動で設定選択する代わりに、項5.7.5に記述された mapping メカニズムをつかってカスタムスクリプトでネットワーク設定を自動的に選択できます。

guessnet パッケージにより提供される guessnet-ifupdown(8) コマンドは mapping スクリプトとして使われるように設計されており、ifupdown システムを拡張する強力な枠組みを提供します。

- **iface** スタンザの下の各ネットワーク設定毎に **guessnet** オプションの値としてテスト条件をリストします。
- mapping は最初のエラーを返さない **iface** をネットワーク設定として選択します。

guessnet オプションは ifupdown システムにより実行されるスクリプトに追加の環境変数をエクスポートするだけなので、”/etc/network/interfaces” ファイルを mapping スクリプトと guessnet-ifupdown とオリジナルのネットワーク設定インフラである ifupdown とで重複して利用することで特に支障は起こりません。詳細は guessnet-ifupdown(8) を参照下さい。

注意
”/etc/network/interfaces” 中に複数の **guessnet** オプション行が必要な時には、オプション行の開始文字列重複を ifupdown パッケージは許さないので、**guessnet1** や **guessnet2** 等で始まるオプション行を使います。

5.8 低レベルネットワーク設定

5.8.1 Iproute2 コマンド

Iproute2 コマンドは完全な低レベルネットワーク設定機能を提供します。型遅れとなった net-tools コマンドと新しい iproute2 コマンド等との翻訳表を次に示します。

型遅れの **net-tools**	新しい **iproute2** 等	操作
ifconfig(8)	ip addr	デバイスのプロトコル (IP または IPv6) アドレス
route(8)	ip route	ルーティングテーブル
arp(8)	ip neigh	ARP または NDISC キャッシュ項目
ipmaddr	ip maddr	マルチキャストアドレス
iptunnel	ip tunnel	IP 経由トンネル
nameif(8)	ifrename(8)	MAC アドレスに基づきネットワークインターフェースを命名
mii-tool(8)	ethtool(8)	イーサーネットデバイスの設定

Table 5.15: 型遅れとなった net-tools コマンドと新しい iproute2 コマンド等との翻訳表

ip(8) と IPROUTE2 Utility Suite Howto を参照下さい。

5.8.2 安全な低レベルネットワーク操作

次の低レベルネットワークコマンドは、ネットワーク設定を変更しないので安全に使えます。

コマンド	説明
`ifconfig`	有効インターフェースのリンクとアドレスの状態を表示
`ip addr show`	有効インターフェースのリンクとアドレスの状態を表示
`route -n`	数字を使ったアドレスで全てのルーティングテーブルを表示
`ip route show`	数字を使ったアドレスで全てのルーティングテーブルを表示
`arp`	ARP キャッシュテーブルの現状の内容を表示
`ip neigh`	ARP キャッシュテーブルの現状の内容を表示
`plog`	ppp デーモンのログを表示
`ping yahoo.com`	"`yahoo.com`" までのインターネット接続の確認
`whois yahoo.com`	ドメインデーターベースに"`yahoo.com`" を誰が登録したかを確認
`traceroute yahoo.com`	"`yahoo.com`" までのインターネット接続の追跡
`tracepath yahoo.com`	"`yahoo.com`" までのインターネット接続の追跡
`mtr yahoo.com`	"`yahoo.com`" までのインターネット接続の追跡 (繰り返し)
`dig [@dns-server.com] example.com [{a|mx|any}]`	"`example.com`" のDNS レコードを"`dns-server.com`" で"`a`" か"`mx`" か"`any`" かのレコードに関して確認します。
`iptables -L -n`	パケットフィルターの確認
`netstat -a`	オープンポートの発見
`netstat -l --inet`	聴取中のポートの発見
`netstat -ln --tcp`	聴取中の TCP ポートの発見 (数字)
`dlint example.com`	"`examle.com`" の DNS ゾーン情報を確認

Table 5.16: 低レベルネットワークコマンドのリスト

ティップ
これらの低レベルネットワーク設定ツールは"`/sbin/`" 中にあります。"`/sbin/ifconfig`" 等のような完全コマンドパスを使うか、"`~/.bashrc`" 中の"`$PATH`" リストに"`/sbin`" を追加する必要があるかもしれません。

5.9 ネットワークの最適化

一般的なネットワークの最適化は本書の射程外です。ここでは消費者用の接続に関する課題にのみ触れます。

パッケージ	ポプコン	サイズ	説明
`iftop`	V:7, I:115	97	ネットワークインターフェースの帯域利用情報を表示
`iperf`	V:4, I:55	263	インターネットプロトコルのバンド幅測定ツール
`ifstat`	V:0, I:8	60	インターフェース統計モニター
`bmon`	V:1, I:17	146	可搬型バンド幅モニター兼速度推定機
`ethstatus`	V:0, I:5	40	ネットワークデバイスのスループットを迅速に測定するスクリプト
`bing`	V:0, I:1	80	経験則的確率バンド幅試験ソフト
`bwm-ng`	V:2, I:17	90	簡単軽量のコンソール式のバンド幅モニター
`ethstats`	V:0, I:0	23	コンソール式のイーサーネット統計モニター
`ipfm`	V:0, I:0	78	帯域分析ツール

Table 5.17: ネットワーク最適化ツールのリスト

5.9.1 最適 MTU の発見

最大送信単位 (MTU) 値は、ping(8) を”`-M do`” オプションとともに使って ICMP パケットをデーターサイズ 1500 (IP+ICMP ヘッダー分の 28 バイトを加えて) から始めて IP フラグメンテーションしない最大サイズを見つけることで実験的に決定できます。

例えば、次を試してみて下さい:

```
$ ping -c 1 -s $((1500-28)) -M do www.debian.org
PING www.debian.org (194.109.137.218) 1472(1500) bytes of data.
From 192.168.11.2 icmp_seq=1 Frag needed and DF set (mtu = 1454)

--- www.debian.org ping statistics ---
0 packets transmitted, 0 received, +1 errors
```

1500 ではなく 1452 を試す

ping(8) が 1454 で成功するのを確認します。

このプロセスはパス MTU (PMTU) 発見 (RFC1191) で、tracepath(8) コマンドで自動化できます。

ティップ
PMTU 値が 1454 となる上記例は Asynchronous Transfer Mode (ATM) をバックボーンネットワークとして使い顧客を PPPoE でサービスしていた FTTP プロバイダーの場合でした。実際の PMTU 値はあなたの環境に依存します。例えば私の新しい FTTP プロバイダーの場合は 1500 です。

ネットワーク環境	**MTU**	理由
ダイヤルアップ接続 (IP: PPP)	576	標準
イーサーネット接続 (IP: DHCP または固定)	1500	標準かつデフォールト
イーサーネット接続 (IP: PPPoE)	1492 (=1500-8)	PPP ヘッダーに 2 バイト、PPPoE ヘッダーに 6 バイト、
イーサーネット接続 (ISP のバックボーン: ATM、IP: DHCP または固定)	1462 (=48*31-18-8)	著者推定: イーサーネットヘッダーに 18、SAR 末尾に 8
イーサーネット接続 (ISP のバックボーン: ATM、IP: PPPoE)	1454 (=48*31-8-18-8)	根拠は”Optimal MTU configuration for PPPoE ADSL Connections” 参照下さい

Table 5.18: 最適 MTU 値の基本的なガイドライン

これらの基本的なガイドラインに加えて、次を覚えておきます。

- 何らかのトンネル手法 (VPN等) を使うと、それらのオーバーヘッドのために最適 MTU を更に減らすかもしれません。
- MTU 値は実験的に決定される PMTU 値を越すべきではありません。
- もし他の制約条件を満たすなら、MTU 値は一般的に大きい方がいいです。

5.9.2 MTU の設定

MTU 値をそのデフォールトの 1500 から 1454 に設定する例を次に示します。

DHCP (項5.6.4参照下さい) の場合、”`/etc/network/interfaces`” 中の該当する **`iface`** スタンザ行を次と交換する事ができます。

```
iface eth0 inet dhcp
 pre-up /sbin/ifconfig $IFACE mtu 1454
```

静的 IP (項5.6.5参照下さい) の場合、”/etc/network/interfaces” 中の該当する **iface** スタンザ行を次と交換する事ができます。

```
iface eth0 inet static
 address 192.168.11.100
 netmask 255.255.255.0
 gateway 192.168.11.1
 mtu 1454
 dns-domain example.com
 dns-nameservers 192.168.11.1
```

直接の PPPoE (項5.5.5参照下さい) の場合、”/etc/ppp/peers/dsl-provider” 中の該当する”mtu” 行を次と交換する事ができます。

```
mtu 1454
```

最大セグメントサイズ (MSS) はパケットサイズの代替尺度として使われます。MSS と MTU の関係は次です。

- IPv4 では MSS = MTU - 40
- IPv6 では MSS = MTU - 60

注意
iptables(8) (項5.10参照下さい) を使う最適化は MSS を使ってパケットサイズを制約できるのでルーターとして有用です。iptables(8) 中の”TCPMSS” を参照下さい。

5.9.3 WAN TCP の最適化

現代的な高帯域でレイテンシーの大きな WAN では、TCP のスループットは TCP バッファーサイズパラメーターを”TCP Tuning Guide” や”TCP tuning” に書かれている手順で調整することで最大化できます。今のところ現在の Debian のデフォールトは高速の 1G bps の FTTP サービスでつながっている私の LAN でも十分機能しています。

5.10 Netfilter インフラ

Netfilter はLinux カーネルのモジュール (項3.3.1参照下さい) を利用するステートフルファイアーウォールとネットワークアドレス変換 (NAT) のインフラを提供します。

netfilter のユーザー空間の主プログラムは iptables(8) です。シェルから対話形式で手動で netfilter を設定し、その状態を iptables-save(8) で保存し、iptables-restore(8) を使って init スクリプト経由でシステムのリブート時に回復できます。

shorewall のような設定ヘルパースクリプトはこの過程を簡単にします。

http://www.netfilter.org/documentation/ (または”/usr/share/doc/iptables/html/” 中) の文書を参照下さい。

- Linux Networking-concepts HOWTO
- Linux 2.4 Packet Filtering HOWTO
- Linux 2.4 NAT HOWTO

ティップ
これらは Linux **2.4** のために書かれたとはいえ、iptables(8) コマンドも netfilter カーネル機能も現在の Linux **2.6** や **3.x** カーネルシリーズにもあてはまります。

パッケージ	ポプコン	サイズ	説明
iptables	V:300, I:993	2520	netfilter の管理ツール (IPv4 用の iptables(8)、IPv6 用の ip6tables(8))
arptables	V:0, I:2	96	netfilter の管理ツール (ARP 用の arptables(8))
ebtables	V:15, I:39	265	netfilter の管理ツール (Ethernet ブリッジング用の ebtables(8))
iptstate	V:0, I:3	116	netfilter の状態を常時モニター (top(1) と類似)
shorewall-init	V:0, I:0	68	Shoreline ファイアーウォール初期化
shorewall	V:5, I:13	2458	Shoreline ファイアーウォール、netfilter 設定ファイル生成システム
shorewall-lite	V:0, I:0	65	Shoreline ファイアーウォール、netfilter 設定ファイル生成システム (軽装備バージョン)
shorewall6	V:1, I:2	779	Shoreline ファイアーウォール、netfilter 設定ファイル生成システム (IPv6 バージョン)
shorewall6-lite	V:0, I:0	64	Shoreline ファイアーウォール、netfilter 設定ファイル生成システム (IPv6 軽装備バージョン)

Table 5.19: ファイアーウォールツールのリスト

Chapter 6

ネットワークアプリケーション

ネットワーク接続を確立した (第5章参照下さい) あとで、各種のネットワークアプリケーションを実行できます。

ティップ
最近の Debian に特化したネットワークインターフェースのガイドは、The Debian Administrator's Handbook — Network Infrastructure を参照ください。

ティップ
もしどこかの ISP で"2 段階認証" を有効にした場合、あなたのプログラムから POP や SMTP サービスにアクセスするアプリケーションパスワードを入手する必要があります。事前にあなたのホスト IP を許可する必要があるかもしれません。

6.1 ウェッブブラウザー

多くのウェッブブラウザーパッケージがハイパーテキストトランスファープロトコル (HTTP) を使って遠隔コンテントにアクセスするために存在します。

6.1.1 ブラウザー設定

次に示す特別の URL 文字列を使うと一部のブラウザーでその設定値を確認する事ができます。

- "`about:`"
- "`about:config`"
- "`about:plugins`"

Debian は、Java (ソフトウエアープラットフォーム) や Flash のみならず、MPEG や MPEG2 や MPEG4 や DivX や Windows Media Video (.wmv) や QuickTime (.mov) や MP3 (.mp3) や Ogg/Vorbis ファイルや DVDs や VCDs 等を取り扱えるブラウザーのプラグインコンポーネントを提供します。Debian では contrib や non-free アーカイブエリアに non-free のブラウザープラグインパッケージを提供しています。

ティップ
上記の Debian パッケージを使うのが遥に簡単であるとはいえ、今でもブラウザーのプラグインは"*.so" をプラグインディレクトリー (例えば"`/usr/lib/iceweasel/plugins/`") 等にインストールしブラウザーを再起動することで手動で有効にすることができます。

パッケージ	ポプコン	サイズ	タイプ	ウェッブブラウザーの説明
chromium	V:51, I:141	180040	X	Chromium、(Google からのオープンオースブラウザー)
firefox	V:13, I:20	205631	, ,	Firefox, (Mozilla からのオープンソースのブラウザー、Debian Unstable でのみ入手可能)
firefox-esr	V:217, I:437	198436	, ,	Firefox ESR, (Firefox 延長サポートリリース)
epiphany-browser	V:4, I:24	3730	, ,	GNOME、HIG 準拠、Epiphany
konqueror	V:18, I:100	20763	, ,	KDE、Konqueror
dillo	V:1, I:7	1536	, ,	Dillo, (軽量ブラウザー, FLTK 準拠)
w3m	V:31, I:284	2289	テキスト	w3m
lynx	V:13, I:98	1948	, ,	Lynx
elinks	V:6, I:28	1767	, ,	ELinks
links	V:6, I:39	2249	, ,	Links (テキストのみ)
links2	V:1, I:15	5417	グラフィクス	Links (X を使わないコンソールグラフィクス)

Table 6.1: ウェッブブラウザーのリスト

パッケージ	ポプコン	サイズ	エリア	説明
pepperflashplugin-nonfree	V:1, I:21	29	contrib	Pepper Flash Player - browser plugin
browser-plugin-freshplayer-pepperflash	I:9	1135	contrib	PPAPI-host NPAPI-plugin adapter for pepperflash

Table 6.2: ブラウザープラグインのリスト

ウェッブサイトによっては使っているブラウザーのユーザーエージェント文字列によって接続を拒否します。こういう状況はユーザーエージェント文字列を偽装することで回避できます。例えば、これは次の内容を”~/.gnome2/epiphany/mozilla/epiphany/user.js”か”~/.mozilla/firefox/*.default/user.js”といったユーザー設定ファイル追加すればできます。

```
user_pref(”general.useragent.override”,”Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0) ↩
    ”);
```

こうする代わりに、URL に”about:config” を入力して表示画面内容を右クリックしてこの変数を追加や再設定することでもできます。

注意
偽装されたユーザーエージェント文字列は Java に対して良からぬ副次効果を引き起こすかもしれません。

6.2 メールシステム

注意
もしインターネットと直接メール交換するメールサーバーを設定するなら、このような初歩的文書が不要なぐらいシステムを熟知しているべきです。

メールシステムには複数のホスト上の多くのサーバープログラムやクライアントプログラムが含まれます。機能的にはメールエージェントには 3 タイプあります:

- メール転送エージェント (MTA、項6.3参照) は異るホスト間でメールを転送するプログラムです。
- メール配送エージェント (MDA、項6.6参照) はホスト内のユーザーのメールボックスにメッセージを配送するプログラム。
- メールユーザーエージェント (MUA, 電子メールクライアントとも呼ばれます、項6.4参照) はメッセージ作成と配送されたメッセージにアクセスをするプログラムです。

注意
以下の設定例は消費者用インターネット接続上の典型的モービルワークステーションにのみ有効です。

6.2.1 E メールの基本

email メッセージは、メッセージのエンベロープ (封筒) と、メッセージのヘッダーと、メッセージの本体との、3 構成要素から成り立っています。

メッセージエンベロープ中の”To” (宛先) と”From” (差出人) 情報は SMTP が電子メールを配達するのに用いられます。(メッセージエンベロープの”From” 情報は バウンスアドレス、From_、等とも呼ばれます。)

メッセージヘッダー中の”To” (宛先) と”From” (差出人) 情報は email クライアント が email を表示するのに用いられます。(通常これらはメッセージエンベロープの情報と共通ですが、必ずしもそうとは限りません。)

電子メイルクライアント は、多目的インターネットメール拡張 (MIME) を持ちいてコンテントのデータータイプやエンコーディングを扱いメッセージヘッダーやボディーのデーターを解釈する必要があります。

6.2.2 近代的メールサービスの基礎

スパム (望みも頼みもしない電子メール) 問題を封じ込めるために、多くの消費者用インターネット接続を提供する ISP は対抗措置を実施しています。

- 彼らの顧客がメッセージを送信するためのスマートホストサービスは、rfc4954 で規定される (SMTP AUTH サービスの) パスワードを使い rfc4409 で規定されるメッセージサブミッションポート (587) を使います。
- ISP のネットワーク内部 (但し ISP 自身の送信メールサーバーを除く) からの SMTP ポート (25) からインターネットへの接続はブロックされます。
- 外部ネットワークの怪しげなホストから ISP の受信メールサーバーへの SMTP ポート (25) 接続はブロックされます。(ダイヤルアップ接続等の消費者用インターネット接続に用いられる動的 IP アドレス範囲上のホストからの接続はほぼ確実にブロックされます。)
- ドメインキー・アイデンティファイド・メール (DKIM) や SPF 認証 や ドメインベースのメッセージ認証、報告および適合 (DMARC) のような アンチスパムテクニック が email のフィルタリング に広範に使用されています。
- ドメインキー・アイデンティファイド・メールサービスがあなたのメールをスマートホスト経由で送信する際に提供されているかもしれません。
- スマートホストはスマートホスト上のあなたのメールアドレスに送信元メールアドレスを書き換えるかもしれません。

メールシステムを設定したりメール配達問題を解決する際には、こうした新たな制約に配慮しなければいけません。

注意
リモートホストに確実にメールを直接送るために、消費者用インターネット接続上で SMTP サーバーを実行するのは現実的ではありません。

注意
無関係の複数の送信元メールアドレスのメールを、単一のスマートホストを使って確実にリモートホストに送ることを期待するのは現実的ではありません。

注意
メールは送付先に到達する途中のどこかのホストによって黙って拒否されるでしょう。リモートホストに確実にメールを送る唯一の手段は、あなたのメールを正真正銘のものであるように見せることです。

このような敵意のあるインターネットの状況と制約を考慮して、Yahoo.com や Gmail.com 等の独立インターネットメール ISP はトランスポートレイヤーセキュリティー (TLS) やその前駆者であるセキュアーソケットレイヤー (SSL) を使ってインターネット上のどこからでも接続できるセキュアーなメールサービスを提供しています。

- 非推奨となっている SSL 経由の SMTP (SMTPS プロトコル) によるポート 465 を用いたスマートホストサービス。
- STARTTLS を用い、ポート 587 で提供されるスマートホストサービス。
- 受信メールには POP3 を使って TLS/POP3 ポート (995) でアクセスできます。

話を簡単にするために、"`smtp.hostname.dom`" にスマートホストがあり、SMTP 認証 が必要で、STARTTLS を使いメッセージサブミッションポート (587) を使うと、以下の話では仮定します。

6.2.3 ワークステーションのメール設定戦略

最も簡単なメール設定は、MUA (項6.4参照下さい) 自身が ISP のスマートホストにメールを送信し ISP の POP3 サーバーからメールを受信する設定です。この設定タイプは機能が充実した icedove(1) や evolution(1) 等の GUI の MUA でよく使われます。メールを種類毎にフィルターする必要がある際には MUA のフィルター機能を使う必要があります。このような場合にはローカルの MTA (項6.3参照下さい) は (送信者と受信者が同一ホスト上にある時には) ローカル配送のみをする必要があります。

Debian システムがマルチユーザーシステムであることに留意ください。あなたが唯一のユーザーでも多くのプログラムが root として実行される多くのプログラムがあり、それらはあなたにメールを送るかもしれません。

これに代わるメール設定は、ローカルの MTA 経由で ISP のスマートホストにメールを送信し ISP の POP3 サーバーからメール取得プログラム (項6.5参照下さい) によってローカルのメイルボックスに受信する設定です。メールの種類毎にフィルターする必要がある場合には、フィルター付きの MDA (項6.6参照下さい) を使って別々のメイルボックスにフィルターします。このタイプの設定は、どんな MUA で設定することができるのですが、単純な mutt(1) や mew(1) 等のコンソールの MUA (項6.4参照下さい) でよく使われます。このような場合にはローカルの MTA (項6.3参照下さい) はスマートホストへの配送とローカル配送の両方をする必要があります。モービルワークステーションは有効な FQDN を持たないので、メール配送エラーを避けるように外部に出すメール中のローカルメール名を隠して偽装するようにローカル MTA を設定します (項6.3.3参照下さい)。

ティップ
Maildir を使いホームディレクトリー下のどこかに email のメッセージを保存するように MUA/MDA を設定したいかもしれません。

6.3 メールトランスポートエージェント (MTA)

普通のワークステーションでは、メールトランスポートエージェント (MTA) に `exim4-*` か `postfix` パッケージのどちらがよく選ばれます。この選択は全くあなた次第です。

パッケージ	ポプコン	サイズ	説明
`exim4-daemon-light`	V:342, I:367	1493	Exim4 メールトランスポートエージェント (MTA: Debian のデフォールト)
`exim4-base`	V:349, I:377	1704	Exim4 文書 (text) と共通ファイル
`exim4-doc-html`	I:1	3662	Exim4 文書 (html)
`exim4-doc-info`	I:1	624	Exim4 文書 (info)
`postfix`	V:145, I:160	4182	Postfix メールトランスポートエージェント (MTA: Debian の代替候補)
`postfix-doc`	I:9	4444	Postfix 文書 (html+text)
`sasl2-bin`	V:5, I:19	428	Cyrus SASL API の実装 (SMTP AUTH について postfix を補完)
`cyrus-sasl2-doc`	I:1	575	Cyrus SASL - 文書

Table 6.3: ワークステーションでの基本的メールトランスポートエージェント関連パッケージのリスト

ポプコンの投票数では `exim4-*` は `postfix` より何倍も人気があるようですが、`postfix` が Debian のデベロッパーに人気がないということではありません。Debian のサーバーシステムは `exim4` も `postfix` も使っています。目立つ Debian のデベロッパーからのメーリングリスト投稿 メールのヘッダーの分析は、これら両方の MTA がともに人気があることを示唆しています。

`exim4-*` パッケージは非常に小さなメモリー消費とその設定が非常にフレキシブルであることで知られています。`postfix` パッケージは非常にコンパクトで高速で単純でセキュアーであることで知られています。両ソフトウエアーは十分な文書とともに提供され、品質とライセンスでもともに良好です。

Debian アーカイブ中には異なった能力と狙いを持ったメールトランスポートエージェント (MTA) パッケージに関して多くの選択肢があります。

パッケージ	ポプコン	サイズ	能力と狙い
exim4-daemon-light	V:342, I:367	1493	高機能
postfix	V:145, I:160	4182	高機能 (セキュリティー)
exim4-daemon-heavy	V:7, I:8	1643	高機能 (柔軟性)
sendmail-bin	V:14, I:15	1854	高機能 (既に慣れている場合)
nullmailer	V:7, I:10	479	超軽量、ローカルメール無し
ssmtp	V:8, I:11	2	超軽量、ローカルメール無し
courier-mta	V:0, I:0	2416	超高機能
masqmail	V:0, I:0	337	軽量
esmtp	V:0, I:0	128	軽量
esmtp-run	V:0, I:0	32	軽量 (esmtp の sendmail 互換性拡張)
msmtp	V:5, I:10	547	軽量
msmtp-mta	V:3, I:4	86	軽量 (msmtp の sendmail 互換性拡張)

Table 6.4: Debian アーカイブ中のメールトランスポートエージェント (MTA) パッケージに関する選択肢リスト

6.3.1 exim4 設定

注意
exim4 を設定して複数の送信元メールアドレスに対応する複数のスマートホストを経由してインターネットメールを送ることは簡単ではありません。popcon や cron 等のシステムプログラムために単一送信元アドレスだけのために exim4 を設定し、mutt 等のようなユーザープログラムのために複数の送信元アドレスだけのために msmtp を設定しましょう。

スマートホスト経由のインターネットメールに関しては、exim4-* パッケージを次のように (再) 設定します。

```
$ sudo /etc/init.d/exim4 stop
$ sudo dpkg-reconfigure exim4-config
```

”General type of mail configuration” に関して、” スマートホストでメール送信; SMTP または fetchmail で受信する” を選択します。

”System mail name:” をそのデフォルトである FQDN (項5.1.1参照下さい) に設定します。

”IP-addresses to listen on for incoming SMTP connections:” をそのデフォルトである”127.0.0.1 ; ::1” と設定します。

”Other destinations for which mail is accepted:” の内容を消去します。

”Machines to relay mail for:” の内容を消去します。

” 送出スマートホストの IP アドレスまたはホスト名:” を”smtp.hostname.dom:587” と設定します。

”Hide local mail name in outgoing mail?” に関して”<No>” を選択します。(この代わりに、項6.3.3にある”/etc/email-address
を使用します。)

”DNS クエリの数を最小限に留めますか (ダイヤルオンデマンド)?” に次の内のひとつの返答をします。

- 起動時にインターネットに接続されている場合は、”No” とします。
- 起動時にインターネットに接続されていない場合は、”Yes” とします。

”Delivery method for local mail:” を”mbox format in /var/mail/” と設定します。

”Split configuration into small files?:” に関して”<Yes>” を選択します。

”/etc/exim4/passwd.client” を編集しスマートホストのためのパスワードエントリーを作成します。

```
$ sudo vim /etc/exim4/passwd.client
 ...
$ cat /etc/exim4/passwd.client
^smtp.*\.hostname\.dom:username@hostname.dom:password
```

次のようにして exim4 を起動します。

```
$ sudo /etc/init.d/exim4 start
```

”/etc/exim4/passwd.client” 中のホスト名はエリアスであってはいけません。真のホスト名は次の様にして確認できます。

```
$ host smtp.hostname.dom
smtp.hostname.dom is an alias for smtp99.hostname.dom.
smtp99.hostname.dom has address 123.234.123.89
```

エリアス問題を回避するために”/etc/exim4/passwd.client” の中に正規表現を用いています。もし ISP がエリアスで示されるホストを移動させても SMTP AUTH がきっと動きます。

次のようにすれば exim4 の設定を手動で更新できます。

- ”/etc/exim4/” 中の exim4 設定ファイルの更新。
  - MACRO を設定するために”/etc/exim4/exim4.conf.localmacros” を作成し、”/etc/exim4/exim4.conf.tem を編集します。(非分割設定)
  - ”/etc/exim4/exim4.conf.d” サブディレクトリー中で、新規ファイルを作成したり既存ファイルを編集したりします。(分割設定)
- ”invoke-rc.d exim4 reload” を実行します。

次に示す正式のガイドを読んで下さい: ”/usr/share/doc/exim4-base/README.Debian.gz” と update-exim4.conf(8)。

注意
”DNS クエリの数を最小限に留めますか (ダイヤルオンデマンド)?” という debconf の質問に”No” (デフォールト値) が選ばれシステムがブート時にインターネットに繋がっていない場合、exim4 の起動は長い時間がかかります。

警告
たとえあなたの ISP が許可していても、暗号化なしに平文パスワードを用いることはセキュリティー上の問題があります。

ティップ
ポート 587 上で STARTTLS を用い SMTP を用いることが推奨されていますが、未だに一部 ISP は非推奨の SMTPS (ポート 465 上の SSL) を用いています。4.77 以降の Exim4 はこの非推奨の SMTPS プロトコルをクライアントとしてもサーバーとしてもサポートしています。

ティップ
あなたのラップトップ PC 用に”/etc/aliases” を尊重する軽量 MTA を探しているなら、exim4(8) の設定を”/etc/default/exim4” 中に”QUEUERUNNER='queueonly'” や”QUEUERUNNER='nodaemon'” 等と設定する事を考慮するべきです。

6.3.2 SASL を使う postfix の設定

スマートホスト経由のインターネットメールに関しては postfix 文書と重要マニュアルページを読むことから始めるべきです。

コマンド	機能
postfix(1)	Postfix コントロールプログラム
postconf(1)	Postfix の設定ユーティリティー
postconf(5)	Postfix 設定パラメーター
postmap(1)	Postfix 検索テーブルのメンテナンス
postalias(1)	Postfix エリアスデーターベースのメンテナンス

Table 6.5: 重要 postfix マニュアルページのリスト

`postfix` と `sasl2-bin` パッケージを次のように (再) 設定します。

```
$ sudo /etc/init.d/postfix stop
$ sudo dpkg-reconfigure postfix
```

” スマートホストを使ってインターネット” を選択します。

”SMTP リレーホスト (なければ空):” を”`[smtp.hostname.dom]:587`” と設定します。

```
$ sudo postconf -e 'smtp_sender_dependent_authentication = yes'
$ sudo postconf -e 'smtp_sasl_auth_enable = yes'
$ sudo postconf -e 'smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd'
$ sudo postconf -e 'smtp_sasl_type = cyrus'
$ sudo vim /etc/postfix/sasl_passwd
```

スマートホストのパスワードエントリーを作成します。

```
$ cat /etc/postfix/sasl_passwd
[smtp.hostname.dom]:587     username:password
$ sudo postmap hush:/etc/postfix/sasl_passwd
```

次に記すように `postfix` を起動します。

```
$ sudo /etc/init.d/postfix start
```

`dpkg-reconfigure` ダイアログと”`/etc/postfix/sasl_passwd`” の中で”`[`” と”`]`” を使うことで MX レコードを確認せずに指定された hostname その物を直接使うように確実にします。”`/usr/share/doc/postfix/html/SASL_RE`
の中の”Enabling SASL authentication in the Postfix SMTP client” を参照下さい。

6.3.3 メールアドレス設定

メールのトランスポートとデリバリーとユーザーのエージェントが使うメールアドレス設定ファイルが少々存在します。

通常”`/etc/mailname`” ファイル中の **mailname** はホストの IP の一つとして解決できる完全修飾ドメイン名 (FQDN) です。解決できる IP アドレスのあるホスト名を持たない可動ワークステーションの場合には、この **mailname** を”`hostname -f`” に設定します。(これは `exim4-*` と `postfix` の両方に有効な安全な選択肢です。)

ティップ
”`/etc/mailname`” の内容は多くの MTA 以外のプログラムによってそのデフォールト挙動のために使われます。`mutt` の場合、`~/muttrc` ファイル中の”`hostname`” と”`from`” 変数を設定して **mailname** の値をオーバーライドします。bts(1) や dch(1) 等の `devscripts` パッケージ中のプログラムの場合、環境変数の”`$DEBFULLNAME`” や”`$DEBEMAIL`” をエキスポートしてその値をオーバーライドします。

ファイル	機能	アプリケーション
/etc/mailname	(送出) メールのデフォールトのホスト名	Debian 固有、mailname(5)
/etc/email-addresses	送出メールのホスト名の偽装	exim(8) 固有、exim4-config_files(5)
/etc/postfix/generic	送出メールのホスト名の偽装	postfix(1) 固有、postmap(1) コマンド実行後有効。
/etc/aliases	受入メールのためのアカウント名のエリアス	一般的、newaliases(1) コマンド実行後有効。

Table 6.6: メールアドレス関連のファイルのリスト

ティップ
通常 popularity-contest パッケージは root アカウントからメールを FQDN 付きで送信します。/usr/share/popularity-contest/default.conf に記載された様に /etc/popularity-contest.conf 中に MAILFROM を設定する必要があります。こうしないと、smarthost の SMTP サーバーによってあなたのメールは拒否されます。少々面倒ですが、root からの全メールの発信元を書き替えるより、この方法は安全ですし、他のデーモンや cron スクリプトに関してもこの方法を適用するべきです。

---

mailname を”hostname -f” と設定した時には、次によって MTA で発信元メールアドレスを偽装することが実現できます。

- exim4(8) の場合、exim4-config_files(5) に説明されているように”/etc/email-addresses”
- postfix(1) の場合、generic(5) に説明されているように”/etc/postfix/generic”

postfix の場合、次に記す追加ステップが必要です。

```
# postmap hash:/etc/postfix/generic
# postconf -e 'smtp_generic_maps = hash:/etc/postfix/generic'
# postfix reload
```

あなたのメール設定は次のようにするとテストできます。

- exim(8) の場合、-brw, -bf, -bF, -bV, …オプションを使用
- postmap(1) の場合、-q オプションを使用

ティップ
Exim は exiqgrep(8) や exipick(8) のようないくつかのユーティリティープログラムとともに供給されます。利用可能なコマンドは”dpkg -L exim4-base|grep man8/” を参照下さい。

6.3.4 基本的な MTA の操作

基本的な MTA 操作が存在します。その一部は sendmail(1) 互換性インターフェース経由で実行する事もできます。

ティップ
”/etc/ppp/ip-up.d/*” 中のスクリプトで全メールを排出するのは良い考えかも知れません。

6.4 メールユーザーエージェント (MUA)

Debian 関連のメーリングリストを購読する場合、参加者のデファクトスタンダードで期待通り挙動をする mutt や mew 等の MUA を使うのは良い考えかも知れません。

exim コマンド	**postfix** コマンド	説明
`sendmail`	`sendmail`	標準入力からメールを読み配送を手配 (`-bm`)
`mailq`	`mailq`	メールキューを状態とキュー ID とともにリスト (`-bp`)
`newaliases`	`newaliases`	エリアスデーターベースを初期化 (`-I`)
`exim4 -q`	`postqueue -f`	待機メールを排出 (`-q`)
`exim4 -qf`	`postsuper -r ALL deferred; postqueue -f`	全メールを排出
`exim4 -qff`	`postsuper -r ALL; postqueue -f`	凍結メールをも排出
`exim4 -Mg queue_id`	`postsuper -h queue_id`	キュー ID によってメッセージを凍結
`exim4 -Mrm queue_id`	`postsuper -d queue_id`	キュー ID によってメッセージを削除
N/A	`postsuper -d ALL`	全メッセージを削除

Table 6.7: 基本的 MTA 操作のリスト

パッケージ	ポプコン	サイズ	タイプ
`evolution`	V:31, I:229	475	X GUI プログラム (GNOME3、グループウエアスイート)
`thunderbird`	V:57, I:138	165180	X GUI プログラム (GNOME2、ブランドを外した Mozilla Thunderbird)
`kmail`	V:34, I:88	18011	X GUI プログラム (KDE)
`mutt`	V:37, I:313	7056	きっと `vim` とともに使われるキャラクターターミナルプログラム
`mew`	V:0, I:0	2325	`(x)emacs` の下でキャラクターターミナルプログラム

Table 6.8: メールユーザーエージェント (MUA) のリスト

6.4.1 基本 MUA —Mutt

vim と組み合わせて mutt をメールユーザーエージェント (MUA) として使うように”~/.muttrc” を使って次に示すようにカスタム化します。

```
#
# User configuration file to override /etc/Muttrc
#
# spoof source mail address
set use_from
set hostname=example.dom
set from=”Name Surname <username@example.dom>”
set signature=”~/.signature”

# vim: ”gq” to reformat quotes
set editor=”vim -c ’set tw=72 et ft=mail’”

# ”mutt” goes to Inbox, while ”mutt -y” lists mailboxes
set mbox_type=Maildir           # use qmail Maildir format for creating mbox
set mbox=~/Mail                 # keep all mail boxes in $HOME/Mail/
set spoolfile=+Inbox            # mail delivered to $HOME/Mail/Inbox
set record=+Outbox              # save fcc mail to $HOME/Mail/Outbox
set postponed=+Postponed        # keep postponed in $HOME/Mail/postponed
set move=no                     # do not move Inbox items to mbox
set quit=ask-yes                # do not quit by ”q” only
set delete=yes                  # always delete w/o asking while exiting
set fcc_clear                   # store fcc as non encrypted

# Mailboxes in Maildir (automatic update)
mailboxes ‘cd ~/Mail; /bin/ls -1|sed -e ’s/^/+/’ | tr ”\n” ” ”‘
unmailboxes Maillog *.ev-summary

## Default
#set index_format=”%4C %Z %{%b %d} %-15.15L (%4l) %s”
## Thread index with senders (collapse)
set index_format=”%4C %Z %{%b %d} %-15.15n %?M?(#%03M)&(%4l)? %s”

## Default
#set folder_format=”%2C %t %N %F %2l %-8.8u %-8.8g %8s %d %f”
## just folder names
set folder_format=”%2C %t %N %f”
```

HTML メールや MS ワードのインラインのアッタチメントを表示するように、”/etc/mailcap” か”~/.mailcap” に次に記す内容を追加します。

```
text/html; lynx -force_html %s; needsterminal;
application/msword; /usr/bin/antiword ’%s’; copiousoutput; description=”Microsoft Word Text ↩
    ”; nametemplate=%s.doc
```

ティップ
Mutt はIMAP クライアントやメールボックス様式変換機として使えます。メッセージを”t” や”T” 他でタグできます。こうしてタグされたメッセージは”;c” を使う異なるメールボックス間の移動や”;d” を使う消去で一括処理できます。

6.4.2 上級 MUA —Mutt + msmtp

Mutt は msmtp を使うと複数の送信元メールアドレスに対応するスマートホストを使って複数の送信元メールアドレスを使うように設定できます。

ティップ
Msmtp は `/usr/sbin/sendmail` コマンドを提供する他の sendmail のエミュレーターと共存してインストールできる sendmail のエミュレーターです。だからシステムのメールを `exim4` や `postfix` のままにできます。

例として 3 つの email アドレスをサポートすることを考えましょう。

- ”My Name1 <myaccount1@gmail.com>”
- ”My Name2 <myaccount2@gmail.com>”
- ”My Name3 <myaccount3@example.org>”

3 つの異なる送信元メールアドレスのための 3 つのスマートホストをサポートするカスタム化 `~/.muttrc` 例を以下に示します。

```
set use_from
set from=”My Name3 <myaccount3@example.org>”
set reverse_name
alternates myaccount1@gmail\.com|myaccount1@gmail\.com|myaccount3@example\.org

# ...

# MACRO
macro compose ”1” ”<edit-from>^UMy Name1 \<myaccount1@gmail.com\>\n”
macro compose ”2” ”<edit-from>^UMy Name2 \<myaccount2@gmail.com\>\n”
macro compose ”3” ”<edit-from>^UMy Name3 \<myaccount3@example.org\>\n”

send2-hook ’~f myaccount1@gmail.com’ ”set sendmail = ’/usr/bin/msmtp --read-envelope-from’”
send2-hook ’~f myaccount2@gmail.com’ ”set sendmail = ’/usr/bin/msmtp --read-envelope-from’”
send2-hook ’~f myaccount3@example.org’ ”set sendmail = ’/usr/bin/msmtp --read-envelope-from ↩
    ’”

# ...
```

`msmtp-gnome` をインストールして `~/.msmtprc` を以下のように設定しましょう。

```
defaults
logfile ~/.msmtp.log
domain myhostname.example.org
tls on
tls_starttls on
tls_certcheck on
tls_trust_file /etc/ssl/certs/ca-certificates.crt
auth on
port 587
auto_from

account myaccount1@gmail.com
host smtp.gmail.com
from  myaccount1@gmail.com
user  myaccount1@gmail.com

account myaccount2@gmail.com
host smtp.gmail.com
from  myaccount2@gmail.com
user  myaccount2@gmail.com

account myaccount3@example.org
host mail.example.org
from  myaccount3@example.org
```

```
user  myaccount3@example.org

account default : myaccount3@example.org
```

そして、Gnome キーリングにパスワードデーターを加えましょう。例えば:

```
$ secret-tool store --label=msmtp \
    host smtp.gmail.com \
    service smtp \
    user myaccount1@gmail.com
...
```

ティップ
もし Gnome キーリングを使いたくない場合は、`msmtp` パッケージを代わりにインストールして各アカウント毎に”`password secret123`” のようなエントリーを `~/.msmtprc` の中に追加しましょう。詳しくは memtp のドキュメンテーション を参照ください。

6.5 リモートメールの取得および転送ユーティリティー

リモートメールにアクセスしそれらをマニュアル処理ために MUA を実行する代わりに、全てのメールをローカルホストに配送するように、そのような処理を自動化したいかもしれません。リモートメール回収・転送ユーティリティーはそんなあなたのためにあります。

fetchmail(1) は GNU/Linux 上のリモートメールの取得のデファクト標準でしたが、著者は現在 getmail(1) が気に入っています。もしバンド幅を節約するためにメールをダウンロードする前に拒否したいなら、`mailfilter` か `mpop` が役に立つかもしれません。どのメールの取得ユーティリティーを使おうとも、取得したメールをパイプ経由で `maildrop` 等の MDA に配送するようにシステム設定をすることをお勧めします。

パッケージ	ポプコン	サイズ	説明
`fetchmail`	V:5, I:17	814	メール取得ユーティリティー (POP3、APOP、IMAP) (旧式)
`getmail`	V:1, I:6	30	メール取得ユーティリティー (POP3、IMAP4、SDPS) (単純、セキュアー、高信頼)
`mailfilter`	V:0, I:0	291	メール取得ユーティリティー (POP3)、regex フィルター機能付き
`mpop`	V:0, I:0	400	メール取得ユーティリティー (POP3) かつ MDA、フィルター機能付き

Table 6.9: リモートメールの取得および転送ユーティリティーのリスト

6.5.1 getmail の設定

getmail(1) の設定は getmail 文書に記載されています。次に示すのがユーザーとして複数の POP3 アカウントにアクセスする著者の設定です。

”`/usr/local/bin/getmails`” を次のように作成します。

```
#!/bin/sh
set -e
if [ -f $HOME/.getmail/running ]; then
  echo ”getmail is already running ... (if not, remove $HOME/.getmail/running)” >&2
  pgrep -l ”getmai[l]”
  exit 1
```

```
else
  echo ”getmail has not been running ... ” >&2
fi
if [ -f $HOME/.getmail/stop ]; then
  echo ”do not run getmail ... (if not, remove $HOME/.getmail/stop)” >&2
  exit
fi
if [ ”x$1” = ”x-l” ]; then
  exit
fi
rcfiles=”/usr/bin/getmail”
for file in $HOME/.getmail/config/* ; do
  rcfiles=”$rcfiles --rcfile $file”
done
date -u > $HOME/.getmail/running
eval ”$rcfiles $@”
rm $HOME/.getmail/running
```

次のように設定します。

```
$ sudo chmod 755 /usr/local/bin/getmails
$ mkdir -m 0700 $HOME/.getmail
$ mkdir -m 0700 $HOME/.getmail/config
$ mkdir -m 0700 $HOME/.getmail/log
```

各 POP3 アカウント毎に”$HOME/.getmail/config/pop3_name” 設定ファイルを次のように作成します。

```
[retriever]
type = SimplePOP3SSLRetriever
server = pop.example.com
username =  pop3_name@example.com
password = <your-password>

[destination]
type = MDA_external
path = /usr/bin/maildrop
unixfrom = True

[options]
verbose = 0
delete = True
delivered_to = False
message_log = ~/.getmail/log/pop3_name.log
```

次のように設定します。

```
$ chmod 0600 $HOME/.getmail/config/*
```

”/usr/local/bin/getmails” が 15 分毎に cron(8) により実行されるようにスケジュールするために、”sudo crontab -e -u <user_name>” と実行して次に記すユーザーの cron エントリーを追加します。

```
5,20,35,50 * * * * /usr/local/bin/getmails --quiet
```

ティップ
POP3 へのアクセス問題は getmail に起因しないかもしれません。一部の有名な無償の POP3 サービスは POP3 のプロトコルに違反しているかも知れませんし、それらのスパムフィルターが完璧でないかも知れません。例えば、RETR コマンドを受信すると DELE コマンドの受信を待たずにメッセージを消去するかもしれませんし、スパムメールボックスに隔離するかも知れません。被害を最小限にするにアクセスされたメッセージをアーカイブして消去しないようにサービスの設定をします。”Some mail was not downloaded” を参照下さい。

6.5.2 fetchmail の設定

fetchmail(1) を設定するには、"/etc/default/fetchmail" や"/etc/fetchmailrc" や"$HOME/.fetchmailrc" を設定します。"/usr/share/doc/fetchmail/examples/fetchmailrc.example" 中の例を参照下さい。

6.6 フィルター付きのメールデリバリーエージェント (MDA)

postfix や exim4 等のほとんどの MTA プログラムは、MDA (メールデリバリーエージェント) として機能します。フィルター機能のある専門の MDA があります。

procmail(1) は GNU/Linux 上のフィルター付きの MDA のデファクト標準でしたが、著者は現在 maildrop(1) が気に入っています。どのフィルターユーティリティーを使おうとも、フィルターされたメールを qmail スタイルの Maildir にデリバリーするようにシステムを設定します。

パッケージ	ポプコン	サイズ	説明
procmail	V:40, I:277	300	フィルター付き MDA (旧式)
mailagent	V:0, I:5	1356	Perl フィルター付き MDA
maildrop	V:0, I:2	1141	構造化フィルター言語付き MDA

Table 6.10: フィルター付きの MDA のリスト

6.6.1 maildrop の設定

maildrop(1) の設定は maildropfilter 文書に記載されています。次に"$HOME/.mailfilter" の設定例を示します。

```
# Local configuration
MAILROOT="$HOME/Mail"
# set this to /etc/mailname contents
MAILHOST="example.dom"
logfile $HOME/.maildroplog

# rules are made to override the earlier value by the later one.

# mailing list mails ?
if (     /^Precedence:.*list/:h || /^Precedence:.*bulk/:h )
{
    # rules for mailing list mails
    # default mailbox for mails from mailing list
    MAILBOX="Inbox-list"
    # default mailbox for mails from debian.org
    if ( /^(Sender|Resent-From|Resent-Sender): .*debian.org/:h )
    {
        MAILBOX="service.debian.org"
    }
    # default mailbox for mails from bugs.debian.org (BTS)
    if ( /^(Sender|Resent-From|Resent-sender): .*@bugs.debian.org/:h )
    {
        MAILBOX="bugs.debian.org"
    }
    # mailbox for each properly maintained mailing list with "List-Id: foo" or "List-Id:  ↩
        ...<foo.bar>"
    if ( /^List-Id: ([^<]*<)?([^<>]*)>?/:h )
    {
        MAILBOX="$MATCH2"
    }
```

```
}
else
{
    # rules for non-mailing list mails
    # default incoming box
    MAILBOX="Inbox-unusual"
    # local mails
    if ( /Envelope-to: .*@$MAILHOST/:h )
    {
        MAILBOX="Inbox-local"
    }
    # html mails (99% spams)
    if ( /DOCTYPE html/:b ||\
         /^Content-Type: text\/html/ )
    {
        MAILBOX="Inbox-html"
    }
    # blacklist rule for spams
    if ( /^X-Advertisement/:h ||\
         /^Subject:.*BUSINESS PROPOSAL/:h ||\
         /^Subject:.*URGENT.*ASISSTANCE/:h ||\
         /^Subject: *I NEED YOUR ASSISTANCE/:h )
    {
        MAILBOX="Inbox-trash"
    }
    # whitelist rule for normal mails
    if ( /^From: .*@debian.org/:h ||\
         /^(Sender|Resent-From|Resent-Sender): .*debian.org/:h ||\
         /^Subject: .*(debian|bug|PATCH)/:h )
    {
        MAILBOX="Inbox"
    }
    # whiltelist rule for BTS related mails
    if ( /^Subject: .*Bug#.*/:h ||\
         /^(To|Cc): .*@bugs.debian.org/:h )
    {
        MAILBOX="bugs.debian.org"
    }
    # whitelist rule for getmails cron mails
    if ( /^Subject: Cron .*getmails/:h )
    {
        MAILBOX="Inbox-getmails"
    }
}

# check existance of $MAILBOX
`test -d $MAILROOT/$MAILBOX`
if ( $RETURNCODE == 1 )
{
    # create maildir mailbox for $MAILBOX
    `maildirmake $MAILROOT/$MAILBOX`
}
# deliver to maildir $MAILBOX
to "$MAILROOT/$MAILBOX/"
exit
```

警告
procmail と違い、maildrop は欠落した maildir ディレクトリーを自動的に作りません。”$HOME/.mailfilter” の例中のように事前に maildirmake(1) を使ってディレクトリーを作らなければいけません。

6.6.2 procmail の設定

procmail(1) 用の”$HOME/.procmailrc” を使う類似設定を次に記します。

```
MAILDIR=$HOME/Maildir
DEFAULT=$MAILDIR/Inbox/
LOGFILE=$MAILDIR/Maillog
# clearly bad looking mails: drop them into X-trash and exit
:0
* 1^0 ^X-Advertisement
* 1^0 ^Subject:.*BUSINESS PROPOSAL
* 1^0 ^Subject:.*URGENT.*ASISSTANCE
* 1^0 ^Subject: *I NEED YOUR ASSISTANCE
X-trash/

# Delivering mailinglist messages
:0
* 1^0 ^Precedence:.*list
* 1^0 ^Precedence:.*bulk
* 1^0 ^List-
* 1^0 ^X-Distribution:.*bulk
{
:0
* 1^0 ^Return-path:.*debian-devel-admin@debian.or.jp
jp-debian-devel/

:0
* ^Resent-Sender.*debian-user-request@lists.debian.org
debian-user/

:0
* ^Resent-Sender.*debian-devel-request@lists.debian.org
debian-devel/

:0
* ^Resent-Sender.*debian-announce-request@lists.debian.org
debian-announce

:0
mailing-list/
}

:0
Inbox/
```

6.6.3 mbox の内容の再配達

もしあなたのホームディレクトリーが一杯になり procmail(1) がうまく機能しなかった場合には、”/var/mail/<username> からホームディレクトリー内の仕分けられたメールボックスの中に手動でメールを配達しなければいけません。ディスク空間を確保した後に、次を実行します。

```
# /etc/init.d/${MAILDAEMON} stop
# formail -s procmail </var/mail/<username>
# /etc/init.d/${MAILDAEMON} start
```

6.7 POP3/IMAP4 サーバー

LAN 上でプライベートのサーバーを実行する場合、LAN クライアントにメールを配達するために POP3 / IMAP4 サーバーを実行することを考えます。

パッケージ	ポプコン	サイズ	タイプ	説明
courier-pop	V:2, I:2	308	POP3	Courier メールサーバー - POP3 サーバー (maildir フォーマットのみ)
cyrus-pop3d	V:0, I:0	160	POP3	Cyrus メールシステム (POP3 サポート)
courier-imap	V:3, I:4	589	IMAP	Courier メールサーバー - IMAP サーバー (maildir フォーマットのみ)
cyrus-imapd	V:1, I:1	484	IMAP	Cyrus メールシステム (IMAP サポート)

Table 6.11: POP3/IMAP サーバーのリスト

6.8 プリントサーバーとユーティリティー

旧来の Unix 的システムでは BSD のラインプリンターデーモンが標準でした。Unix 的システム上のフリーソフトウエアーの標準プリント出力フォーマットは PostScript なので、Ghostscript とともに何らかのフィルターシステムを使って non-PostScript プリンターへの印刷が可能になっています。

最近、共通 UNIX 印刷システム (CUPS) が新しいデファクトスタンダードです。CUPS は、インターネット印刷プロトコル (IPP) を使います。IPP は現在 Windows XP や Mac OS X 等の他の OS でもサポートされ、新たなクロスプラットフォームの両方向通信能力のあるリモート印刷のデファクト標準となっています。

Debian システム上のアプリケーションの標準の印刷可能データーフォーマットは、ページ記述言語である PostScript (PS) です。PS フォーマットのデーターは Ghostscript という PostScript のインタープリターに供給され、プリンター固有の印刷可能なデーターを生成します。項11.4.1を参照下さい。

CUPS システムのファイルフォーマット依存の自動変換機能のおかげで、どんなデーターでも `lpr` コマンドに供給すると期待される印刷出力が生成されます。(CUPS では、`lpr` は `cups-bsd` パッケージをインストールすると有効となります。)

Debian システムには、プリントサーバーやユーティリティーで留意すべきパッケージがいくつかあります。

ティップ
CUPS システムはウェッブブラウザーを”http://localhost:631/” に向けることで設定できます。

6.9 リーモートアクセスサーバーとユーティリティー (SSH)

セキュアーシェル (SSH) はインターネット経由で接続するセキュアーな方法です。Debian では、OpenSSH と呼ばれるフリーバージョンの SSH が `openssh-client` と `openssh-server` パッケージとして利用可能です。

パッケージ	ポプコン	サイズ	ポート	説明
lpr	V:3, I:4	362	printer (515)	BSD lpr/lpd (ラインプリンターデーモン)
lprng	V:1, I:1	3064	, ,	, , (拡張)
cups	V:140, I:395	1141	IPP (631)	インターネット印刷 CUPS サーバー
cups-client	V:56, I:454	493	, ,	CUPS 用System V プリンターコマンド: lp(1) と lpstat(1) と lpoptions(1) と cancel(1) と lpmove(8) と lpinfo(8) と lpadmin(8) 等
cups-bsd	V:36, I:385	122	, ,	CUPS 用BSD プリンターコマンド: lpr(1) と lpq(1) と lprm(1) と lpc(8) 等
printer-driver-gutenprint	V:100, I:372	937	非該当	CUPS 用のプリンタードライバー

Table 6.12: プリントサーバーとユーティリティーのリスト

パッケージ	ポプコン	サイズ	ツール	説明
openssh-client	V:803, I:996	4298	ssh(1)	セキュアーシェルクライアント
openssh-server	V:690, I:834	1567	sshd(8)	セキュアーシェルサーバー
ssh-askpass-fullscreen	V:0, I:0	42	ssh-askpass-fullscreen(1)	ユーザーに ssh-add 用のパスフレーズを質問する (GNOME2)
ssh-askpass	V:3, I:34	106	ssh-askpass(1)	ユーザーに ssh-add 用のパスフレーズを質問する (プレーン X)

Table 6.13: リモートアクセスサーバーとユーティリティーのリスト

注意
あなたの SSH がインターネットからアクセスできる場合には、項4.7.3を参照下さい。

ティップ
リモートのシェルプロセスが回線接続の中断の際にも継続するようにするために screen(1) プログラムを使いましょう (項9.1参照下さい)。

6.9.1 SSH の基本

警告
OpenSSH サーバーを実行したい場合には、"`/etc/ssh/sshd_not_to_be_run`" が存在してはいけません。

SSH には 2 つの認証プロトコルがあります。

注意
非 Debian システムを使う際にはこれらの相違点に注意します。

SSH プロトコル	**SSH** 手法	説明
SSH-1	”RSAAuthentication”	RSA アイデンティティ鍵を用いるユーザー認証
, ,	”RhostsAuthentication”	”.rhosts” に基づくホスト認証 (インセキュアー、無効化済み)
, ,	”RhostsRSAAuthentication”	RSA ホストキーと組み合わせの、”.rhosts” に基づくホスト認証 (無効化済み)
, ,	”ChallengeResponseAuthentication”	RSA チャレンジ応答認証
, ,	”PasswordAuthentication”	パスワードを用いる認証
SSH-2	”PubkeyAuthentication”	公開鍵を用いるユーザー認証
, ,	”HostbasedAuthentication”	公開キークライアントホスト認証と組み合わせの、”~/.rhosts” か”/etc/hosts.equiv” に基づくホスト認証 (無効化済み)
, ,	”ChallengeResponseAuthentication”	チャレンジ応答認証
, ,	”PasswordAuthentication”	パスワードを用いる認証

Table 6.14: SSH の認証プロトコルと方法のリスト

詳細は、”/usr/share/doc/ssh/README.Debian.gz” と ssh(1) と sshd(8) と ssh-agent(1) と ssh-keygen(1) を参照下さい。

次に示すのがキーとなる設定ファイルです。

設定ファイル	設定ファイルの説明
/etc/ssh/ssh_config	SSH クライアントのデフォールト、ssh_config(5) 参照下さい
/etc/ssh/sshd_config	SSH サーバーのデフォールト、sshd_config(5) 参照下さい
~/.ssh/authorized_keys	当該 SSH サーバーの当該アカウント接続用にクライアントが使用するデフォールト公開 SSH キー
~/.ssh/identity	ユーザーの秘密 SSH-1 RSA キー
~/.ssh/id_rsa	ユーザーの秘密 SSH-2 RSA キー
~/.ssh/id_dsa	ユーザーの秘密 SSH-2 DSA キー

Table 6.15: SSH 設定ファイルのリスト

ティップ
公開と秘密の SSH キーをどう使うかに関しては、ssh-keygen(1) と ssh-add(1) と ssh-agent(1) を参照下さい。

ティップ
接続をテストして設定を確認します。何らかの問題がある際には、”ssh -v” を使います。

ティップ
ローカルの秘密 SSH キーを暗号化するパスフレーズは”ssh-keygen -p” として後から変更できます。

ティップ
ホストを制限したり特定コマンドを実行するように”~/.ssh/authorized_keys” 中に記載してオプションを追加できます。詳細は、sshd(8) を参照下さい。

コマンド	説明
`ssh username@hostname.domain.ext`	デフォールトモードで接続します。
`ssh -v username@hostname.domain.ext`	デバグメッセージを有効にしてデフォールトモードで接続します。
`ssh -1 username@hostname.domain.ext`	SSH version 1 での接続を強制します。
`ssh -1 -o RSAAuthentication=no -l username hostname.domain.ext`	SSH version 1 でパスワードを使うことを強制します。
`ssh -o PreferredAuthentications=password -l username hostname.domain.ext`	SSH version 2 でパスワードを使うことを強制します。

Table 6.16: SSH クライアント起動例のリスト

次に示す内容は、クライアントから ssh(1) 接続をスタートします。

もしローカルとリモートで同一ユーザー名を使う際には、”`username@`” とタイプするのを省略できます。たとえローカルとリモートで異なるユーザー名を使う際にでも、”`~/.ssh/config`” を用いるとユーザー名のタイプを省略できます。例えば Debian Salsa サービスでのユーザー名が”`foo-guest`” という場合には、”`~/.ssh/config`” が次を含むように設定します。

```
Host salsa.debian.org people.debian.org
    User foo-guest
```

ssh(1) はユーザーにとってより賢明でよりセキュアーな telnet(1) として機能します。`telnet` コマンドと異なり、`ssh` コマンドは `telnet` エスケープ文字 (初期デフォールト CTRL-]) に出会うことで中断される事がありません。

6.9.2 SMTP/POP3 トンネルをするためのポートフォワーディング

`ssh` を通して `localhost` のポート 4025 から `remote-server` のポート 25 へと、`localhost` のポート 4110 から `remote-server` のポート 110 へと接続するパイプを設定するには、ローカルホスト上で次のように実行します。

```
# ssh -q -L 4025:remote-server:25 4110:remote-server:110 username@remote-server
```

このようにするとインターネット経由で SMTP/POP3 サーバーへとセキュアーに接続できます。リモートホストの”`/etc/ssh/sshd_config`” 中の”`AllowTcpForwarding`” エントリーを”`yes`” と設定します。

6.9.3 リモートパスワード無しでの接続

”`RSAAuthentication`” (SSH-1 プロトコル) もしくは”`PubkeyAuthentication`” (SSH-2 プロトコル) を使うと、リモートシステムのパスワードを覚える必要が無くなります。

リモートシステム上の”`/etc/ssh/sshd_config`” 中に”`RSAAuthentication yes`” か”`PubkeyAuthentication yes`” という対応する設定をします。

次に示すように、ローカルで認証キーを生成しリモートシステム上に公開キーをインストールします。

- ”`RSAAuthentication`”: SSH-1 の RSA キー (置き換えられたので非推奨。)

```
$ ssh-keygen
$ cat .ssh/identity.pub | ssh user1@remote "cat - >>.ssh/authorized_keys"
```

- ”`PubkeyAuthentication`”: SSH-2 の RSA キー

```
$ ssh-keygen -t rsa
$ cat .ssh/id_rsa.pub | ssh user1@remote "cat - >>.ssh/authorized_keys"
```

- "PubkeyAuthentication": SSH-2 の DSA キー (遅いので非推奨。)

```
$ ssh-keygen -t dsa
$ cat .ssh/id_dsa.pub | ssh user1@remote "cat - >>.ssh/authorized_keys"
```

ティップ
SSH-2 の DSA キーを使うことは、キーが小さく遅いので非推奨です。特許が期限切れとなったので DSA を使って RSA 特許を回避する理由はありません。DSA はデジタル署名アルゴリズムで遅いです。また DSA-1571-1 も参照下さい。

注意
SSH-2 で"HostbasedAuthentication" が機能するには、サーバーホストの"/etc/ssh/sshd_config" と、クライアントホストの"/etc/ssh/ssh_config" か"~/.ssh/config" という両方のホスト設定で"HostbasedAuthentication" を"yes" と調節する必要があります。

6.9.4 外部 SSH クライアントへの対処法

他のプラットフォーム上で利用可能なフリーな SSH クライアントがいくつかあります。

環境	フリーの SSH プログラム
Windows	puTTY (http://www.chiark.greenend.org.uk/~sgtatham/putty/) (GPL)
Windows (cygwin)	cygwin 中の SSH (http://www.cygwin.com/) (GPL)
古典的 Macintosh	macSSH (http://www.macssh.com/) (GPL)
Mac OS X	OpenSSH; ターミナルアプリケーションの ssh を使用しましょう (GPL)

Table 6.17: 他のプラットフォーム上で使えるフリーな SSH クライアントのリスト

6.9.5 ssh-agent の設定

SSH の認証キーをパスフレーズで保護する方がより安全です。もしパスフレーズが設定されていない場合には"ssh-keygen -p" で設定できます。

上記のようにパスワードを使って接続したリモートホスト上の"~/.ssh/authorized_keys" 中にあなたの公開 SSH キー (例えば"~/.ssh/id_rsa.pub") を設定します。

```
$ ssh-agent bash
$ ssh-add ~/.ssh/id_rsa
Enter passphrase for /home/<username>/.ssh/id_rsa:
Identity added: /home/<username>/.ssh/id_rsa (/home/<username>/.ssh/id_rsa)
```

次に示すように、今後リモートパスワードは必要ありません。

```
$ scp foo <username>@remote.host:foo
```

ssh-agent のセッションを終了するのに ^D を押します。

X サーバーの場合、通常の Debian の起動スクリプトは親プロセスとして ssh-agent を実行します。だから ssh-add は 1 回だけ実行すれば十分です。詳細は ssh-agent(1) と ssh-add(1) を参照下さい。

6.9.6 SSH 上のリモートシステムをシャットダウンする方法

”shutdown -h now” (項1.1.8参照下さい) を実行しているプロセスを at(1) コマンド (項9.3.13参照下さい) を使って次のようにして SSH が終了することから守る必要があります。

```
# echo ”shutdown -h now” | at now
```

”shutdown -h now” を screen(1) (項9.1参照下さい) セッション中で実行しても同様のことができます。

6.9.7 SSH のトラブルシュート

問題に出会ったら、設定ファイルのパーミッションを確認し、ssh を”-v” オプションとともに実行します。

root でファイアーウォールと問題を起こした場合には、”-p” オプションを使いましょう; こうするとサーバーポートの 1 —1023 を使うのを回避します。

リモートサイトへの ssh 接続が急に動作し無くなった際は、システム管理者による変更、特に可能性が高いのはシステムメンテナンス中に”host_key” が変更された結果かもしれません。実際にこういう状況で誰も洒落たハックでリモートホストとしてなりすまそうとしていないことを確認した後に、”host_key” エントリーをローカルホストの”~/.ssh/known_hosts” から削除すると再び接続できるようになります。

6.10 他のネットワークアプリケーションサーバー

他のネットワークアプリケーションサーバーを次に示します。

パッケージ	ポプコン	サイズ	プロトコル	説明
telnetd	V:1, I:3	115	TELNET	TELNET サーバー
telnetd-ssl	V:0, I:0	170	, ,	, , (SSL サポート)
nfs-kernel-server	V:38, I:79	342	NFS	Unix 式ファイル共有
samba	V:102, I:159	16629	SMB	Windows のファイルとプリンター共有
netatalk	V:2, I:3	2077	ATP	Apple/Mac のファイルとプリンター共有 (AppleTalk)
proftpd-basic	V:24, I:32	488	FTP	汎用ファイルダウンロード
apache2	V:246, I:315	610	HTTP	汎用ウェッブサーバー
squid	V:13, I:15	8385	, ,	汎用ウェッブプロキシサーバー
squid3	V:4, I:10	240	, ,	, ,
bind9	V:52, I:65	1063	DNS	他のホストの IP アドレス
isc-dhcp-server	V:18, I:54	1471	DHCP	クライアント自身の IP アドレス

Table 6.18: 他のネットワークアプリケーションサーバー

コモンインターネットファイルシステムプロトコル (CIFS) はサーバーメッセージブロック (SMB) と同じプロトコルで Microsoft Windows で広く使われています。

ティップ
サーバーシステムの統合には、項4.5.2 を参照下さい。

ティップ
ホスト名の解決は通常 DNS サーバーによって提供されます。ホストの IP アドレスが DHCP によって動的にアサインされる場合には Debian wiki 上の DDNS ページ に書かれているようにして bind9 と isc-dhcp-server を使いホスト名解決のための ダイナミック DNS が設定できます。

ティップ
Debian アーカイブの全内容のローカルのミラーサーバーを使うより、squid 等のプロキシサーバーを使う方がはるかにバンド幅を節約上ではるかに効率的です。

6.11 他のネットワークアプリケーションクライアント

他のネットワークアプリケーションクライアントを次に示します。

パッケージ	ポプコン	サイズ	プロトコル	説明
netcat	I:41	16	TCP/IP	TCP/IP 用万能ツール (スイス陸軍ナイフ)
openssl	V:794, I:993	1465	SSL	セキュアソケットレイヤー (SSL) のバイナリーと関連する暗号化ツール
stunnel4	V:5, I:17	507	, ,	万能 SSL ラッパー
telnet	V:65, I:904	163	TELNET	TELNET クライアント
telnet-ssl	V:0, I:3	210	, ,	, , (SSL サポート)
nfs-common	V:181, I:343	768	NFS	Unix 式ファイル共有
smbclient	V:16, I:174	2016	SMB	MS Windows のファイルとプリンター共有
cifs-utils	V:32, I:123	299	, ,	リモートの MS Windows ファイルをマウントやアンマウントするコマンド
ftp	V:18, I:282	137	FTP	FTP クライアント
lftp	V:6, I:39	2255	, ,	, ,
ncftp	V:3, I:22	1339	, ,	フルスクリーンの FTP クライアント
wget	V:288, I:988	3477	HTTP と FTP	ウェッブダウンローダー
curl	V:151, I:548	426	, ,	, ,
axel	V:0, I:4	216	, ,	加速ダウンローダー
aria2	V:2, I:19	1854	, ,	BitTorrent と Metalink サポート付き、加速ダウンローダー
bind9-host	V:382, I:948	365	DNS	bind9 由来の host(1) コマンド、”Priority: standard”
dnsutils	V:64, I:517	256	, ,	bind 由来の dig(1) コマンド、”Priority: standard”
isc-dhcp-client	V:231, I:979	686	DHCP	IP アドレス獲得
ldap-utils	V:14, I:75	718	LDAP	LDAP サーバーからデーター獲得

Table 6.19: 他のネットワークアプリケーションクライアント

6.12 システムデーモンの診断

telnet プログラムを使うとシステムデーモンへの手動接続とその診断ができます。

プレーンな POP3 サービスをテストするには、次のようにします。

```
$ telnet mail.ispname.net pop3
```

一部の ISP が提供する TLS/SSL を有効にした POP3 サービスをテストするには、telnet-ssl か openssl パッケージによる、TLS/SSL を有効にした telnet クライアントが必要です。

```
$ telnet -z ssl pop.gmail.com 995
```

```
$ openssl s_client -connect pop.gmail.com:995
```

次の RFC は各システムデーモンに関する必要な知見を提供します。

RFC	説明
rfc1939 と rfc2449	POP3 サービス
rfc3501	IMAP4 サービス
rfc2821 (rfc821)	SMTP サービス
rfc2822 (rfc822)	メールファイルフォーマット
rfc2045	Multipurpose Internet Mail Extensions (MIME)
rfc819	DNS サービス
rfc2616	HTTP サービス
rfc2396	URI 定義

Table 6.20: よく使われる RFC のリスト

”`/etc/services`” の中にポートの使用され方が記載されています。

Chapter 7

X Window システム

警告
本章は、2013 年にリリースされた Debian 7.0 (Wheezy) に基づいているため、内容が陳腐化しつつあります。

Debian システム上の X Window システムは X.Org 由来のソースに基づいています。

7.1 重要パッケージ

インストールを簡略化するための (メタ) パッケージが少々あります。

X の基本に関しては、X(7) と the LDP XWindow-User-HOWTO を参照下さい。

7.2 デスクトップ環境の設定

デスクトップ環境は、通常 X ウィンドウマネージャーとファイルマネージャーと互換性あるユーティリティープログラムのスイートの組み合わせです。

GNOME や KDE や Xfce や LXDE 等の充実したデスクトップ環境を `aptitude` のタスクメニューを使って設定できます。

ティップ
タスクメニューは Debian の `unstable/testing` 環境下では最新のパッケージの変遷状態を反映していないかもしれません。そのような状況ではパッケージ間のコンフリクトを避けるために aptitude(8) のタスクメニューの下でいくつかの (メタ) パッケージを非選択にする必要があります。(メタ) パッケージを非選択にする場合には、それらに依存関係を提供しているパッケージが自動削除されないように特定のパッケージを手動選択しなければいけません。

上記と違う方法として、Fluxbox 等の X ウィンドウマネージャーだけを使って簡単な環境を設定する事もできます。

X ウィンドウマネージャーやデスクトップ環境のガイドは X のためのウィンドウマネージャーを参照下さい。

7.2.1 Debian メニュー

Debian メニューシステムは `menu` パッケージの update-menus(1) を使って、テキストと X の両指向のプログラムに関して一般化されたインターフェースを提供します。各パッケージは”`/usr/share/menu/`” ディレクトリーにメニューデーターをインストールします。”`/usr/share/menu/README`” を参照下さい。

(メタ) パッケージ	ポプコン	サイズ	説明
xorg	I:457	52	X ライブラリー、X サーバー、フォントセット、基本的な X クライアントとユーティリティーの集合 (メタパッケージ)
xserver-xorg	V:66, I:492	238	X サーバーのフルスイーツとその設定
xbase-clients	I:26	46	X クライアントの雑多な集合 (メタパッケージ)
x11-common	V:372, I:755	308	X Window システムのためのファイルシステムインフラ
xorg-docs	I:6	2036	X.Org ソフトウエアースイーツの雑多な文書
menu	V:54, I:197	1509	メニューに対応しているアプリケーションに関して Debian メニューを生成
menu-xdg	V:31, I:109	27	Debian メニュー構造を freedesktop.org の xdg メニュー構造に変換
xdg-utils	V:229, I:521	327	freedesktop.org によって提供される統合デスクトップ環境のためのユーティリティー
task-gnome-desktop	I:175	9	標準の GNOME デスクトップ環境 (メタパッケージ)
task-kde-desktop	I:66	6	コアの KDE デスクトップ環境 (メタパッケージ)
task-xfce-desktop	I:106	9	Xfce 軽量デスクトップ環境 (メタパッケージ)
task-lxde-desktop	I:35	9	LXDE 軽量デスクトップ環境 (メタパッケージ)
fluxbox	V:2, I:9	3860	Fluxbox: 自由自在に設定可能でリソース消費が少ない X ウィンドウマネージャー

Table 7.1: X Window のためのキーとなる (メタ) パッケージのリスト

7.2.2 Freedesktop.org メニュー

Freedesktop.org の xdg メニューシステム対応の各パッケージは”/usr/share/applications/” の下の”*.desktop” で提供されるそのメニューデーターをインストールします。Freedesktop.org スタンダード対応の現代的デスクトップ環境は xdg-utils パッケージを使ってこれらのデーターからそれぞれのメニューを生成します。”/usr/share/doc/xdg-utils/README” を参照下さい。

7.2.3 Freedesktop.org メニューからの Debian メニュー

GNOME や KDE のような Freedesktop.org メニュー 準拠のウィンドーマネージャー環境から伝統的な Debian メニューにアクセスするのには、menu-xdg パッケージをインストールしなければいけません。

7.3 サーバー/クライアント関係

X Window システムはサーバーとクライアントのプログラムの組み合わせとして起動されます。ローカルとリモートと言う言葉に対応するサーバーとクライアントと言う言葉の意味に注意を払う必要があります。

最近の X サーバーは MIT 共有メモリー拡張 機能があり、ローカルの X クライアントとローカルの共有メモリーを使って通信します。これはネットワーク透過性の Xlib プロセス間通信チャンネルをバイパスし、大きなイメージを扱う際の性能が得られるようにしています。

7.4 X サーバー

X サーバーの情報は xorg(1) を参照下さい。

タイプ	説明
X サーバー	ユーザーのディスプレーや入力デバイスが接続されたローカルホスト上で実行されるプログラム。
X クライアント	データーを処理し X サーバーへ話しかけるリモートホスト上で実行されるプログラム。
アプリケーションサーバー	データーを処理しクライアントへ話しかけるリモートホスト上で実行されるプログラム。
アプリケーションクライアント	ユーザーのディスプレーや入力デバイスが接続されたローカルホスト上で実行されるプログラム。

Table 7.2: サーバー/クライアントの用語法のリスト

7.4.1 X サーバーの (再) 設定

X サーバーの (再) 設定は以下のようにします:

```
# dpkg-reconfigure --priority=low x11-common
```

注意
最近の Linux カーネルは DRM や KMS や udev によりグラフィクスや入力デバイスを良好にサポートします。X サーバーはこれらを使うように書き換えられました。だから”`/etc/X11/xorg.conf`” は通常あなたのシステムにありません。これらのパラメーターはカーネルにより設定されます。Linux カーネルドキュメンテーションの”`fb/modedb.txt`” を参照ください。

あなたのモニターのスペックに関して注意深く確認します。大きな高解像度の CRT モニターの場合、チラつきを軽減するためにモニターの許容する限りできるだけ高いリフレッシュレート (85 Hz なら十二分、75 Hz で十分) 設定することが望ましい。LCD モニターの場合、その低速反応性のためにより低速の標準リフレッシュレート (60 Hz) 設定で通常問題はありません。

注意
あなたのモニターシステムのハードウエアーを破壊するかもしれないので、高過ぎるリフレッシュレートを使わないように注意して下さい。

7.4.2 X サーバーへの接続方法

”X サーバー” (ディスプレー側) が”X クライアント” (アプリケーション側) からの接続を許可するようにするにはいくつかの方法があります。

警告
暗号手法を使っている等といった非常に良い理由無しには、X 接続のためにセキュアーされていないネットワーク経由のリモート TCP/IP 接続を使ってはいけません。暗号化無しのリモート TCP/IP ソケット接続は盗聴の被害に会いやすく、Debian システムではデフォールトで無効化されています。”`ssh -X`” を使います。

警告
セキュアーされていないネットワーク経由で XDMCP 接続も使ってはいけません。XDMCP 接続は、UDP/IP 経由で暗号化せずデーターを送信するので盗聴攻撃を受けやすいです。

パッケージ	ポプコン	サイズ	ユーザー	暗号化	方法	適切な用途
xbase-clients	I:26	46	非確認	いいえ	xhost コマンド	非推奨
xbase-clients	I:26	46	確認済み	いいえ	xauth コマンド	パイプ経由のローカル接続用
openssh-client	V:803, I:996	4298	確認済み	はい	ssh -X コマンド	リモートネットワーク接続用
gdm3	V:165, I:229	5101	確認済み	いいえ (XDMCP)	GNOME ディスプレーマネージャー	パイプ経由のローカル接続用
sddm	V:53, I:95	1742	確認済み	いいえ (XDMCP)	KDE ディスプレーマネージャー	パイプ経由のローカル接続用
xdm	V:3, I:6	686	確認済み	いいえ (XDMCP)	X ディスプレーマネージャー	パイプ経由のローカル接続用
wdm	V:31, I:284	2289	確認済み	いいえ (XDMCP)	WindowMaker ディスプレーマネージャー	パイプ経由のローカル接続用
ldm	V:0, I:0	436	確認済み	はい	LTSP ディスプレーマネージャー	リモート SSH ネットワーク接続用 (シンクライアント)

Table 7.3: X サーバーへの接続方法のリスト

ティップ
LTSP は、Linux ターミナルサーバープロジェクトのことです。

7.5 X Window システムの起動

X Window システムは X サーバーとそれに接続する X クライアントの組み合わせの X セッションとしてよく起動されます。通常のデスクトップ環境ではそれらの両方ともワークステーション上で実行されます。

X session は次のようにして起動されます。

- コマンドラインからの `startx` コマンド
- "`/etc/rc?.d/`" ディレクトリー ("?" はランレベルに対応) 中の最後にある起動スクリプトから起動される X ディスプレーマネージャーデーモンプログラム `*dm` の 1 つ

ティップ
ディスプレーマネージャーデーモンの起動スクリプトは実際に実行される前に"`/etc/X11/default-display-manager`" ファイルの内容を確認します。こうすることで X ディスプレーマネージャーデーモンプログラムが 1 つだけが実行されることを確実にします。

ティップ
X ディスプレーマネージャーの初期環境変数に関しては、項8.4.5を参照下さい。

本質的にこれらすべてのプログラムは"`/etc/X11/Xsession`" スクリプトを実行します。そうすることで、"`/etc/X11/Xsession`" スクリプトは、"`/etc/X11/Xsession.d/`" ディレクトリー中のスクリプトを run-parts(8) 風に実行します。これは本質的に次の順番で見つかる最初のプログラムを exec builtin コマンドで実行することです。

1. もし定義されていた場合には、X ディスプレーマネージャーにより"`/etc/X11/Xsession`" の引数として指定されたスクリプト。
2. もし定義されていた場合には、"`~/.xsession`" か"`~/.Xsession`" スクリプト。
3. もし定義されていた場合には、"`/usr/bin/x-session-manager`" コマンド。
4. もし定義されていた場合には、"`/usr/bin/x-window-manager`" コマンド。
5. もし定義されていた場合には、"`/usr/bin/x-terminal-emulator`" コマンド。

このプロセスは"`/etc/X11/Xsession.options`" の内容に影響されます。これらの"`/usr/bin/x-*`" コマンドが指し示すプログラムが正確に何であるかは Debian の alternative システムにより決定され、"`update-alternatives --config x-session-manager`" 等によって変更されます。

詳細は Xsession(5) 参照ください。

7.5.1 gdm3 で X セッションをスタート

gdm3(1) はメニューから X セッションのセッションのタイプ (デスクトップ環境: 項7.2) とか、言語 (ロケール: 項8.4) を選択できるようにします。それは"`~/.dmrc`" の中に選択されたデフォールト値を次のように保存します。

```
[Desktop]
Session=default
Language=ja_JP.UTF-8
```

7.5.2 X セッションのカスタム化 (古典的方法)

”/etc/X11/Xsession.options” が、”#” 文字が前に付いていない”allow-user-xsession” と言う行を含んでいるシステム上では、誰でも”~/.xsession” か”~/.Xsession” を定義することでシステムコードを完全にオーバーライドして”/etc/X11/Xsession” の挙動をカスタム化できます。”~/.xsession” ファイル中の最後のコマンドはあなたの最も好む X window/ セッションマネージャーを起動するように”exec some-window/session-manager” という形式の使う必要があります。

もし当該機能が使われる場合、システムユーティリティーによるディスプレー (やログイン) マネージャー (DM) や、セッションマネージャーや、ウィンドーマネージャー (WM) の選択は無視されます。

7.5.3 X セッションのカスタム化 (新方法)

上記のように完全にシステムコードをオーバーライドすること無しに X セッションをカスタム化する新方法を次に示します。

- ディスプレーマネージャー gdm3 は特定のセッションを選択する事ができて、それを”/etc/X11/Xsession” の引数に設定できます。
  - ”/etc/profile” や”~/.profile” や”/etc/xprofile” や”~/.xprofile” ファイルが gdm3 起動プロセスの一部として実行されます。
- ”~/.xsessionrc” ファイルが起動プロセスの一部として実行されます。(デスクトップ非依存)
  - ”#allow-user-xsession” が”/etc/X11/Xsession.options” 中にあっても”~/.xsessionrc” ファイルの実行を妨げることはありません。
- ”~/.gnomerc” ファイルが起動プロセスの一部として実行されます。(GNOME デスクトップのみ)

システムユーティリティーによるディスプレー (やログイン) マネージャー (DM) や、セッションマネージャーや、ウィンドーマネージャー (WM) の選択は尊重されます。

これらの設定ファイル中に”exec …” や”exit” があってはいけません。

7.5.4 リモート X クライアントを SSH 経由で接続

”ssh -X” を使うことで、ローカルの X サーバーからリモートのアプリケーションサーバーへのセキュアーな接続が可能となります。

コマンドラインオプション”-X” を使わないでおくには､リモートホストの”/etc/ssh/sshd_config” 中の”X11Forward エントリーを”yes” と設定します。

ローカルホスト上の X サーバーの起動します。

ローカルホスト上で xterm を開きます。

ssh(1) を実行してリモートサイトとの接続を次のように確立します。

```
localname @ localhost $ ssh -q -X loginname@remotehost.domain
Password:
```

リモートホスト上の”gimp” 等の X アプリケーションコマンドを次のように実行します。

```
loginname @ remotehost $ gimp &
```

ここに書かれた手法はリモート X クライアントがあたかもローカルの UNIX ドメインソケット経由でローカル接続されているかのようにして、リモート X クライアントからの出力を表示できるようにします。

7.5.5 インターネット経由のセキュアーな X ターミナル

インターネット経由のセキュアーな X ターミナルは `ldm` 等の専用のパッケージを使えば簡単に実現でき、リモートで実行される X デスクトップ環境の全てを表示します。あなたのローカル機器は SSH 経由で接続されたリモートのアプリケーションサーバーのシンクライアントになります。

7.6 X Window でのフォント

2002 年に、Fontconfig 2.0 がフォントアクセスの設定とカスタム化のためのディストリビューション非依存のライブラリーとして作られました。Debian は `squeeze` 以降 Fontconfig 2.0 だけをそのフォント設定に使います。

X Window システムのフォントサポートは次のように要約できます。

- 旧来の X サーバー側フォントサポートシステム
  - 旧式バージョンの X クライアントアプリケーションとの下位互換性ためにオリジナルの中核 X11 フォントシステムが提供されています。
  - オリジナルの中核 X11 フォントは X サーバーにインストールされます。
- 現代的な X クライアント側フォントサポートシステム
  - 現代的な X システムはこの後にリストされる (項7.6.1と項7.6.2と項7.6.3) 全てのフォントをアンチエリアシングなどの先進的機能とともにサポートします。
  - Xft 2.0 はGNOME や KDE や LibreOffice 由来等の現代的な X アプリケーションを FreeType 2.0 ライブラリーと結びつけます。
  - FreeType 2.0 はフォントのラスター化ライブラリーを提供します。
  - Fontconfig はXft 2.0 のためのフォント規定を提供します。その設定は fonts.conf(5) を参照下さい。
  - 現代的な Xft 2.0 を使う X アプリケーションは現代的な X サーバーに X レンダリング拡張を使って話しかけます。
  - X レンダリング拡張はフォントアクセスとグリフイメージ生成を X サーバーから X クライアントに移動します。

パッケージ	ポプコン	サイズ	説明
`xfonts-utils`	V:66, I:542	415	X Window システムフォントユーティリティープログラム
`libxft2`	V:143, I:662	122	Xft、X アプリケーションと FreeType フォントラスター化ライブラリーをつなげるライブラリー
`libfreetype6`	V:426, I:994	896	FreeType 2.0 フォントラスター化ライブラリー
`fontconfig`	V:354, I:776	583	Fontconfig、汎用フォント設定ライブラリー—サポートバイナリー
`fontconfig-config`	V:367, I:871	442	Fontconfig、汎用フォント設定ライブラリー—設定データー

Table 7.4: X Window フォントシステムをサポートするパッケージのテーブル

フォント設定情報は次のようにして確認できます。

- 中核 X11 フォントパスに関しては”`xset q`”
- fontconfig のフォントデフォールトに関しては”`fc-match`”
- fontconfig で利用可能なフォントに関しては”`fc-list`”

ティップ
”The Penguin and Unicode” は現代的な X Window システムの良い概論です。http://unifont.org/ にある他の文書も Unicode フォントや Unicode 化されたソフトや国際化や Unicode のフリー (英語で自由と無償という意味)/ リブレ (仏語等で自由の意味、無償という意味は無い)/ オープンソース (FLOSS) オペレーティングシステム上での使い勝手の問題に関する良い情報源です。

7.6.1 基本的フォント

コンピューターフォントには大きくわけて 2 つのタイプがあります。

- ビットマップフォント (低解像度のラスター化で良好)
- アウトラインやストロークフォント (高解像度ラスター化で良好)

ビットマップフォントを拡大するとギザギザのイメージになってしまいますが、アウトラインやストロークフォント拡大するとスムーズなイメージになります。

Debian システム上のビットマップフォントは、”.pcf.gz” というファイル拡張子を持った圧縮された X11 pcf ビットマップフォントファイルとして提供されます。

Debian システム上のアウトラインフォントは次で提供されます。

- ”.pfb” (バイナリーフォントファイル) と”.afm” (フォントメトリクスファイル) というファイル拡張子を持った PostScript Type 1 フォントファイル。
- ”.ttf” というファイル拡張子を通常持った TrueType (もしくは OpenType) フォントファイル。

ティップ
OpenType はTrueType と PostScript Type 1 の両方を置き換えることを目指しています。

フォントパッケージ	ポプコン	サイズ	サンセリフフォント	セリフフォント	モノスペースフォント	フォントの起源
PostScript	N/A	N/A	Helvetica	Times	Courier	Adobe
gsfonts	I:599	4439	Nimbus Sans L	Nimbus Roman No9 L	Nimbus Mono L	URW (Adobe 互換サイズ)
gsfonts-x11	I:82	95	Nimbus Sans L	Nimbus Roman No9 L	Nimbus Mono L	PostScript Type 1 フォントでの X フォントサポート。
t1-cyrillic	I:19	4878	Free Helvetian	Free Times	Free Courier	拡張 URW (Adobe 互換サイズ)
lmodern	V:13, I:113	33270	LMSans*	LMRoman*	LMTypewriter*	Computer Modern (TeX 由来) に準拠したスケーラブルな PostScript と OpenType のフォント

Table 7.5: PostScript Type 1 フォントへの対応表

ティップ
DejaVu フォントは Bitstream Vera フォントに基づきそれを包含します。

フォントパッケージ	ポプコン	サイズ	サンセリフフォント	セリフフォント	モノスペースフォント	フォントの起源
ttf-mscorefonts-installer	V:1, I:64	92	Arial	Times New Roman	Courier New	Microsoft (Adobe 互換サイズ) (これは non-free データーをインストールします)
fonts-liberation	I:469	2093	Liberation Sans	Liberation Serif	Liberation Mono	Liberation フォントプロジェクト (Microsoft 互換サイズ)
fonts-freefont-ttf	V:50, I:276	6656	FreeSans	FreeSerif	FreeMono	GNU freefont (Microsoft 互換サイズ)
fonts-dejavu	I:478	39	DejaVu Sans	DejaVu Serif	DejaVu Sans Mono	DejaVu、Unicode 対応 Bitstream Vera
fonts-dejavu-core	V:220, I:809	2954	DejaVu Sans	DejaVu Serif	DejaVu Sans Mono	DejaVu、Unicode 対応 Bitstream Vera (sans, sans-bold, serif, serif-bold, mono, mono-bold)
fonts-dejavu-extra	I:516	7493	N/A	N/A	N/A	DejaVu、Unicode 対応 Bitstream Vera (oblique, italic, bold-oblique, bold-italic, condensed)
ttf-unifont	I:21	21	N/A	N/A	unifont	GNU Unifont、Unicode 5.1 基本多言語面 (BMP) 中の全印刷可能文字

Table 7.6: TrueType フォントへの対応表

7.6.2 追加のフォント

aptitude(8) を使うと追加のフォントを簡単に見つけられます。

- ”Tasks” → ”Localization” の下の短いパッケージ一覧
- debtag への正規表現: ”`~Gmade-of::data:font`” を使ってフォントデーターにフィルターされた平坦なパッケージ一覧
- パッケージ名への正規表現: ”`~nxfonts-`” を使って BDF (ビットマップ) フォントパッケージにフィルターされた平坦なパッケージ一覧
- パッケージ名への正規表現: ”`~nttf-|~nfonts-`” を使って TrueType (アウトライン) フォントパッケージにフィルターされた平坦なパッケージ一覧

フリーなフォントは限られていることがあるので、Debian ユーザーにとっていくつかの商用 TrueType フォントをインストールする選択肢があります。こういったことをユーザーが簡単しやすいようにいくつかの利便性のためのパッケージが作成されています。

- `mathematica-fonts`
- `fonts-mscorefonts-installer`

あなたのフリーなシステムを non-Free のフォントで汚染する事になるとはいえ、TrueType フォントの選択肢は非常に沢山あります。

フォントタイプ	日本語フォント名	中国語フォント名	韓国語フォント名
サンセリフ	gothic, ゴチック	hei, gothic	dodum, gulim, gothic
セリフ	mincho, 明朝	song, ming	batang

Table 7.7: CJK フォント名中でフォントタイプを示すために使われるキーワード表

7.6.3 CJK フォント

CJK (中日韓) 文字のフォントに焦点を当てキーポイントを記します。

”P” の付いた”VL PGothic” のようなフォント名は、固定幅フォントの”VL Gothic” フォントに対応するプロポーショナルフォントです。

例えば、Shift_JIS コードテーブルには 7070 文字があります。それらは次のように分類できます。

- JIS X 0201 1 バイト文字 (191 文字、別名: 半角文字)
- JIS X 0208 2 バイト文字 (6879 文字、別名: 全角文字)

2 バイト文字は CJK 固定幅フォントを使うコンソールターミナル上で倍の幅を占めます。このような状況に対応するために、ファイル拡張子”.hbf” を使う Hanzi ビットマップフォント (HBF) ファイルが 1 バイトと 2 バイトの文字を含むフォントのために使えます。

TrueType フォントファイルのための空間を節約するために、ファイル拡張子”.ttc” を持つ TrueType フォントコレクションファイルを使う事ができます。

文字の複雑なコード空間をカバーするために、CID でキーされた PostScript Type 1 フォントは”%!PS-Adobe-3.0 Resource-CMap” で始まる CMap ファイルとともに使われます。これは通常の X ディスプレーではほとんど使われませんが PDF のレンダリング等では使われます (項7.7.2参照下さい)。

ティップ
ハン (漢) 統一のために複数のグリフがいくつかの Unicode コードポイントに対して期待されています。最も気になることの一つは CJK 国間で文字の位置が異なる”U+3001 IDEOGRAPHIC COMMA” と”U+3002 IDEOGRAPHIC FULL STOP” です。”~/.fonts.conf” を使って日本語中心のフォントを中国語中心のフォントより優先順位を上げるよう設定することで日本人は安心できるようになります。

7.7 X アプリケーション

7.7.1 X オフィスアプリケーション

基本的なオフィスアプリケーションのリストを記します (LO は LibreOffice)。

7.7.2 X ユーティリティーアプリケーション

著者の目に止まった基本的ユーティリティーアプリケーションのリストを記します。

注意
evince や okular によって CJK の PDF 文書を Cmap データー (項7.6.3) を使って表示する際には poppler-data パッケージ (以前は non-free だった、項11.4.1参照下さい) が必要です。

パッケージ	ポプコン	パッケージサイズ	タイプ	説明
libreoffice-writer	V:188, I:441	39333	LO	ワードプロセッサ
libreoffice-calc	V:188, I:436	32973	LO	スプレッドシート
libreoffice-impress	V:176, I:433	9934	LO	プレゼンテーション
libreoffice-base	V:145, I:325	7473	LO	データーベース管理
libreoffice-draw	V:177, I:434	14600	LO	ベクトル画像エディター (ドロー)
libreoffice-math	V:174, I:437	1963	LO	数式エディター
abiword	V:1, I:12	5141	GNOME	ワードプロセッサ
gnumeric	V:6, I:21	9933	GNOME	スプレッドシート
gimp	V:68, I:341	22313	GTK	ビットマップ画像エディター (ペイント)
inkscape	V:55, I:209	84823	GNOME	ベクトル画像エディター (ドロー)
dia	V:5, I:31	3727	GTK	フローチャートやダイアグラムエディター
planner	V:0, I:5	1146	GNOME	プロジェクト管理
calligrawords	V:0, I:7	5717	KDE	ワードプロセッサ
calligrasheets	V:0, I:6	10890	KDE	スプレッドシート
calligrastage	V:0, I:5	5102	KDE	プレゼンテーション
calligraplan	V:0, I:2	15342	KDE	プロジェクト管理
kexi	V:0, I:2	7576	KDE	データーベース管理
karbon	V:0, I:7	3473	KDE	ベクトル画像エディター (ドロー)

Table 7.8: 基本的な X オフィスアプリケーションのリスト

パッケージ	ポプコン	パッケージサイズ	タイプ	説明
evince	V:116, I:329	954	GNOME	文書 (pdf) ビューワー
okular	V:46, I:118	14646	KDE	文書 (pdf) ビューワー
calibre	V:9, I:36	54876	KDE	e-book コンバーターとライブラリーの管理
fbreader	V:2, I:15	3074	GTK	e-book リーダー
evolution	V:31, I:229	475	GNOME	個人情報管理 (グループウエアと電子メール)
kontact	V:1, I:16	2152	KDE	個人情報管理 (グループウエアと電子メール)
scribus	V:2, I:23	30375	KDE	デスクトップページレイアウトエディター
glabels	V:0, I:4	1326	GNOME	ラベルエディター
gnucash	V:3, I:12	32304	GNOME	個人会計
homebank	V:0, I:3	1044	GTK	個人会計
kmymoney	V:0, I:2	12036	KDE	個人会計
shotwell	V:19, I:223	6451	GTK	デジタル写真オーガナイザー
xsane	V:17, I:173	2346	GTK	スキャナーのフロントエンド

Table 7.9: 基本的 X ユーティリティーアプリケーションのリスト

注意
scribus (KDE) のようなソフトウエアーを GNOME デスクトップ環境にインストールすることは、同様の機能が GNOME デスクトップ環境下で利用でき無いのでまったく問題ありません。ただ、機能が重複するパッケージをインストールしすぎるとあなたのメニューが忙しくなってしまいます。

7.8 X トリビア

7.8.1 クリップボード

3 つのマウスボタンを使っての X 選択は X の本来のクリップボード機能です (項1.4.4参照)。

ティップ
Shift-Insert は真ん中のマウスボタンのクリックと同等の動作をします。

パッケージ	ポプコン	パッケージサイズ	タイプ	説明
xsel	V:10, I:44	59	X	X 選択のコマンドラインインターフェース
xclip	V:9, I:49	64	X	X 選択のコマンドラインインターフェース

Table 7.10: 基本的 X 選択プログラムのリスト

最近のデスクトップ環境 (GNOME, KDE, ⋯) は、左のマウスボタンとキー (CTRL-X と CRTL-C と CTRL-V) を使った、カット、コピー、ペーストのための別個のクリップボードシステムを提供します。

7.8.2 X でのキーマップとポインターボタンのマッピング

xmodmap(1) は X Window システム中でのキーマップとポインターボタンのマッピングのためのユーティリティーです。**keycode** を知るには、X 環境下で xev(1) を実行してキーを押さえます。**keysym** の意味を知るには、"/usr/include/X11/keysymdef.h" ファイル (x11proto-core-dev パッケージ) 中の MACRO 定義を覗いて下さい。このファイル中の全ての"#define" 文は **keysym** 名に"XK_" を前付けして名づけられています。

7.8.3 古典的 X クライアント

xterm(1) のような多くの伝統的 X クライアントプログラムは、ジオメトリやフォントや表示を規定する標準化されたコマンドラインオプションの組み合わせを使って起動できます。

それらはその見栄えを設定するのに X リソースデーターベースも用います。X リソースのシステム全体のデフォールトは"/etc/X11/Xresources/*" の中に保存されており、それらのアプリケーションのデフォールトは"/etc/X11/app-defaults/*" の中に保存されています。これらの設定をスタート点として使います。

"~/.Xresources" ファイルはユーザーのリソース規定を保存するために使われます。ログイン時にこのファイルは自動的にデフォールトの X リソースに合流されます。この設定変更をしてすぐ有効にするには、それを次のコマンドを使ってデーターベースに合流させます。

```
$ xrdb -merge ~/.Xresources
```

x(7) と xrdb(1) を参照下さい。

7.8.4 X ターミナルエミュレーター—xterm

xterm(1) に関することは、http://dickey.his.com/xterm/xterm.faq.html で学びます。

7.8.5 X クライアントを root で実行

警告
gdm3 のようなディスプレーマネージャーのプロンプトに root と入力して X ディスプレー・セッションマネージャーを root アカウントの下で実行してはいけません。なぜなら、たとえシステム管理業務を行おうとしている時ですら、こういう行為は安全でない (インセキュアー) と認識されているからです。X アーキテクチャ全てが root として実行するとインセキュアーと認識されています。通常ユーザーのような、可能な限り最低レベルの特権を使うように常にするべきです。

例えば”foo” 等の特定の X クライアントを root として実行する最も簡単な方法は次に記すように sudo(8) を使うことです。

```
$ sudo foo &
```

```
$ sudo -s
# foo &
```

```
$ ssh -X root@localhost
# foo &
```

注意
この目的だけのために上記のように ssh(1) を使うことはリソースの無駄遣いです。

X クライアントが X サーバーに接続するためには次のことに注意下さい。

- 元のユーザーの”$XAUTHORITY” と”$DISPLAY” 環境変数の値は新たなユーザーの環境変数値にコピーされなければいけません。
- ”$XAUTHORITY” 環境変数の値で指示されるファイルが新たなユーザーによって読めなければいけません。

Chapter 8

I18N と L10N

アプリケーションソフトの多言語化 (M17N) とかネイティブ言語サポートは 2 段階で行います。

- 国際化 (I18N): ソフトが複数のロケール (地域) を扱えるようにします。
- 地域化 (L10N): 特定のロケール (地域) を扱えるようにします。

ティップ
M17N、I18N、L10N に対応する英語の multilingualization、internationalization、localization の中の”m” と”n”、”i” と”n”、”l” と”n” の間には 17、18、10 の文字があります。

GNOME や KDE 等の現代的なソフトは多言語化されています。UTF-8 データーを扱えるようにすることで国際化され、gettext(1) インフラで翻訳されたメッセージを提供することで地域化されています。翻訳されたメッセージは別の地域化パッケージとして供給されているかもしれません。該当する環境変数を適切なロケールに設定することだけで翻訳されたメッセージが選ばれます。

最も簡単なテキストデーターの表現法は **ASCII** です。これは英語では十分で (7 ビットで表現できる) 127 文字以下しか使いません。国際化サポートのためにより多くの文字をサポートするために多くの文字の符号化 (エンコーディング) システムが発明されています。現代的かつ賢明な符号化システムは、人類が知っている事実上全ての文字が扱える **UTF-8** です (項8.4.1参照下さい)。

詳細は Introduction to i18n を参照下さい。

国際化ハードウエアーサポートは地域化した設定データーを使って実現されています。

警告
本章は、2013 年にリリースされた Debian 7.0 (Wheezy) に基づいているため、内容が陳腐化しつつあります。

8.1 キーボード入力

Debian システムは `keyboard-configuration` と `console-setup` パッケージを使い多くの国際キーボード配列として機能するように設定できます。

```
# dpkg-reconfigure keyboard-configuration
# dpkg-reconfigure console-setup
```

これは”/etc/default/keyboard” と”/etc/default/console-setup” にある Linux コンソールと X Window のキーボード設定の設定パラメーターを更新します。これは Linux コンソールのフォントも設定します。

多くの欧州言語で用いられるアクセント付きの文字を含めた多くの非-ASCII 文字はデッドキー や AltGr キー や コンポーズキー を使うことでアクセスできます。

アジア言語に関しては次に記す IBus のような、より複雑な インプットメソッドが必要です。

8.1.1 IBus を使うインプットメソッドのサポート

アプリケーションへの多言語入力は次のように処理されます。

```
b'' キ b''b'' ー b''b'' ボ b''b'' ー b''b'' ド
    b''                                                  b'' ア b''b'' プ b''b'' リ
    b''b'' ケ b''b'' ー b''b'' シ b''b'' ョ b''b'' ン b''
   |                                                                          ^
   |                                                                           |
   +-> Linux b'' カ b''b'' ー b''b'' ネ b''b'' ル b'' -> b'' イ b''b'' ン b''b'' プ b''b'' ッ
       b''b'' ト b''b'' メ b''b'' ソ b''b'' ッ b''b'' ド b'' -> Gtk, Qt, or X
```

Debian システムための多言語入力の設定は、im-config パッケージとともに IBus ファミリーのパッケージを使うことで簡素化されました。IBus パッケージのリストは次です。

パッケージ	ポプコン	サイズ	サポートされたロケール
ibus	V:70, I:87	1581	dbus を用いるインプットメソッドのフレームワーク
ibus-mozc	V:1, I:2	999	日本語
ibus-anthy	V:0, I:1	8723	, ,
ibus-kkc	V:0, I:0	214	, ,
ibus-skk	V:0, I:0	244	, ,
ibus-pinyin	V:0, I:1	1434	中国語 (zh_CN 用)
ibus-chewing	V:0, I:0	415	, , (zh_TW 用)
ibus-hangul	V:0, I:1	288	韓国語
ibus-table	V:0, I:1	1801	IBus 用のテーブルエンジン
ibus-table-thai	I:0	47	タイ語
ibus-unikey	V:0, I:0	318	ベトナム語
ibus-m17n	V:0, I:1	187	多言語: インド系言語、アラビア語、他

Table 8.1: IBus を用いるインプットメソッドサポートのリスト

kinput2 法や他のロケール依存のアジアの古典的インプットメソッドはまだありますが、現代的な UTF-8 の X 環境下ではお勧めできません。SCIM や uim ツールチェインは現代的な UTF-8 の X 環境下での国際インプットメソッドの少し古いアプローチです。

8.1.2 日本語の例

日本語インプットメソッドを英語環境 (”en_US.UTF-8”) 下で起動すると非常に便利です。GNOME3 環境下で IBus を使ってどう実現したかを以下に記します。

1. 日本語インプットツールパッケージの ibus-anthy を im-config 等の推奨 (recommended) されたパッケージとともにインストールします。
2. ユーザーのシェルから”im-config” を実行して”ibus” をインプットメソッドとして選択します。
3. ”Settings” → ”Keyboard” → ”Input Sources” → click ”+” in ”Input Sources” → ”Japanese” → ”Japanese (anthy)” を選択し”Add” をクリックします。

4. ”Japanese” を選択し”Add” をクリックして文字変換なしの日本語配列キーボードをサポートします。(インプットソースは任意に複数選べます。)
5. ユーザーアカウントへの再ログイン
6. ”`im-config`” として設定を確認します。
7. GUI ツールバーアイコンを右クリックしてインプットソースを設定します。
8. インプットソース間を、SUPER-SPACE を用いて切り替えます。(SUPER は通常 Windows キーです。)

以下に注意下さい。

- im-config(8) は実行されるのが root からかどうかによって違った挙動をします。
- im-config(8) はユーザーからのアクション無しにシステム上で最も好ましいインプットメソッドを有効にします。
- im-config(8) のための GUI メニューエントリーは乱雑になることを防ぐためにデフォルトでは無効にされています。

8.1.3 インプットメソッドを無効化

XIM を経由せずに入力したい場合には、プログラムを起動する際に”`$XMODIFIERS`” の値を”none” と設定します。もし emacs(1) 上で日本語入力インフラ `egg` を使う場合にはこれが当てはまるかもしれません。シェルから、次を実行します。

```
$ XMODIFIERS=none emacs
```

Debian メニューによって実行されるコマンドを調整するには、”`/usr/share/doc/menu/html`” に記された方法にしたがって”`/etc/menu/`” 中にカスタム化した設定を置きます。

8.2 ディスプレー出力

Linux コンソールは限定された文字しか表示できません。(非 X コンソール上で非ヨーロッパ言語を表示するには jfbterm(1) のような特別なターミナルプログラムを使う必要があります。)

X Window は必要なフォントデーターがあれば UTF-8 中の全ての文字を表示できます。(オリジナルフォントデーターで使われた符号化方式は X Window システムが面倒を見るのでユーザーからは直接見えません。)

8.3 東アジア不明瞭文字幅文字

東アジアのロケールでは、箱描画文字やギリシャ文字やキリル文字はあなたが望より広い幅で表示されターミナル出力が揃わなくなるかもしれません (Unicode 標準附属書 #11 参照)。

この問題は回避可能です:

- `gnome-terminal`: Edit → Preferences → Profiles → Edit → Compatibility → Ambiguous-wide characters → Narrow
- `ncurses`: 環境変数を `export NCURSES_NO_UTF8_ACS=0` と設定します。

8.4 ロケール

以下では gdm3(1) から起動された X Window 環境下で実行されるアプリケーションのためのロケールに焦点を当てます。

8.4.1 符号化方式の基本

環境変数”LANG=xx_YY.ZZZZ” は、ロケールを言語コード”xx” と国コード”yy” と符号化方式 (エンコーディング) ”ZZZZ” に設定します (項1.5.2参照下さい)。

現在の Debian システムは通常ロケールを”LANG=xx_YY.UTF-8” と設定します。これは Unicode 文字セットとともに UTF-8 符号化方式を使用します。この UTF-8 符号化システムはマルチバイトコードシステムでコードポイントを上手に使います。ASCII データーは、7 ビットのコード域のみで構成されているので、1 文字 1 バイトのみからなる必ず有効な UTF-8 データーです。

以前の Debian システムはロケールを”LANG=C” か”LANG=xx_YY” (”.UTF-8” は無しで) 設定していました。

- ASCII 文字セットが”LANG=C” か”LANG=POSIX” の場合に使われます。
- 伝統的な Unix での符号化方式が”LANG=xx_YY” の場合に使われます。

”LANG=xx_YY” の場合に実際に使われる符号化方式は”/usr/share/i18n/SUPPORTED” を確認することで識別できます。例えば、”en_US” は”ISO-8859-1” 符号化方式を使い、”fr_FR@euro” は”ISO-8859-15” 符号化方式を使います。

ティップ
符号化方式の値の意味に関しては、表 11.2を参照下さい。

8.4.2 UTF-8 ロケールを使う根拠

ユニコード 文字セットは実質的に人類が知り得る全ての文字を 16 進表記で 0 から 10FFFF までのコードポイント範囲で表記できます。そのストレージには最小限 21 ビット必要です。

UTF-8 符号化方式は I18N のための現代的で気の利いた符号化方式で、Unicode 文字である人類が知る実質的に全ての文字を表せます。**UTF** とは Unicode 変換フォーマット (Unicode Transformation Format: UTF) 方式のことです。

私は例えば”LANG=en_US.UTF-8” という UTF-8 ロケールをあなたのデスクトップで使うことをお薦めします。ロケールの最初の部分がアプリケーションが提示するメッセージを決めます。例えば、”LANG=fr_FR.UTF-8” ロケールの下の gedit(1) (GNOME デスクトップのテキストエディター) は、必要なフォントとインプットメソッドがインストールされていれば、メニューをフランス語で提示しながら中国語の文字データーを表示し編集できます。

ロケールを”$LANG” 環境変数のみを用いて設定する事をお薦めします。UTF-8 ロケールの下で”LC_*” 変数 (locale(1) 参照下さい) の複雑な組み合わせ設定する意味はあまり無いと考えます。

プレーンな英語のテキストですら非 ASCII 文字を含んでいるかもしれません、例えば左右のクオーテーションマークは ASCII の中にでは利用できません。

```
b’’ “b’’double quoted textb’’” b’’
b’’ ‘b’’single quoted textb’’’ b’’
```

ASCII のプレーンテキストを UTF-8 のテキストに変換した時には、オリジナルの ASCII のテキストとまったく同じ内容とサイズとなります。ですから、UTF-8 ロケールを採用して何ら失うものはありません。

一部のプログラムは I18N をサポートした後でより多くのメモリーを消費するようになります。それらのプログラムは、実行速度最適化のために内部的に UTF-32 (UCS4) で Unicode のサポートをコードされていて、選ばれたロケールに無関係にそれぞれの ASCII 文字データー毎に 4 バイトを消費するからです。ここでも、UTF-8 ロケールを使ったからといって何も失うわけではありません。

ベンダー固有の旧式非 UTF-8 の符号化システムは、多くの国でグラフィック文字のような一部文字に関して仔細だが困惑する相違がありがちでした。現代的な OS が UTF-8 システムを採用したことはこのような符号化方式 (エンコーディング) の問題を実質的に解決しました。

8.4.3 ロケールの再設定

システムが特定のロケールにアクセスするために、ロケールデーターはロケールデーターベースにコンパイルされなければいけません。(`locales-all` パッケージをインストールしない限り、Debian システムは全ての利用可能なロケールを事前にコンパイルして提供されません。) コンパイルできるサポートされているロケールの全リストは”`/usr/share/i18n/SUPPORTED`” に記載されています。全ての正確なロケール名がこのファイルにリストされています。次のようにすると全ての既にバイナリー形式にコンパイルされ使える UTF-8 ロケールがリストされます。

```
$ locale -a | grep utf8
```

次のコマンド実行をすると `locales` パッケージが再設定されます。

```
# dpkg-reconfigure locales
```

このプロセスは 3 段階あります。

1. 使用可能なロケールのリストを更新
2. それをバイナリー形式にコンパイル
3. PAM (項4.5参照下さい) によって使われるように”`/etc/default/locale`” 中のシステム全体のデフォールトのロケール値を設定

使用可能なロケールには、”`en_US.UTF-8`” と”`UTF-8`” 付きの全ての関心のある言語が含まれているべきです。

米国英語での推奨のデフォールトロケールは”`en_US.UTF-8`” です。他の言語では”`UTF-8`” 付きのロケールを選ぶようにして下さい。これらの設定の内のいずれを使おうとも、いかなる国際化文字でも扱えます。

注意
ロケールを”`C`” に設定すると、メッセージは米国英語になりますが、ASCII 文字しか扱えなくなります。

8.4.4 ”$LANG” 環境変数の値

”`$LANG`” 環境変数の値は多くのアプリケーションによって設定や変更されます。

- Linux コンソールプログラムに関しては、login(1) の PAM 機構によって初期設定
- 全ての X プログラムに関しては、ディスプレーマネージャーの PAM 機構によって初期設定
- リモートコンソールプログラムに関しては、ssh(1) の PAM 機構によって初期設定
- 全ての X プログラムに関しては、gdm3(1) のような一部ディスプレーマネージャーによって変更
- 全ての X プログラムに関しては、”`~/.xsessionrc`” を経由する X セッションの起動コードによって変更 (`lenny` の機能)
- 全てのコンソールプログラムに関しては、”`~/.bashrc`” 等のシェルの起動コードによって変更

ティップ
互換性を最大限に考えると、システムワイドのデフォールトロケールを”`en_US.UTF-8`” と設定するのが賢明です。

8.4.5 X Window の下でのみ特定ロケール

PAM のカスタム化 (項4.5参照下さい) を使えば、システムワイドのデフォールトロケールに関わらず、特定のロケールを X Window の下だけで選ぶ事ができます。

この環境は安定度を確保したままあなたに最良のデスクトップ経験を提供します。X Window システムが機能していないときでも読めるメッセージを表示する機能する文字ターミナルに常にアクセス出きます。中国語や日本語や韓国語のように非ローマ文字を使う言語では、これは非常に重要です。

注意
X セッションマネージャーパッケージが改良されれば別の方法が利用可能になるかもしれませんが、一般的かつ基本的なロケールの設定方法として以下をお読み下さい。gdm3(1) に関して、X セッションのロケールはメニューから選べると認識しています。

”/etc/pam.d/gdm3” のような PAM 設定ファイル中で言語の環境変数を定義する場所は次の行により定義されます。

```
auth    required        pam_env.so read_env=1 envfile=/etc/default/locale
```

これを次のように変更します。

```
auth    required        pam_env.so read_env=1 envfile=/etc/default/locale-x
```

日本語の場合、”-rw-r--r-- 1 root root” パーミッションで次のように”/etc/default/locale-x” ファイルを作成します。

```
LANG=”ja_JP.UTF-8”
```

他のプログラムのためにデフォールトの”/etc/default/locale” ファイルは次のように元のままにします。

```
LANG=”en_US.UTF-8”
```

これはロケールをカスタム化する最も一般的なテクニックで、gdm3(1) 自身のメニュー選択ダイアローグを地域化します。

この場合の代策として、”~/.xsessionrc” ファイル使って簡単にロケールを変更してもいいです。

8.4.6 ファイル名の符号化方式

クロスプラットフォームのデーター交換 (項10.1.7参照下さい) のために、特定の符号化方式 (エンコーディング) でファイルシステムをマウントする必要があるかもしれません。例えば、vfat ファイルシステムに関して mount(8) はオプション無しの場合 CP437 とみなします。ファイル名に UTF-8 とか CP932 を使うためには明示的にマウントオプションを提供する必要があります。

注意
GNOME のような現代的なデスクトップ環境の下では、デスクトップアイコンを右クリックし”Drive” タブをクリックし”Setting” を開くようにクリックし”Mount options:” に”utf8” を入力すれば、ホットプラグできる USB メモリーを自動マウントする時のマウントオプションを設定できます。このメモリースティックを次にマウントする機会には UTF-8 でのマウントが有効です。

注意
もしシステムをアップグレードしたり旧式非 UTF-8 システムからディスクを移動したりする場合には、非 ASCII 文字のファイル名は ISO-8859-1 とか eucJP 等の今は非推奨の歴史的符号化方式で符号化をしているかもしれません。テキスト変換ツールの助力を得て、ファイル名を UTF-8 に変換します。項11.1を参照下さい。

Samba は新規クライアント (Windows NT、200x、XP) には Unicode を使いますが、旧式クライアント (DOS、Windows 9x/Me) には CP850 をデフォールトで使います。この旧式クライアントへのデフォールトは”/etc/samba/smb.conf” ファイル中の”dos charset” を使って例えば日本語なら CP932 等と変更できます。

8.4.7 地域化されたメッセージと翻訳された文書

Debian システム中で表示されるエラーメッセージや標準のプログラムの出力やメニューやマニュアルページ等のテキストメッセージや文書の多くに翻訳があります。ほとんどの翻訳行為のバックエンドツールとして GNU gettext(1) コマンドツールチェインが使われています。

”Tasks” → ”Localization” の下の aptitude(8) リストは地域化されたメッセージをアプリケーションに追加したり翻訳された文書を提供する有用なバイナリーパッケージの徹底的なリストを提供します。

例えば、`manpages-<LANG>` パッケージをインストールするとマンページで地域化したメッセージに使えるようになります。<programname> に関するイタリア語のマンページを”`/usr/share/man/it/`” から読むには、次を実行します。

```
LANG=it_IT.UTF-8 man <programname>
```

8.4.8 ロケールの効果

sort(1) を使う際のソートオーダー (並べ替え順序) はロケールの言語選択に影響されます。スペイン語と英語のロケールでは異なる並べ替えが違います。

ls(1) の日付フォーマットはロケールに影響されます。”`LANG=C ls -l`” と”`LANG=en_US.UTF-8`” の日付フォーマットは違います (項9.2.5参照下さい)。

数字の区切り方はロケール毎に異なります。例えば、英語のロケールでは一千一百一点一は”`1,000.1`” と表示されますが、ドイツ語のロケールでは”`1.000,1`” と表示されます。スプレッドシートプログラムでこの違いに出会うかもしれません。

Chapter 9

システムに関するティップ

主にコンソールからシステムを設定や管理する基本的なティップを次に記します。

9.1 screen プログラム

screen(1) は、ネットワーク接続中断をサポートするので信頼性が低く断続的な接続経由でリモートサイトをアクセスする人にとっては非常に有用なツールです。

パッケージ	ポプコン	サイズ	説明
screen	V:127, I:281	1013	VT100/ANSI ターミナルエミュレーションを使ってのターミナルマルチプレクサ
tmux	V:34, I:136	830	代替のターミナルマルチプレクサ (代わりに”Control-B” を用いる)

Table 9.1: ネットワーク切断の中断をサポートするプログラムのリスト

9.1.1 screen(1) の使い方のシナリオ

screen(1) は複数のプロセスを 1 つのターミナルウィンドウでうまく動作させるのみならず、接続が中断してもリモートシェルプロセスを生き延びさせる事もできます。screen(1) の使われ方の典型的シナリオは次です。

1. リモート機器にログインします。
2. 単一のコンソール上で screen を起動します。
3. ^A c (”Control-A” に続いて”c”) によって作られた screen のウィンドウ中で複数のプログラムを実行します。
4. ^A n (”Control-A” に続いて”n”) によって、複数の screen のウィンドウ間を切り替えます。
5. 突然ターミナルを離れる必要ができたけれども、接続を継続してあなたが実行中の作業を失いたくありません。
6. 次のようないかなる方法ででも、screen のセッションをデタッチできます。
   - 暴力的にネットワーク接続を引き抜く
   - ^A d (”Control-A” に続いて”d”) とタイプしてリモート接続から手動でログアウト
   - ^A DD (”Control-A” に続いて”DD”) とタイプして screen をデタッチしてログアウト

7. 同じリモート機器に (たとえ異なるターミナルからでも) 再びログインします。
8. screen を”screen -r” として起動します。
9. screen は全アクティブなプログラムが実行されている過去の全 screen ウィンドウを魔法のようにリアタッチします。

ティップ
screen を使うと、切断してもプロセスをアクティブにしておけその後で再接続した時にリアタッチできるので、ダイヤルアップやパケット接続のような計量されたネットワーク接続での接続料金の節約ができます。

9.1.2 screen コマンドのキーバインディング

screen セッションではコマンドキーストローク以外の全てのキーボード入力は現在のウィンドウに送られます。全ての screen コマンドキーストロークは ^A (”Control-A”) と単一キー [プラス何らかのパラメーター] をタイプすることによって入力されます。次に覚えておくべき重要なコマンドキーストロークを記します。

キーバインディング	意味
^A ?	ヘルプスクリーンを表示 (キーバインディングを表示)
^A c	新規ウィンドウを作成しそれに切り替える
^A n	次のウィンドウに切り替える
^A p	前のウィンドウに切り替える
^A 0	0 番のウィンドウに切り替える
^A 1	1 番のウィンドウに切り替える
^A w	ウィンドウのリストを表示
^A a	Ctrl-A を現在のウィンドウにキーボード入力として送る
^A h	現在のウィンドウのハードコピーをファイルに書く
^A H	現在のウィンドウのファイルへのロギングを開始/終了する
^A ^X	ターミナルをロック (パスワードで保護)
^A d	ターミナルから screen のセッションをデタッチ
^A DD	screen のセッションをデタッチしてログアウト

Table 9.2: screen キーバインディングのリスト

詳細は screen(1) を参照下さい。

9.2 データーの記録と表現

9.2.1 ログデーモン

多くのプログラムは”/var/log/” ディレクトリーの下にそれぞれの活動を記録します。

- システムログデーモン: rsyslogd(8)

項3.2.5と項3.2.4を参照下さい。

9.2.2 ログアナライザー

注目すべきログアナライザー (aptitude(8) で”~Gsecurity::log-analyzer”) を次に記します。

パッケージ	ポプコン	サイズ	説明
logwatch	V:16, I:18	2265	綺麗な出力の Perl で書かれたログアナライザー
fail2ban	V:112, I:123	2092	複数回の認証エラーを発生させる IP を使用禁止にします
analog	V:4, I:109	3534	ウェッブサーバーのログアナライザー
awstats	V:9, I:15	6910	強力で機能の多いウェッブサーバーのログアナライザー
sarg	V:3, I:3	843	squid の分析レポートジェネレター
pflogsumm	V:1, I:4	111	Postfix ログ項目サマライザー
syslog-summary	V:0, I:2	30	syslog ログファイルの内容をまとめる
fwlogwatch	V:0, I:0	479	ファイアウォールログアナライザー
squidview	V:0, I:1	189	squid の access.log ファイルのモニターと分析
swatch	V:0, I:0	101	正規表現マッチ、ハイライト、フック機能付きログファイルビューワー
crm114	V:0, I:0	1119	制御可能な正規表現切断機とスパムフィルター (CRM114)
icmpinfo	V:0, I:0	44	ICMP メッセージの解釈

Table 9.3: システムログアナライザーのリスト

注意
CRM114 はTRE 正規表現ライブラリーを使うファジーなフィルターを書く言語インフラを提供します。そのよくある応用はスパムメールのフィルターですが、ログアナライザーとしても使えます。

9.2.3 シェルの活動を綺麗に記録

単に script(1) を使ってシェル活動を記録すると (項1.4.9参照下さい)、コントロール文字の入ったファイルが生成されます。このような事は次のようにして col(1) を使うことで避けられます。

```
$ script
Script started, file is typescript
```

何なりとします…そして `script` から脱出するために `Ctrl-D` を押します。

```
$ col -bx <typescript >cleanedfile
$ vim cleanedfile
```

(例えば、initramfs 中のブートプロセスの途中のように) `script` が無い場合には、その代わりに次のようにすれば良いです。

```
$ sh -i 2>&1 | tee typescript
```

ティップ
`gnome-terminal` のような一部の `x-terminal-emulator` は記録できます。スクロールバック用バッファーを拡大するのが良いかもしれません。

ティップ
screen(1) を"`^A H`" と一緒に使っても (項9.1.2参照下さい) コンソールの記録が録れます。

ティップ
emacs(1) を"`M-x shell`" か"`M-x eshell`" か"`M-x term`" と一緒に使ってもコンソールの記録が録れます。後で"`C-x C-w`" とするとバッファーをファイルに書き出せます。

9.2.4 テキストデーターのカスタム化表示

more(1) や less(1) 等のページャーツール (項1.4.5参照下さい) や、ハイライトやフォーマット用のカスタムツール (項11.1.8参照下さい) はテキストデーターを綺麗に表示できますが、汎用エディター (項1.4.6参照下さい) が最も汎用性がありカスタム化が可能です。

ティップ
vim(1) やそのページャーモードのエリアス view(1) では、"`:set hls`" とするとハイライトサーチが可能になります。

9.2.5 時間と日付のカスタム化表示

"`ls -l`" コマンドによる時間と日付のデフォールトの表示形式はロケール (値は項1.2.6を参照下さい) に依存します。"`$LANG`" 変数が最初に参照され、それを"`$LC_TIME`" 変数によりオーバーライドする事ができます。

実際の各ロケールでのデフォールトの表示形式は使われた標準 C ライブラリー (`libc6` パッケージ) のバージョンに依存します。つまり Debian の異なるリリースは異なるデフォールトです。

ロケール以上にこの時間や日付の表示フォーマットをカスタム化したいと真摯に望むなら、"`--time-style`" 引数か"`$TIME_STYLE`" 値を使って時間スタイル値を設定するべきです (ls(1) と date(1) と"`info coreutils 'ls invocation'`" を参照下さい)。

時間スタイル値	ロケール	時間と日付の表示
`iso`	不問	`01-19 00:15`
`long-iso`	不問	`2009-01-19 00:15`
`full-iso`	不問	`2009-01-19 00:15:16.000000000 +0900`
`locale`	`C`	`Jan 19 00:15`
`locale`	`en_US.UTF-8`	`Jan 19 00:15`
`locale`	`es_ES.UTF-8`	`ene 19 00:15`
`+%d.%m.%y %H:%M`	不問	`19.01.09 00:15`
`+%d.%b.%y %H:%M`	`C` または `en_US.UTF-8`	`19.Jan.09 00:15`
`+%d.%b.%y %H:%M`	`es_ES.UTF-8`	`19.ene.09 00:15`

Table 9.4: `wheezy` での"`ls -l`" コマンドによる時間と日付の表示例

ティップ
例えば"`alias ls='ls --time-style=+%d.%m.%y\ %H:%M'`" とするコマンドエリアスを使うことでコマンドライン上に長いオプションを入力しなくてよくなります (項1.5.9を参照下さい)。

ティップ
このような iso フォーマットは ISO 8601 に準拠しています。

9.2.6 着色化されたシェル出力

殆どの現代的なターミナルへのシェル出力は ANSI エスケープコードを使って着色化できます ("`/usr/share/doc/xter`
を参照下さい)。

例えば、次を試してみて下さい:

```
$ RED=$(printf "\x1b[31m")
$ NORMAL=$(printf "\x1b[0m")
$ REVERSE=$(printf "\x1b[7m")
$ echo "${RED}RED-TEXT${NORMAL} ${REVERSE}REVERSE-TEXT${NORMAL}"
```

9.2.7 着色化されたコマンド

着色化されたコマンドは対話環境で出力を検査するのに便利です。私は、私の"~/.bashrc" に次を含めています。

```
if [ "$TERM" != "dumb" ]; then
    eval "`dircolors -b`"
    alias ls='ls --color=always'
    alias ll='ls --color=always -l'
    alias la='ls --color=always -A'
    alias less='less -R'
    alias ls='ls --color=always'
    alias grep='grep --color=always'
    alias egrep='egrep --color=always'
    alias fgrep='fgrep --color=always'
    alias zgrep='zgrep --color=always'
else
    alias ll='ls -l'
    alias la='ls -A'
fi
```

エリアスを使うことで色効果を対話コマンド使用時に限定します。こうすると less(1) 等のページャープログラムの下でも色を見られるので、環境変数"export GREP_OPTIONS='--color=auto'" をエキスポートするより都合が良いです。他のプログラムにパイプする際に色を使いたくないなら、先ほどの"~/.bashrc" 例中で代わりに"--color=auto" とします。

ティップ
このような着色するエリアスは、対話環境でシェルを"TERM=dumb bash" として起動することで無効にできます。

9.2.8 複雑な反復のためにエディターでの活動を記録

複雑な反復のためにエディターでの活動を記録できます。

Vim の場合以下のようにします。

- "qa": 名前付きレジスタ"a" にタイプした文字の記録を開始。
- …エディターでの活動
- "q": タイプした文字の記録を終了。
- "@a": レジスター"a" の内容を実行。

Emacs の場合は以下のようにします。

- "C-x (": キーボードマクロの定義開始。
- …エディターでの活動
- "C-x)": キーボードマクロの定義終了。
- "C-x e": キーボードマクロの実行。

9.2.9 X アプリケーションの画像イメージの記録

xterm の表示を含めた、X アプリケーションの画像イメージを記録するにはいくつか方法があります。

パッケージ	ポプコン	サイズ	コマンド
xbase-clients	I:26	46	xwd(1)
gimp	V:68, I:341	22313	GUI メニュー
imagemagick	I:400	218	import(1)
scrot	V:8, I:80	70	scrot(1)

Table 9.5: 画像の操作ツールのリスト

9.2.10 設定ファイルの変更記録

DVCS システムを使って設定ファイルの変更を記録する専用ツールがあります。

パッケージ	ポプコン	サイズ	説明
etckeeper	V:27, I:32	162	Git (デフォールト) か Mercurial か Bazaar を使って設定ファイルとそのメタデーターを保存 (新規)
changetrack	V:0, I:0	71	RCS を使って設定ファイルを保存 (旧式)

Table 9.6: VCS 中に設定の履歴を記録するパッケージのリスト

git(1) which put entire "/etc" 全てを VCS のコントロール下に置くように、git(1) とともに etckeeper パッケージを使うことをお薦めします。そのインストール案内とチュートリアルは"/usr/share/doc/etckeeper/README.gz" にあります。

本質的に"sudo etckeeper init" を実行すると、"/etc" に関する git レポジトリーが、徹底的な手順を踏む特別のフックスクリプト付きでちょうど項10.6.5と同様に初期化されます。

あなたが設定を変える毎に、普通に git(1) を使って記録できます。パッケージ管理コマンドを使うと、変更に関して上手に毎回自動記録もします。

ティップ
"/etc" の変更履歴を閲覧するには、"sudo GIT_DIR=/etc/.git gitk" と実行すると、新規インストールされたパッケージや削除されたパッケージやパッケージのバージョンの変化が一目瞭然です。

9.3 プログラム活動の監視と制御と起動

プログラム活動は専用ツールを用いて監視と制御できます。

ティップ
procps パッケージはプログラム活動の監視と制御と起動の基本中の基本を提供します。このすべてを習得するべきです。

パッケージ	ポプコン	サイズ	説明
coreutils	V:891, I:999	17478	nice(1): スケジューリングの優先順位の変更してプログラムを実行
bsdutils	V:673, I:999	393	renice(1): 実行中プロセスのスケジューリングの優先順位を変更
procps	V:739, I:999	792	"/proc" ファイルシステムのユーティリティー: ps(1) と top(1) と kill(1) と watch(1) 等
psmisc	V:427, I:845	679	"/proc" ファイルシステムのユーティリティー: killall(1) と fuser(1) と pstree(1) と pstree(1)
time	V:15, I:279	82	time(1): 時間に関するシステムリソース使用状況を報告するためにプログラムを実行
sysstat	V:161, I:183	1918	sar(1)、iostat(1)、mpstat(1)、…: Linux 用のシステムパーフォーマンスツール
isag	V:0, I:3	116	sysstat の対話型システムアクティビティーグラフ化ソフト
lsof	V:391, I:946	451	lsof(8): "-p" オプションを使い実行中のプロセスが開いているファイルをリスト
strace	V:16, I:153	2367	strace(1): システムコールやシグナルを追跡
ltrace	V:1, I:21	363	ltrace(1): ライブラリーコールを追跡
xtrace	V:0, I:0	353	xtrace(1): X11 のクライアントとサーバーの間の通信を追跡
powertop	V:9, I:217	662	powertop(1): システムの電力消費情報
cron	V:805, I:997	263	cron(8) デーモンからバックグランドでスケジュール通りプロセスを実行
anacron	V:409, I:482	99	1 日 24 時間動作でないシステム用の cron 類似のコマンドスケジューラー
at	V:162, I:310	161	at(1) と batch(1) コマンド: 特定の時間や特定のロードレベル以下でジョブを実行

Table 9.7: プログラム活動の監視と制御のツールのリスト

9.3.1 プロセスの時間計測

コマンドが起動したプロセスにより使われた時間を表示します。

```
# time some_command >/dev/null
real    0m0.035s        # b'' 壁 b''b'' 時 b''b'' 計 b''b'' の b''b'' 時 b''b'' 間 b'' (b'' 実
    b''b'' 経 b''b'' 過 b''b'' 時 b''b'' 間 b'')
user    0m0.000s        # b'' ユ b''b'' ー b''b'' ザ b''b'' ー b''b'' モ b''b'' ー b''b'' ド
    b''b'' の b''b'' 時 b''b'' 間 b''
sys     0m0.020s        # b'' カ b''b'' ー b''b'' ネ b''b'' ル b''b'' モ b''b'' ー b''b'' ド
    b''b'' の b''b'' 時 b''b'' 間 b''
```

9.3.2 スケジューリングのプライオリティー

ナイス値はプロセスのスケジューリングのプライオリティーを制御するのに使われます。

ナイス値	スケジューリングのプライオリティー
19	最低優先順位プロセス (ナイス)
0	ユーザーにとっての非常に高優先順位プロセス
-20	root にとっての非常に高優先順位プロセス (非ナイス)

Table 9.8: スケジューリングのプライオリティーのためのナイス値のリスト

```
# nice  -19 top                                     # b'' 非 b''b'' 常 b''b'' に b''b'' ナ
    b''b'' イ b''b'' ス b''
# nice --20 wodim -v -eject speed=2 dev=0,0 disk.img # b'' 非 b''b'' 常 b''b'' に b''b'' 高
    b''b'' 速 b''
```

極端なナイス値はシステムに害を与えるかもしれません。本コマンドは注意深く使用下さい、

9.3.3 ps コマンド

Debian 上の ps(1) コマンドは BSD と SystemV 機能の両方をサポートしプロセスの活動を静的に特定するのに有用です。

スタイル	典型的コマンド	特徴
BSD	`ps aux`	%CPU %MEM を表示
System V	`ps -efH`	PPID を表示

Table 9.9: ps コマンドのスタイルのリスト

ゾンビ (動作していない) 子プロセスに関して、”`PPID`” フィールドで識別される親プロセス ID を使ってプロセスを停止できます。

pstree(1) コマンドはプロセスの木 (ツリー) を表示します。

9.3.4 top コマンド

Debian 上の top(1) は機能が豊富で、どのプロセスがおかしな動きをしているかを動的に識別することに役立ちます。

それはインタラクティブなフルスクリーンプログラムです。”h”-キーを押すことで使用法のヘルプが得られ、”q”-キーを押すことで終了できます。

9.3.5 プロセスによって開かれているファイルのリスト

プロセス ID (PID)、例えば 1 を使うプロセスによって開かれている全ファイルは次のようにしてリストできます。

```
$ sudo lsof -p 1
```

PID=1 は通常 `init` プログラムです。

9.3.6 プログラム活動の追跡

プラグラムの活動状況は、システムコールとシグナルは strace(1) で、ライブラリーコールは ltrace(1) で、X11 のクライアントとサーバーの通信は xtrace(1) でプラグラムの活動状況を追跡できます。

`ls` コマンドのシステムコールを次のようにして追跡できます。

```
$ sudo strace ls
```

9.3.7 ファイルやソケットを使っているプロセスの識別

例えば”`/var/log/mail.log`” 等のファイルを使っているプロセスは fuser(1) によって次のようにして識別できます。

```
$ sudo fuser -v /var/log/mail.log
                     USER        PID ACCESS COMMAND
/var/log/mail.log:   root       2946 F.... rsyslogd
```

”`/var/log/mail.log`” ファイルが rsyslogd(8) コマンドによって書込みのために開かれている事が分かります。

例えば”`smtp/tcp`” 等のソケットを使っているプロセスは fuser(1) によって次のようにして識別できます。

```
$ sudo fuser -v smtp/tcp
                     USER        PID ACCESS COMMAND
smtp/tcp:            Debian-exim   3379 F.... exim4
```

SMTP ポート (25) への TCP 接続を処理するためにあなたのシステムでは exim4(8) が実行されている事がこれで分かります。

9.3.8 一定間隔でコマンドを反復実行

watch(1) はプログラムを一定間隔で反復実行しながらフルスクリーンでその出力を表示します。

```
$ watch w
```

こうすると 2 秒毎更新でシステムに誰がログオンしているかを表示します。

9.3.9 ファイルに関してループしながらコマンドを反復実行

例えばグロブパターン”`*.ext`” へのマッチ等の何らかの条件にマッチするファイルに関してループしながらコマンドを実行する方法がいくつかあります。

- シェルの for-loop 法 (項12.1.4参照下さい):

```
for x in *.ext; do if [ -f "$x"]; then command "$x" ; fi; done
```

- find(1) と xargs(1) の組み合わせ:

```
find . -type f -maxdepth 1 -name '*.ext' -print0 | xargs -0 -n 1 command
```

- コマンド付きの"-exec" オプションを使って find(1):

```
find . -type f -maxdepth 1 -name '*.ext' -exec command '{}' \;
```

- 短いシェルスクリプト付きの"-exec" オプションを使って find(1):

```
find . -type f -maxdepth 1 -name '*.ext' -exec sh -c "command '{}' && echo 'successful'" \;
```

上記の例はスペースを含む等の変なファイル名でも適正に処理できるように書かれています。find(1) に関する上級の使用法の詳細は項10.1.5を参照下さい。

9.3.10 GUI からプログラムをスタート

コマンドラインインターフェース (CLI) の場合、$PATH 環境変数で指定されるディレクトリー中で最初にマッチした名前のプログラムが実行されます。項1.5.3 を参照ください。

freedesktop.org スタンダード準拠の グラフィカルユーザーインターフェース (GUI) の場合、/usr/share/applications/ ディレクトリー中の *.desktop ファイルにより各プログラムの GUI メニュー表示に必要なアトリビュートが提供されます。項7.2.2 参照ください。

例えば chromium.desktop ファイルは、プログラム名の"Name" や、プログラムの実行パスと引数の"Exec" や、使用するアイコンの"Icon" 等の属性 (Desktop Entry Specification 参照) を"Chromium Web Browser" に関して以下のようにして定義します:

```
[Desktop Entry]
Version=1.0
Name=Chromium Web Browser
GenericName=Web Browser
Comment=Access the Internet
Comment[fr]=Explorer le Web
Exec=/usr/bin/chromium %U
Terminal=false
X-MultipleArgs=false
Type=Application
Icon=chromium
Categories=Network;WebBrowser;
MimeType=text/html;text/xml;application/xhtml_xml;x-scheme-handler/http;x-scheme-handler/ ↩
    https;
StartupWMClass=Chromium
StartupNotify=true
```

これは簡略化しすぎた記述ですが、*.desktop ファイルは以下のようにしてスキャンされます。

デスクトップ環境は $XDG_DATA_HOME と $XDG_DATA_DIR 環境変数を設定します。例えば GNOME 3 では:

- $XDG_DATA_HOME が未定義。(デフォルト値の $HOME/.local/share が使われます。)
- $XDG_DATA_DIRS は /usr/share/gnome:/usr/local/share/:/usr/share/ に設定されます。

以上により、ベースディレクトリー (XDG Base Directory Specification 参照) や applications ディレクトリーは以下となります。

- `$HOME/.local/share/` → `$HOME/.local/share/applications/`
- `/usr/share/gnome/` → `/usr/share/gnome/applications/`
- `/usr/local/share/` → `/usr/local/share/applications/`
- `/usr/share/` → `/usr/share/applications/`

`*.desktop` ファイルはこれらの `applications` ディレクトリーでこの順番でスキャンされます。

ティップ
ユーザーによるカスタムの GUI メニュー項目は `*.desktop` ファイルを `$HOME/.local/share/applications/` ディレクトリーに追加することで生成できます。

ティップ
同様に、もしこれらのベースディレクトリーの下の `autostart` ディレクトリーの中に `*.desktop` ファイルが作成されれば、`*.desktop` ファイル中に指定されたプログラムがデスクトップ環境が起動された時点に自動実行されます。Desktop Application Autostart Specification を参照ください。

ティップ
同様に、もし `$HOME/Desktop` ディレクトリーの中に `*.desktop` ファイルが作成され、デスクトップ環境がローンチャーアイコンを表示する機能を有効としていれば、そこに指定されたプログラムがアイコンをクリックした際に実行されます。xdg-user-dirs-update(1) を参照ください。

9.3.11 スタートするプログラムのカスタム化

一部のプログラムは他のプログラムを自動的にスタートします。このプロセスをカスタム化する上でのチェックポイントを次に記します。

- アプリケーション設定メニュー:
  - GNOME デスクトップ: ”Settings” → ”System” → ”Details” → ”Default Applications”
  - KDE デスクトップ: ”K” → ”Control Center” → ”KDE Components” → ”Component Chooser”
  - Iceweasel ブラウザー: ”Edit” → ”Preferences” → ”Applications”
  - mc(1): ”`/etc/mc/mc.ext`”
- ”`$BROWSER`” や”`$EDITOR`” や”`$VISUAL`” や”`$PAGER`” といった環境変数 (eviron(7) 参照下さい)
- ”`editor`” や”`view`” や”`x-www-browser`” や”`gnome-www-browser`” や”`www-browser`” 等のプログラムに関する update-alternatives(8) システム (項1.4.7参照下さい)
- MIME タイプとプログラムと関係づける、”`~/.mailcap`” や”`/etc/mailcap`” ファイルの内容 (mailcap(5) 参照下さい)
- ファイル拡張子と MIME タイプとプログラムと関係づける、”`~/.mime.types`” や”`/etc/mime.types`” ファイルの内容 (run-mailcap(1) 参照下さい)

ティップ
update-mime(8) は”`/etc/mailcap.order`” ファイルを使って”`/etc/mailcap`” ファイルを更新します (mailcap.order(5) 参照下さい)。

ティップ
debianutils パッケージは、どのエディターやページャーやウェッブブラウザーを呼び出すかに関してそれぞれ賢明な判断をする sensible-browser(1) や sensible-editor(1) や sensible-pager(1) を提供します。これらのシェルスクリプトを読む事をお薦めします。

ティップ
X の下で mutt のようなコンソールアプリケーションをあなたの好むアプリケーションとして実行するには、次のようにして X アプリケーションを作成し、前記の方法であなたの好む起動されるアプリケーションとして”/usr/local/bin/mutt-term” を設定します。

```
# cat /usr/local/bin/mutt-term <<EOF
#!/bin/sh
gnome-terminal -e ”mutt \$@”
EOF
chmod 755 /usr/local/bin/mutt-term
```

9.3.12 プロセスの停止

kill(1) を使ってプロセス ID を使ってプロセスを停止 (プロセスへシグナルを送信) します。

killall(1) や pkill(1) プロセスコマンド名や他の属性を使ってプロセスを停止 (プロセスへシグナルを送信) します。

シグナル値	シグナル名	機能
1	HUP	デーモンの再スタートします
15	TERM	普通に停止します
9	KILL	徹底的に停止します

Table 9.10: kill コマンドが良く使うシグナルのリスト

9.3.13 タスク 1 回実行のスケジュール

at(1) コマンドを次のように実行して 1 回だけのジョブをスケジュールします。

```
$ echo ’command -args’| at 3:40 monday
```

9.3.14 タスク定期実行のスケジュール

cron(8) コマンドを実行して定期的タスクをスケジュールします。crontab(1) と crontab(5) を参照下さい。

例えば foo というノーマルユーザーとして”crontab -e” コマンドを使って”/var/spool/cron/crontabs/foo” という crontab(5) ファイルを作成することでプロセスをスケジュールして実行する事ができます。

crontab(5) ファイルの例を次に記します。

```
# use /bin/sh to run commands, no matter what /etc/passwd says
SHELL=/bin/sh
# mail any output to paul, no matter whose crontab this is
MAILTO=paul
# Min Hour DayOfMonth Month DayOfWeek command (Day... are OR’ed)
# run at 00:05, every day
5  0  *  * *   $HOME/bin/daily.job >> $HOME/tmp/out 2>&1
```

```
# run at 14:15 on the first of every month -- output mailed to paul
15 14 1  * *   $HOME/bin/monthly
# run at 22:00 on weekdays(1-5), annoy Joe. % for newline, last % for cc:
0 22 *   * 1-5 mail -s "It's 10pm" joe%Joe,%%Where are your kids?%.%%
23 */2 1 2 *   echo "run 23 minutes after 0am, 2am, 4am ..., on Feb 1"
5  4 *   * sun echo "run at 04:05 every Sunday"
# run at 03:40 on the first Monday of each month
40 3 1-7 * *   [ "$(date +%a)" == "Mon" ] && command -args
```

ティップ
連続的に稼働していないシステムでは、機器のアップタイム上可能な限り指定間隔に近く定期的にコマンドをスケジュールするために anacron パッケージをインストールします。anacron(8) と anacrontab(5) を参照下さい。

ティップ
スケジュールされたシステムメインテナンススクリプトは、そのようなスクリプトを"/etc/cron.hourly/" か"/etc/cron.daily/" か"/etc/cron.weekly/" か"/etc/cron.monthly/" 中に置くことで root アカウントからそれらを定期的に実行できます。これらのスクリプトの実行時間は"/etc/crontab" と"/etc/anacrontab" でカスタム化できます。

9.3.15 Alt-SysRq キー

システム異常に対する保険はカーネルコンパイルオプションの" マジック SysRq キー" (SAK キー) によって提供されます。現在の Debian カーネルではこれがデフォールトです。Alt-SysRq に続いて次のキーの中から 1 つを押すことでシステムのコントロールを救済するマジックが起きます。

Alt-SysRq に続くキー	アクションの説明
r	X クラッシュの後でキーボードを **r**aw (生コード発生) モードから復旧
0	エラーメッセージを減らすべくコンソールログレベルを **0** と変更
k	全ての現仮想ターミナル上の全てのプロセスを停止 (**K**ll)
e	init(8) 以外の全てのプロセスに、SIGT**E**RM を送信
i	init(8) 以外の全てのプロセスに、SIGK**I**LL を送信
s	データーが壊れないように全てのマウントされたファイルシステムを **s**ync (同期) します。
u	全てのマウントされたファイルシステムを読出し専用で再マウント (アンマウント、**u**mount)
b	同期することもアンマウントする事も無しに、システムをリブート (Re**b**oot)

Table 9.11: SAK コマンドキーのリスト

ティップ
常軌を理解するためには、signal(7) と kill(1) と sync(1) のマンページを読みましょう。.

"Alt-SysRq s" と"Alt-SysRq u" と"Alt-SysRq r" の組み合わせは、非常に悪い状況からの脱出しシステムを止めることなく使えるキーボドアクセスを回復するのに有効です。

"/usr/share/doc/linux-doc-3.*/Documentation/sysrq.txt.gz" を参照下さい。

注意
Alt-SysRq 機能があることはユーザーに root 特権機能をアクセスさせることになるのでセキュリティーリスクと考える事もできます。”/etc/rc.local” 中に”echo 0 >/proc/sys/kernel/sysrq” とか”/etc/sysctl.conf” 中に”kernel.sysrq = 0” と置くと Alt-SysRq 機能を無効にできます。

ティップ
SSH ターミナルなどからは、”/proc/sysrq-trigger” に書き込むことで Alt-SysRq 機能が使えます。例えば、リモートのシェルプロンプトから”echo s > /proc/sysrq-trigger; echo u > /proc/sysrq-trigger” とすると、全てのマウントされたファイルシステムを **s**ync (同期) して **u**mount (アンマウント) します。

9.4 システム管理ティップ

9.4.1 だれがシステムを利用している?

だれがシステムを利用しているかは、次のようにしてチェックできます。

- who(1) は、誰がログオンしているかを表示します。
- w(1) は、誰がログオンしていて何をしているかを表示します。
- last(1) は、最後にログインしたユーザーのリストを表示します。
- lastb(1) は、最後にログイン失敗したユーザーのリストを表示します。

ティップ
”/var/run/utmp” と”/var/log/wtmp” はこのようなユーザー情報を保持します。login(1) と utmp(5) を参照下さい。

9.4.2 全員への警告

wall(1) を使うと、次のようにしてシステムにログオンしている全員にメッセージを送れます。

```
$ echo ”We are shutting down in 1 hour” | wall
```

9.4.3 ハードウエアーの識別

PCI 的デバイス (AGP、PCI-Express、CardBus、ExpressCard、等) では、(きっと”-nn” オプションとともに使う) lspci(8) がハードウエアー識別の良いスタート点です。

この代わりに、”/proc/bus/pci/devices” の内容を読むか、”/sys/bus/pci” の下のディレクトリーツリーを閲覧することでハードウエアーの識別ができます (項1.2.12参照下さい)。

9.4.4 ハードウエアー設定

GNOME や KDE のような現代的な GUI のデスクトップ環境ではほとんどのハードウエアー設定が付随する GUI 設定ツールを通じて管理できますが、それらの設定の基本的手法を知っておくのは良い事です。

上記で、ACPI はAPM より新しい電力管理システムの枠組みです。

パッケージ	ポプコン	サイズ	説明
pciutils	V:195, I:992	196	Linux PCI ユーティリティー: lspci(8)
usbutils	V:84, I:862	324	Linux USB ユーティリティー: lsusb(8)
pcmciautils	V:13, I:21	97	Linux のための PCMCIA ユーティリティー: pccardctl(8)
scsitools	V:0, I:3	390	SCSI ハードウエアー管理のためのツール集: lsscsi(8)
procinfo	V:0, I:13	135	”/proc” から得られるシステム情報: lsdev(8)
lshw	V:12, I:94	842	ハードウエアー設定に関する情報: lshw(1)
discover	V:41, I:947	90	ハードウエアー識別システム: discover(8)

Table 9.12: ハードウエアー識別ツールのリスト

パッケージ	ポプコン	サイズ	説明
console-setup	V:137, I:959	411	Linux コンソールのフォントとキーテーブルユーティリティー
x11-xserver-utils	V:282, I:534	511	X サーバーユーティリティー: xset(1)、xmodmap(1)
acpid	V:145, I:318	176	Advanced Configuration and Power Interface (ACPI) によって起こるイベントの管理のためのデーモン
acpi	V:17, I:302	45	ACPI デバイス上の情報を表示するユーティリティー
sleepd	V:0, I:0	86	非使用状況のときにラップトップをスリープさせるデーモン
hdparm	V:408, I:718	256	ハードディスクアクセスの最適化 (項9.5.9参照下さい)
smartmontools	V:134, I:197	2117	S.M.A.R.T. を使ってストレージシステムを制御監視
setserial	V:5, I:9	117	シリアルポートの管理ツール集
memtest86+	V:1, I:29	2391	メモリーハードウエアー管理のためのツール集
scsitools	V:0, I:3	390	SCSI ハードウエアー管理のためのツール集
setcd	V:0, I:1	35	コンパクトデバイスアクセス最適化
big-cursor	I:1	27	X のための大きなマウスカーソール

Table 9.13: ハードウエアー設定ツールのリスト

ティップ
最近のシステム上の CPU フリーケンシースケーリングは acpi_cpufreq のようなカーネルモジュールで管理されています。

9.4.5 システムとハードウエアーの時間

以下はシステムとハードウエアーの時間を MM/DD hh:mm, CCYY (月/日時: 分, 年) に設定します。

```
# date MMDDhhmmCCYY
# hwclock --utc --systohc
# hwclock --show
```

Debian システムでは時間は通常地域の時間が表示されますが、ハードウエアーとシステムの時間は通常 UTC(GMT) を使います。

ハードウエアー (BIOS) 時間が UTC に設定されている場合は、”/etc/default/rcS” の中の設定を”UTC=yes” と変更します。

Debian システムが使うタイムゾーンは以下のようにして再設定できます。

```
# dpkg-reconfigure tzdata
```

ネットワーク経由でシステムの時間を更新したい場合には、ntp や ntpdate や chrony 等のパッケージを使って NTP サービスを利用することを考えます。

ティップ
systemd の下では、ネットワーク時間同期には上記と代わり systemd-timesyncd を使います。詳細は systemd-timesyncd(8) を参照ください。

次を参照下さい。

- 正確な日時の管理ハウツー
- NTP 公共サービスプロジェクト
- ntp-doc パッケージ

ティップ
ntp パッケージ中の ntptrace(8) を使うと、NTP サービスの継がりを第一義的根源まで溯ることができます。

9.4.6 ターミナルの設定

文字コンソールと ncurses(3) システム機能を設定するのはいくつかの要素があります。

- ”/etc/terminfo/*/*” ファイル (terminfo(5))
- ”$TERM” 環境変数 (term(7))
- setterm(1)、stty(1)、tic(1)、toe(1)

もし xterm 用の terminfo エントリーが非 Debian の xterm でうまく機能しない場合には、リモートから Debian システムにログインする時にターミナルタイプ、”$TERM”、を”xterm” から”xterm-r6” のような機能限定版に変更します。詳細は”/usr/share/doc/libncurses5/FAQ” を参照下さい。”dumb” は”$TERM” の最低機能の共通項です。

9.4.7 音のインフラ

現在の Linux のためのサウンドカードのためのデバイスドライバーは Advanced Linux Sound Architecture (ALSA) で提供されています。ALSA は過去の Open Sound System (OSS) と互換性のためのエミュレーションモードを提供します。

ティップ
”`cat /dev/urandom > /dev/audio`” か speaker-test(1) を使ってスピーカをテストします。(^C で停止)

ティップ
音が出ない場合ですが、あなたのスピーカーが消音された出力につながっているかもしれません。現代的なサウンドシステムには多くの出力があります。`alsa-utils` パッケージ中の alsamixer(1) は音量や消音の設定をするのに便利です。

アプリケーションソフトはサウンドデバイスに直接アクセスするようにばかりでなく標準的なサウンドサーバーシステム経由で間接的にアクセスするように設定されているかもしれません。

パッケージ	ポプコン	サイズ	説明
`alsa-utils`	V:341, I:476	2283	ALSA を設定し使用するユーティリティー
`oss-compat`	V:2, I:29	20	ALSA の下で”`/dev/dsp not found`” エラーを防ぐ OSS 互換性
`jackd`	V:4, I:27	9	JACK Audio Connection Kit. (JACK) サーバー (低遅延)
`libjack0`	V:1, I:13	338	JACK Audio Connection Kit. (JACK) ライブラリー (低遅延)
`nas`	V:0, I:0	243	Network Audio System (NAS) サーバー
`libaudio2`	V:60, I:488	165	Network Audio System (NAS) ライブラリー
`pulseaudio`	V:350, I:471	6398	PulseAudio サーバー、ESD 代替
`libpulse0`	V:289, I:604	969	PulseAudio クライアント、ESD 代替
`libgstreamer1.0-0`	V:372, I:574	5280	GStreamer: GNOME サウンドエンジン
`libphonon4`	I:121	680	Phonon: KDE サウンドエンジン

Table 9.14: サウンドパッケージのリスト

各ポピュラーなデスクトップ環境では普通共通のサウンドエンジンがあります。アプリケーションに使われるそれぞれのサウンドエンジンはそれと異なるサウンドサーバーにつなぐようにもできます。

9.4.8 スクリーンセーバーの無効化

スクリーンセーバーを無効にするには、次のコマンドを使います。

環境	コマンド
Linux コンソール	`setterm -powersave off`
X Window (スクリーンセーバー消去)	`xset s off`
X Window (dpms 無効)	`xset -dpms`
X Window (スクリーンセーバーの GUI 設定)	`xscreensaver-command -prefs`

Table 9.15: スクリーンセーバーを無効にするコマンドのリスト

9.4.9 ブザー音の無効化

PC スピーカーのコネクタを外すとブザー音は確実に無効にできます。pcspkr カーネルモジュールを削除すると同じ事ができます。

次のようにすると bash(1) が使う readline(3) プログラムが”\a” (ASCII=7) に出会った際にブザー音を発生するのを防げます。

```
$ echo ”set bell-style none”>> ~/.inputrc
```

9.4.10 使用メモリー

メモリー使用状況を確認するのに 2 つのリソースがあります。

- ”/var/log/dmesg” 中にあるカーネルブートメッセージには、利用可能なメモリーの正確な全サイズが書かれています。
- free(1) や top(1) は稼働中システムのメモリーリソース情報を表示します。

以下がその例です。

```
# grep ’\] Memory’ /var/log/dmesg
[    0.004000] Memory: 990528k/1016784k available (1975k kernel code, 25868k reserved, 931k ↩
     data, 296k init)
$ free -k
             total       used       free     shared    buffers     cached
Mem:        997184     976928      20256          0     129592     171932
-/+ buffers/cache:     675404     321780
Swap:      4545576          4    4545572
```

「dmesg は 990 MB 空いているという一方、free -k は 320 MB 空いていると言っている。600 MB 以上行方不明だ…」と不思議かもれません。

”Mem:” 行の”used” のサイズが大きかったり”free” のサイズが小さかったりについて悩まないでおきましょう。それらの 1 行下の (次の例では 675404 と 321780) を読んで安心して下さい。

1GB=1048576k の DRAM (video システムがこのメモリーの一部を使用) が付いている私の MacBook では次のようになっています。

報告	サイズ
dmesg 中の全サイズ (Total)	1016784k = 1GB - 31792k
dmesg 中の未使用 (free)	990528k
shell 下での全 (total)	997184k
shell 下での未使用 (free)	20256k (しかし実質は 321780k)

Table 9.16: レポートされるメモリーサイズのリスト

9.4.11 システムのセキュリティーと整合性のチェック

ダメなシステム管理をするとあなたのシステムを外界からの攻撃にさらすことになるかもしれません。

システムのセキュリティーと整合性のチェックには、次の事から始めるべきです。

- debsums パッケージ、debsums(1) と項2.5.2を参照下さい。
- chkrootkit パッケージ、chkrootkit(1) 参照下さい。

- `clamav` パッケージ類、clamscan(1) と freahclam(1) 参照下さい。
- Debian セキュリティー FAQ.
- Securing Debian Manual.

パッケージ	ポプコン	サイズ	説明
logcheck	V:8, I:10	102	システムログの異常を管理者にメールするデーモン
debsums	V:5, I:42	107	MD5 チェックサムを使ってインストールされたパッケージファイルを検証するユーティリティー
chkrootkit	V:5, I:24	970	ルートキット検出ソフト
clamav	V:13, I:58	774	Unix 用アンチウィルスユーティリティー - コマンドラインインターフェース
tiger	V:2, I:3	7822	システムセキュリティーの脆弱性を報告
tripwire	V:2, I:3	11521	ファイルやディレクトリーの整合性チェックソフト
john	V:2, I:12	452	アクティブなパスワードクラッキングツール
aide	V:1, I:2	2063	先進的進入検出環境 - 静的ライブラリー
integrit	V:0, I:0	329	ファイル整合性確認プログラム
crack	V:0, I:1	149	パスワード推定プログラム

Table 9.17: システムセキュリティーや整合性確認のためのツールリスト

次のシンプルなスクリプトを使うと、典型的な間違いの全員書込み可のファイルパーミッションをチェックできます。

```
# find / -perm 777 -a \! -type s -a \! -type l -a \! \( -type d -a -perm 1777 \)
```

注意
debsums パッケージはローカルに保存された MD5 チェックサムを使うので、悪意ある攻撃に対抗するセキュリティー監査ツールとしては完全には信頼できません。

9.5 データー保存のティップ

Linux のlive CDs とかレスキューモードで debian-installer CDs であなたのシステムをブートすることでブートデバイス上のデーターストレージの再設定が簡単にできます。

9.5.1 ディスク空間の利用状況

ディスク空間使用状況は `mount` と `coreutils` と `xdu` パッケージが提供するプログラムで評価できます:

- mount(8) はマウントされたファイルシステム (= ディスク) すべてを報告します。
- df(1) はファイルシステムのディスク空間使用状況を報告します。
- du(1) はディレクトリーツリーのディスク空間使用状況を報告します。

ティップ
du(8) の出力を xdu(1x) に”`du -k . |xdu`” や”`sudo du -k -x / |xdu`” 等として注ぎ込むとそのグラフィカルでインタラクティブな表現が作成できます。

9.5.2 ディスクパーティション設定

ディスクのパーティションの設定に関して、fdisk(8) は標準と考えられてきていますが、parted(8) も注目に値します。” ディスクパーティションデーター” や” パーティションテーブル” や” パーティションマップ” や” ディスクラベル” は全て同意語です。

殆どの PC では、ディスクのパーティションデーターが最初のセクターつまり LBA セクター 0 (512 バイト) に保持される、古典的なマスターブートレコード (MBR) 方式が使われています。

注意
新規の Intel ベースの Mac のような拡張ファームウェアインターフェイス (EFI) 付きの一部 PC では、ディスクパーティションデーターをセクターの最初以外に保持する GUID Partition Table (GPT) 方式が使われています。

---

fdisk(8) はディスクパーティションツールの標準でしたが、parted(8) がそれを置き換えつつあります。

パッケージ	ポプコン	サイズ	**GPT**	説明
`util-linux`	V:891, I:999	4598	非サポート	fdisk(8) と cfdisk(8) を含む雑多なシステムユーティリティー
`parted`	V:363, I:561	304	サポート	GNU Parted ディスクパーティションとリサイズのプログラム
`gparted`	V:19, I:132	2046	サポート	`libparted` ベースの GNOME パーティションエディター
`gdisk`	V:278, I:513	852	サポート	GPT ディスク用パーティションエディター
`kpartx`	V:16, I:29	87	サポート	パーティション用のデバイスマッピングを作成するプログラム

Table 9.18: ディスクパーティション管理パッケージのリスト

注意
parted(8) はファイルシステムを生成やリサイズも出きるということですが、そのようなことは mkfs(8) (mkfs.msdos(8) と mkfs.ext2(8) と mkfs.ext3(8) と mkfs.ext4(8) と…) とか resize2fs(8) 等の最もよくメンテされている専用ツールを使って行う方がより安全です。

注意
GPT と MBR 間で切り替えるには、ディスクの最初数ブロックの内容を直接消去し (項9.7.6参照下さい)、”`parted /dev/sdx mklabel gpt`” か”`parted /dev/sdx mklabel msdos`” を使ってそれを設定する必要があります。ここで”`msdos`” がMBR のために使われていることを覚えておきます。

9.5.3 UUID を使ってパーティションをアクセス

あなたのパーティションの再設定やリムーバブルストレージメディアの起動順序はパーティションの名前を変えることになるかもしれませんが、それに首尾一貫してアクセスできます。もしディスクが複数ありあなたの BIOS がそれに首尾一貫したデバイス名をつけない時にも、これは役に立ちます。

- ”`-U`” オプションを使って mount(8) を実行すると”`/dev/sda3`” のようなファイル名を使うのではなく UUID を使ってブロックデバイスをマウントできます。
- ”`/etc/fstab`” (fstab(5) 参照下さい) は UUID を使えます。

- ブートローダー (項3.1.2) もまた UUID を使えます。

ティップ
ブロックスペシャルデバイスの UUID は blkid(8) を使って見極められます。

ティップ
リムーバブルストレージメディア等のデバイス名は、必要なら udev rules を使って静的になります。項3.3を参照下さい。

9.5.4 LVM2

LVM2 は Linux カーネル用の 論理ボリュームマネージャー です。LVM2 を使うと、ディスクパーティションを物理的ハードディスクではなく論理ボリューム上の作成できるようになります。

LVM には以下が必要です。

- Linux カーネルによる device-mapper サポート (Debian カーネルではデフォルト)
- ユーザースペースの device-mapper サポートライブラリー (`libdevmapper*` パッケージ)
- ユーザースペースの LVM2 ツール (`lvm2` パッケージ)

以下のマンページから LVM2 を学び始めましょう。

- lvm(8): LVM2 機構の基本 (全 LVM2 コマンドのリスト)
- lvm.conf(5): LVM2 の設定ファイル
- lvs(8): 論理ボリュームの情報を報告します
- vgs(8): ボリュームグループの情報を報告します
- pvs(8): 物理ボリュームの情報を報告します

9.5.5 ファイルシステム設定

ext4 ファイルシステム用に `e2fsprogs` パッケージは次を提供します。

- 新規の ext4 ファイルシステムを作成するための mkfs.ext4(8)
- 既存の ext4 ファイルシステムをチェックと修理するための fsck.ext4(8)
- ext4 ファイルシステムのスーパーブロックを設定するための tune2fs(8)
- debugfs(8) を使ってext4 ファイルシステムをインタラクティブにデバグします。(削除したファイルを回復する `undel` コマンドがあります。)

mkfs(8) と fsck(8) コマンドは各種ファイルシステム依存プログラム (`mkfs.fstype` や `fsck.fstype`) のフロントエンドとして `e2fsprogs` により提供されています。ext4 ファイルシステム用は、mkfs.ext4(8) と fsck.ext4(8) で、それぞれ mke2fs(8) と e2fsck(8) にシムリンクされています。

Linux によってサポートされる各ファイルシステムでも、類似コマンドが利用可能です。

パッケージ	ポプコン	サイズ	説明
e2fsprogs	V:576, I:999	1449	ext2/ext3/ext4 ファイルシステムのためのユーティリティー
reiserfsprogs	V:11, I:29	1132	Reiserfs ファイルシステムのためのユーティリティー
dosfstools	V:128, I:524	235	FAT ファイルシステムのためのユーティリティー (Microsoft: MS-DOS, Windows)
xfsprogs	V:21, I:98	3191	XFS ファイルシステムのためのユーティリティー (SGI: IRIX)
ntfs-3g	V:186, I:512	1479	NTFS ファイルシステムのためのユーティリティー (Microsoft: Windows NT, ⋯)
jfsutils	V:1, I:12	1577	JFS ファイルシステムのためのユーティリティー (IBM: AIX, OS/2)
reiser4progs	V:0, I:4	1373	Reiser4 ファイルシステムのためのユーティリティー
hfsprogs	V:0, I:8	356	HFS と HFS Plus ファイルシステムのためのユーティリティー (Apple: Mac OS)
btrfs-progs	V:38, I:64	4027	btrfs ファイルシステムのためのユーティリティー
zerofree	V:3, I:94	25	ext2/3/4 ファイルシステムのフリーブロックをゼロにセットするプログラム

Table 9.19: ファイルシステム管理用パッケージのリスト

ティップ
Ext4 ファイルシステムは Linux システムのためのデフォルトのファイルシステムで、特定の使用しない理由がない限りこれを使用することが強く推奨されます。

ティップ
Btrfs ファイルシステムが Linux カーネル 3.2 (Debian wheezy) では利用可能です。ext4 ファイルシステムは次のデフォルトのファイルシステムとなると期待されています。

警告
カーネル空間でのライブ fsck(8) 機能やブートローダーサポートが提供されるようになるまでは、Btrfs をあなたのクリチカルなデーターに用いるべきではありません。

ティップ
一部のツールはファイルシステムへのアクセスを Linux カーネルのサポート無しでも可能にします (項9.7.2参照下さい)。

9.5.6 ファイルシステムの生成と整合性チェック

mkfs(8) コマンドは Linux システム上でファイルシステムを生成します。fsck(8) コマンドは Linux システム上でファイルシステムの整合性チェックと修理機能を提供します。

現在 Debian は、ファイルシステム形成後に定期的な `fsck` 無しがデフォルトです。

注意
一般的に `fsck` をマウントされているファイルシステムに実行することは安全ではありません。

ティップ
”/etc/mke2fs.conf” 中 に”enable_periodic_fsck” と 設 定 し、”tune2fs -c0 /dev/<partition_name>” を実行して最大マウント回数を 0 と設定すれば、再起動時に fsck(8) コマンドを root ファイルシステムを含む全ファイルシステムに安全に実行可能です。mke2fs.conf(5) と tune2fs(8) を参照ください。

ティップ
ブートスクリプトから実行される fsck(8) コマンドの結果を”/var/log/fsck/” 中のファイルからチェックします。

9.5.7 マウントオプションによるファイルシステムの最適化

”/etc/fstab” により静的なファイルシステム設定がなされます。例えば、

```
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
proc            /proc           proc    defaults        0       0
UUID=709cbe4c-80c1-56db-8ab1-dbce3146d2f7 / ext4 noatime,errors=remount-ro 0 1
UUID=817bae6b-45d2-5aca-4d2a-1267ab46ac23 none swap sw  0       0
/dev/scd0       /media/cdrom0   udf,iso9660 user,noauto 0       0
```

ティップ
UUID (項9.5.3参照下さい) は、”/dev/hda3” や”/dev/hda3” 等の通常のブロックデバイス名の代わりにブロックデバイスを指定するのに使えます。

ファイルシステムのパーフォーマンスや性格はそれに使われるマウントオプションによって最適化できます (fstab(5) と mount(8) 参照下さい)。

- ”defaults” オプションはデフォールトのオプションが次の意味です: ”rw,suid,dev,exec,auto,nouser,async”。(一般的)
- ”noatime” もしくは”relatime” オプションは読出しアクセスを高速化するのに非常に効果的です。(一般的)
- ”user” オプションは通常ユーザーがファイルシステムをマウント出来るようにします。このオプションは”noexec,nosuid,nodev” オプションの組み合わせの意味です。(一般的、CD や usb ストレージデバイスに使用)
- ”noexec,nodev,nosuid” オプションの組み合わせはセキュリティーの強化に使われます。(一般的)
- ”noauto” オプションは明示的操作のみにマウントを制限します。(一般的)
- ext3fs への”data=journal” オプションは、書込み速度を犠牲ににしますが、停電時のデーター整合性を強化します。

ティップ
ルートファイルシステムに非デフォールトのジャーナルモードを設定するには、例えば”rootflags=data=journal” 等の、カーネルブートパラメーター (項3.1.2参照下さい) を与える必要があります。lenny の場合、デフォールトのジャーナルモードは”rootflags=data=ordered” です。squeeze の場合、デフォールトのジャーナルモードは”rootflags=data=writeback” です。

9.5.8 スーパーブロックによるファイルシステムの最適化

tune2fs(8) コマンドを用いてファイルシステムのスーパーブロックによってファイルシステムを最適化できます。

- "`sudo tune2fs -l /dev/hda1`" を実行するとそのファイルシステムスーパーブロックを表示します。
- "`sudo tune2fs -c 50 /dev/hda1`" を実行するとファイルシステムのチェック (ブートアップ時の `fsck` 実行) の頻度を 50 回のブート毎に変更します。
- "`sudo tune2fs -j /dev/hda1`" の実行は ext2 から ext3 へとファイルシステム変換してファイルシステムにジャーナリングの機能を追加します。(アンマウントしたファイルシステムに対して実行します。)
- "`sudo tune2fs -O extents,uninit_bg,dir_index /dev/hda1 && fsck -pf /dev/hda1`" の実行はファイルシステムを ext3 から ext4 に変換します。(アンマウントしたファイルシステムに対して実行します。)

ティップ
tune2fs(8) は、その名前にもかかわらず、ext2 ファイルシステムに機能するだけでなく ext3 とか ext4 ファイルシステムに関しても機能します。

9.5.9 ハードディスクの最適化

警告
ハードディスクの設定はデーターの整合性にとって非常に危険な事なので、その設定をさわる前にお使いのハードウエアーをチェックし hdparam(8) のマンページをチェックします。

例えば"`/dev/hda`" に対して"`hdparm -tT /dev/hda`" とするとハードディスクのアクセス速度をテストできます。(E)IDE を使って接続された一部のハードディスクでは、"(E)IDE 32 ビット I/O サポート" を有効にし"using_dma フラグ" を有効にし"interrupt-unmask フラグ" を設定し" 複数 16 セクター I/O" を設定するように、"`hdparm -q -c3 -d1 -u1 -m16 /dev/hda`" とすると高速化できます (危険です!)。

例えば"/dev/sda" に対して"`hdparm -W /dev/sda`" とするとハードディスクの書込みキャッシュ機能をテストできます。"`hdparm -W 0 /dev/sda`" とするとハードディスクの書込みキャッシュ機能を無効にできます。

不良プレスの CDROM を現代的な高速 CD-ROM ドライブで読むには、"`setcd -x 2`" としてそれを減速して使えば読めるかもしれません。

9.5.10 ソリッドステートドライブの最適化

ソリッドステートドライブ (SSD) のパーフォーマンスやディスク消耗は以下のようにすると最適化できます。

- 最新の Linux カーネルを使用。(>=3.2)
- ディスク読み出しアクセス時のディスク書き込みの低減。
  - "`noatime`" か"`relatime`" マウントオプションを `/etc/fstab` 中で設定します。
- TRIM コマンドを有効にします。
  - "`discard`" マウントオプションを `/etc/fstab` 中で ext4 ファイルシステム、スワップ (swap) パーティション、Btrfs、他に設定します。fstab(5) 参照下さい。
  - "`discard`" オプションを `/etc/lvm/lvm.conf` 中で LVM に関して設定します。lvm.conf(5) 参照下さい。
  - "`discard`" オプションを `/etc/crypttab` 中で dm-crypt に関して設定します。crypttab(5) 参照下さい。

- SSD に最適化したディスクスペース配分手順の有効化。
  - ”ssd” マウントオプションを /etc/fstab 中で Btrfs に関して設定します。
- ラップトップ PC の場合、システムがデーターをディスクに書き出すのを 10 分間隔とします。
  - ”commit=600” マウントオプションを /etc/fstab 中で設定します。fstab(5) 参照下さい。
  - AC 操作時も laptop-mode を用いるように pm-utils を設定します。Debian BTS #659260 参照下さい。

警告
書き出しを通常の 5 秒間隔から 10 分間隔に変更すると電源喪失時にデーターが壊れやすくなります。

9.5.11 SMART を用いたハードディスクの破壊の予測

smartd(8) デーモンを使うと SMART に文句を言うハードディスクの監視と記録ができます。

1. BIOS のSMART 機能を有効にします。
2. smartmontools パッケージをインストールします。
3. df(1) を使ってリストすることであなたのハードディスクを識別します。
   - 監視対象のハードディスクを”/dev/hda” と仮定します。
4. SMART 機能が実際に有効となっているかを”smartctl -a /dev/hda” のアウトプットを使ってチェックします。
   - もし有効でない場合には、”smartctl -s on -a /dev/hda” として有効にします。
5. 次のようにして smartd(8) デーモンを実行します。
   - ”/etc/default/smartmontools” ファイル中の”start_smartd=yes” をアンコメントします。
   - ”sudo /etc/init.d/smartmontools restart” として smartd(8) デーモンを再実行します。

ティップ
smartd(8) デーモンは、警告の通知の仕方を含めて /etc/smartd.conf ファイルを用いてカスタム化できます。

9.5.12 $TMPDIR 経由で一時保存ディレクトリーを指定

通常アプリケーションは一時保存ディレクトリー”/tmp” のもとに一時ファイルを作成します。もし”/tmp” が十分なスペースを提供できない場合、行儀のいいプログラムなら $TMPDIR 変数を使ってそのような一時保存ディレクトリを指定できます。

9.5.13 LVM を使う使用可能なストレージ空間の拡張

インストール時に論理ボリュームマネージャー (LVM) (Linux 機能) 上に作られたパーティションは、大掛かりなシステムの再設定無しに複数のストレージデバイスにまたがる LVM 上のエクステントを継ぎ足したりその上のエクステントを切り捨てることで簡単にサイズ変更が出きます。

9.5.14 他パーティションをマウントする使用可能なストレージ空間の拡張

空のパーティションがあれば (例えば”/dev/sdx”)、それを mkfs.ext4(1) を使ってフォーマットし、それをあなたが空間をより必要とするディレクトリーに mount(8) することができます。(元来あったデーター内容はコピーする必要があります。)

```
$ sudo mv work-dir old-dir
$ sudo mkfs.ext4 /dev/sdx
$ sudo mount -t ext4 /dev/sdx work-dir
$ sudo cp -a old-dir/* work-dir
$ sudo rm -rf old-dir
```

ティップ
上記の代わりに、空のディスクイメージファイル (項9.6.5参照下さい) をループデバイスとしてマウントする (項9.6.3参照下さい) 事もできます。実際のディスク使用は実際にデーターを溜め込むとともに成長します。

9.5.15 他ディレクトリーをバインドマウントする使用可能なストレージ空間の拡張

使える空間がある他のパーティション中に空のディレクトリーがあれば (例えば”/path/to/emp-dir”)、そのディレクトリーを --bind” オプションを使って、空間を必要としているディレクトリー (例えば”work-dir”) にマウントすることができます。

```
$ sudo mount --bind /path/to/emp-dir work-dir
```

9.5.16 他ディレクトリーをオーバーレーマウントすることで使用可能なストレージ空間を拡張

Linux カーネル 3.18 以降 (Debian Stetch 9.0 以降) を使うと、他のパーティション中に使える空間 (例えば”/path/to/empty” と”/path/to/work”) があれば、その中にディレクトリーを作成し、容量が必要な古いディレクトリー (e.g., ”/path/to/old”) の上に OverlayFS を使って積み重ねることができます。

```
$ sudo mount -t overlay overlay \
  -olowerdir=/path/to/old-dir,upperdir=/path/to/empty,workdir=/path/to/work
```

ここで、”/path/to/old” 上に書き込むには、読み書きが許可されたパーティション上に”/path/to/empty” と”/path/to/work” があることが必要です。

9.5.17 シムリンクを使う使用可能なストレージ空間の拡張

注意
ここに書かれている事は非推奨です。ソフトウエアーによっては「ディレクトリーへのシムリンク」ではうまく機能しません。上記の「マウントする」アプローチを代わりに使ってください。

使える空間がある他のパーティション中に空のディレクトリーがあれば (例えば”/path/to/emp-dir”)、そのディレクトリーへ ln(8) を使ってシムリンクを作成することができます。

```
$ sudo mv work-dir old-dir
$ sudo mkdir -p /path/to/emp-dir
$ sudo ln -sf /path/to/emp-dir work-dir
$ sudo cp -a old-dir/* work-dir
$ sudo rm -rf old-dir
```

警告
” ディレクトリーへのシムリンク” を”/opt” のようなシステムが管理するディレクトリーに使用してはいけません。システムがアップグレードされる際にそのようなシムリンクは上書きされるかもしれません。

9.6 ディスクイメージ

次に、ディスクイメージの操作を論じます。

9.6.1 ディスクイメージの作成

例えば 2 番目の SCSI もしくはシリアル ATA ドライブ”/dev/sdb” 等の、アンマウントされたドライブのディスクイメージファイル”disk.img” は cp(1) か dd(1) を用いれば次のようにして作れます。

```
# cp /dev/sdb disk.img
# dd if=/dev/sdb of=disk.img
```

プライマリ IDE ディスクの最初のセクターにある伝統的 PC のマスターブートレコード (MBR) (項9.5.2参照下さい) のディスクイメージは、dd(1) を用いれば次のようにして作れます。

```
# dd if=/dev/hda of=mbr.img bs=512 count=1
# dd if=/dev/hda of=mbr-nopart.img bs=446 count=1
# dd if=/dev/hda of=mbr-part.img skip=446 bs=1 count=66
```

- ”mbr.img”: パーティションテーブル付きの MBR
- ”mbr-nopart.img”: パーティションテーブル抜きの MBR。
- ”mbr-part.img”: MBR のパーティションテーブルのみ。

ブートディスクとして SCSI ドライブもしくはシリアル ATA デバイスが使われる場合、”/dev/hda” を”/dev/sda” に置き換えて下さい。

オリジナルディスクのパーティションのイメージを作る場合には、”/dev/hda” を”/dev/hda1” 等で置き換えます。

9.6.2 ディスクに直接書込み

ディスクイメージファイル”disk.img” は dd(1) を使ってサイズがマッチする例えば”/dev/sdb” という 2 番目の SCSI ドライブに次のようにして書き込むことができます。

```
# dd if=disk.img of=/dev/sdb
```

同様にディスクパーティションイメージファイル”partition.img” はサイズがマッチする例えば”/dev/sdb1” という 2 番目の SCSI ドライブの 1 番目のパーティションに次のようにして書き込むことができます。

```
# dd if=partition.img of=/dev/sdb1
```

9.6.3 ディスクイメージファイルをマウント

単一パーティションイメージを含むディスクイメージ”partition.img” は次のように loop デバイスを使いマウントしアンマウントできます。

```
# losetup -v -f partition.img
Loop device is /dev/loop0
# mkdir -p /mnt/loop0
# mount -t auto /dev/loop0 /mnt/loop0
...hack...hack...hack
# umount /dev/loop0
# losetup -d /dev/loop0
```

これは以下のように簡略化出来ます。

```
# mkdir -p /mnt/loop0
# mount -t auto -o loop partition.img /mnt/loop0
...hack...hack...hack
# umount partition.img
```

複数のパーティションを含むディスクイメージ”disk.img” の各パーティションは loop デバイスを使ってマウント出来ます。loop デバイスはパーティションをデフォルトでは管理しないので、次のようにそれをリセットする必要があります。

```
# modinfo -p loop # verify kernel capability
max_part:Maximum number of partitions per loop device
max_loop:Maximum number of loop devices
# losetup -a # verify nothing using the loop device
# rmmod loop
# modprobe loop max_part=16
```

これで、loop デバイスは 16 パーティションまで管理出来ます。

```
# losetup -v -f disk.img
Loop device is /dev/loop0
# fdisk -l /dev/loop0

Disk /dev/loop0: 5368 MB, 5368709120 bytes
255 heads, 63 sectors/track, 652 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x452b6464

      Device Boot      Start         End      Blocks   Id  System
/dev/loop0p1               1         600     4819468+  83  Linux
/dev/loop0p2             601         652      417690   83  Linux
# mkdir -p /mnt/loop0p1
# mount -t ext4 /dev/loop0p1 /mnt/loop0p1
# mkdir -p /mnt/loop0p2
# mount -t ext4 /dev/loop0p2 /mnt/loop0p2
...hack...hack...hack
# umount /dev/loop0p1
# umount /dev/loop0p2
# losetup -d /dev/loop0
```

この他、同様の効果は kpartx パッケージの kpartx(8) により作られるデバイスマッパーデバイスを用いて次のようにして実現も出来ます。

```
# kpartx -a -v disk.img
...
# mkdir -p /mnt/loop0p2
# mount -t ext4 /dev/mapper/loop0p2 /mnt/loop0p2
```

```
...
...hack...hack...hack
# umount /dev/mapper/loop0p2
...
# kpartx -d /mnt/loop0
```

注意
MBR 等をスキップするオフセットを使った loop デバイスによっても、このようなディスクイメージの単一パーティションをマウント出来ます。しかしこれは失敗しがちです。

9.6.4 ディスクイメージのクリーニング

ディスクイメージファイル"disk.img" は消去済みのファイルを綺麗に無くした綺麗なスパースイメージ"new.img" に次のようにしてできます。

```
# mkdir old; mkdir new
# mount -t auto -o loop disk.img old
# dd bs=1 count=0 if=/dev/zero of=new.img seek=5G
# mount -t auto -o loop new.img new
# cd old
# cp -a --sparse=always ./ ../new/
# cd ..
# umount new.img
# umount disk.img
```

もし"disk.img" が ext2 か ext3 か ext4 の場合には、zerofree パッケージの zerofree(8) を使うことも出来ます。

```
# losetup -f -v disk.img
Loop device is /dev/loop3
# zerofree /dev/loop3
# cp --sparse=always disk.img new.img
```

9.6.5 空のディスクイメージ作成

5GiB まで成長可能な空のディスクイメージファイル"disk.img" は dd(1) と mke2fs(8) を使って次のようにして作成できます。

```
$ dd bs=1 count=0 if=/dev/zero of=disk.img seek=5G
```

loop デバイスを使ってこのディスクイメージ"disk.img" 上に ext4 ファイルシステムを作成できます。

```
# losetup -f -v disk.img
Loop device is /dev/loop1
# mkfs.ext4 /dev/loop1
...hack...hack...hack
# losetup -d /dev/loop1
$ du  --apparent-size -h disk.img
5.0G  disk.img
$ du -h disk.img
83M disk.img
```

"sparse" に関して、そのファイルサイズは 5.0GiB でその実ディスク使用はたったの 83MiB です。この相違は ext4 がスパースファイルを保持できるから可能となっています。

ティップ
スパースファイルによる実際のディスク使用はそこに書かれるデーターとともに成長します。

項9.6.3にあるように loop デバイスまたはデバイスマッパーデバイスによりデバイスに同様の操作をすることで、このディスクイメージ”disk.img” を parted(8) または fdisk(8) を使ってパーティションし mkfs.ext4(8) や mkswap(8) 等を使ってファイルシステムを作れます。

9.6.6 ISO9660 イメージファイル作成

”source_directory” のソースディレクトリーツリーから作られる ISO9660 イメージファイル”cd.iso” はcdrkit が提供する genisoimage(1) を使って次のようにして作成できます。

```
#  genisoimage -r -J -T -V volume_id -o cd.iso source_directory
```

同様に、ブート可能な ISO9660 イメージファイル”cdboot.iso” は、debian-installer のような”source_directory” にあるディレクトリーツリーから次のようにして作成できます。

```
#  genisoimage -r -o cdboot.iso -V volume_id \
   -b isolinux/isolinux.bin -c isolinux/boot.cat \
   -no-emul-boot -boot-load-size 4 -boot-info-table source_directory
```

上記では、Isolinux ブートローダー (項3.1.2参照下さい) がブートに使われています。

次のようにすると CD-ROM デバイスから直接 md5sum 値を計算し ISO9660 イメージを作成できます。

```
$ isoinfo -d -i /dev/cdrom
CD-ROM is in ISO 9660 format
...
Logical block size is: 2048
Volume size is: 23150592
...
# dd if=/dev/cdrom bs=2048 count=23150592 conv=notrunc,noerror | md5sum
# dd if=/dev/cdrom bs=2048 count=23150592 conv=notrunc,noerror > cd.iso
```

警告
正しい結果を得るために上記のように Linux の ISO9660 ファイルシステム先読みバグを注意深く避けなければいけません。

9.6.7 CD/DVD-R/RW に直接書込み

ティップ
DVD は、cdrkit が提供する wodim(1) にとっては単に大きな CD です。

使えるデバイスは次のようにするとみつかります。

```
# wodim --devices
```

そしてブランクの CD-R をドライブに挿入して、例えば”/dev/hda” というこのデバイスに ISO9660 イメージファイル”cd.iso” に wodim(1) を使って次のようにして書込みます。

```
# wodim -v -eject dev=/dev/hda cd.iso
```

もし CD-R ではなく CD-RW が使われている場合には、次を代わりに実行して下さい。

```
# wodim -v -eject blank=fast dev=/dev/hda cd.iso
```

ティップ
もしあなたのデスクトップシステムが CD を自動的にマウントする場合、wodim(1) を使う前に”sudo unmount /dev/hda” として CD をアンマウントします。

9.6.8 ISO9660 イメージファイルをマウント

もし”cd.iso” の内容が ISO9660 イメージの場合、次のようにするとそれを”/cdrom” に手動でマウントできます。

```
# mount -t iso9660 -o ro,loop cd.iso /cdrom
```

ティップ
現代的なデスクトップシステムでは ISO9660 フォーマットされた CD のようなリムーバブルメディアを自動的にマウントします (項10.1.7参照下さい)。

9.7 バイナリーデーター

次に、ストレージメディア上のバイナリーデーターを直接操作することを論じます。

9.7.1 バイナリーデーターの閲覧と編集

もっとも基本的なバイナリーファイルを閲覧方法は”od -t x1” コマンドを使うことです。

パッケージ	ポプコン	サイズ	説明
coreutils	V:891, I:999	17478	ファイルをダンプする od(1) がある基本パッケージ (HEX, ASCII, OCTAL, …)
bsdmainutils	V:60, I:996	26	ファイルをダンプする hd(1) があるユーティリティーパッケージ (HEX, ASCII, OCTAL, …)
hexedit	V:1, I:12	72	バイナリーエディターとビューワー (HEX, ASCII)
bless	V:0, I:4	1028	フル機能の 16 進エディター (GNOME)
okteta	V:1, I:15	1508	フル機能の 16 進エディター (KDE4)
ncurses-hexedit	V:0, I:2	132	バイナリーエディターとビューワー (HEX, ASCII, EBCDIC)
beav	V:0, I:0	133	バイナリーエディターとビューワー (HEX, ASCII, EBCDIC, OCTAL, …)

Table 9.20: バイナリーデーターを閲覧や編集するパッケージのリスト

ティップ
HEX は底が 16 の16 進フォーマットです。OCTAL は底が 8 の8 進フォーマットです。ASCII (アスキー) は情報交換用アメリカ標準コードで、通常の英文テキストです。EBCDIC (エビシディック) は IBM メインフレームオペレーティングシステム上で使われる拡張二進化十進数互換コードです。

9.7.2 ディスクをマウントせずに操作

ディスクをマウントせずに読出しや書込みをするツールがあります。

パッケージ	ポプコン	サイズ	説明
mtools	V:10, I:83	389	MSDOS ファイルをマウントせずに使うツール
hfsutils	V:0, I:7	1884	HFS や HFS+ ファイルをマウントせずに使うツール

Table 9.21: ディスクをマウントせずに操作するパッケージのリスト

9.7.3 データーの冗長性

Linux カーネルが提供するソフトウエアー RAID システムは高いレベルのストレージ信頼性を達成するためにカーネルのファイルシステムのレベルでデーターの冗長性を提供します。

アプリケーションプログラムレベルでストレージの高い信頼性を達成するようにデーター冗長性を付加するツールもあります。

パッケージ	ポプコン	サイズ	説明
par2	V:4, I:15	271	ファイルのチェックと修理のためのパリティーアーカイブセット
dvdisaster	V:0, I:2	1741	CD/DVD メディアのデーターロス/傷つき/老化の防止
dvbackup	V:0, I:0	413	MiniDV カメラレコーダを使うバックアップツール (rsbep(1) を提供)
vdmfec	V:0, I:0	97	前方エラー修正を使って失われたブロックの復元

Table 9.22: ファイルにデーターの冗長性を追加するツールのリスト

9.7.4 データーファイルの復元と事故の証拠解析

データーファイルの復元と事故の証拠解析のツールがあります。

ティップ
e2fsprogs パッケージ中の debugfs(8) の `list_deleted_inodes` または `undel` コマンドを用いると ext2 ファイルシステム上でファイルのアンデリートができます。

9.7.5 大きなファイルを小さなファイルに分割

単一ファイルでバックアップするにはデーターが大きすぎる場合、そのファイル内容を例えば 2000MiB の断片にしてバックアップし、それらの断片を後日マージしてオリジナルのファイルに戻せます。

```
$ split -b 2000m large_file
$ cat x* >large_file
```

注意
名前がかち合わないように”x” で始まるファイル名のファイルが無いようにします。

パッケージ	ポプコン	サイズ	説明
testdisk	V:3, I:38	1426	パーティションのスキャンとディスク復元のためのユーティリティー
magicrescue	V:0, I:3	259	マジックバイトを探してファイルを復元するユーティリティー
scalpel	V:0, I:4	87	質素で高性能なファイル彫刻刀
myrescue	V:0, I:3	83	破壊したハードディスクからデーターを救出
extundelete	V:1, I:11	148	ext3/4 ファイルシステム上のファイルの削除復元ユーティリティー
ext4magic	V:0, I:4	233	ext3/4 ファイルシステム上のファイルの削除復元ユーティリティー
ext3grep	V:0, I:3	281	ext3 ファイルシステム上のファイルの削除復元ヘルプツール
scrounge-ntfs	V:0, I:3	50	NTFS ファイルシステム上のデーター復元プログラム
gzrt	V:0, I:0	33	gzip 復元ツールキット
sleuthkit	V:2, I:24	1511	証拠解析のためのツール (Sleuthkit)
autopsy	V:0, I:2	1027	SleuthKit のための GUI
foremost	V:0, I:7	104	データー復元のための証拠解析アプリケーション
guymager	V:0, I:1	1030	Qt 使用の証拠解析用イメージ作成ソフト
dcfldd	V:0, I:5	106	証拠解析とセキュリティーのための dd の強化版

Table 9.23: データーファイルの復元と事故の証拠解析のリスト

9.7.6 ファイル内容の消去

ログファイルのようなファイルの内容を消去するためには、rm(1) を使ってファイルを消去しその後新しい空ファイルを作成することは止めましょう。コマンド実行間にファイルがアクセスされているかもしれないのがこの理由です。次のようにするのがファイル内容を消去する安全な方法です。

```
$ :>file_to_be_cleared
```

9.7.7 ダミーファイル

次のコマンドはダミーや空のファイルを作成します。

```
$ dd if=/dev/zero    of=5kb.file bs=1k count=5
$ dd if=/dev/urandom of=7mb.file bs=1M count=7
$ touch zero.file
$ : > alwayszero.file
```

次のファイルを見つかります。

- "`5kb.file`" は 5KB のゼロの連続です。
- "`7mb.file`" は 7MB のランダムなデーターです。
- "`zero.file`" は 0 バイト長のファイルかもしれません。もしファイルが存在する時は、その `mtime` を更新しその内容と長さを保持します。
- "`alwayszero.file`" は常に 0 バイト長ファイルです。もしファイルが存在する時は `mtime` を更新しファイル内容をリセットします。

9.7.8 ハードディスクの全消去

"`/dev/sda`" にある USB メモリースティック等のハードディスク類似デバイス全体のデーターを完全に消すいくつかの方法があります。

注意
次のコマンドを実行する前にまず USB メモリースティックの場所を mount(8) を使ってチェックします。”/dev/sda” によって指し示されるデバイスは SCSI ハードディスクかも知れませんしあなたの全システムのあるシリアル ATA ハードディスクかも知れません。

次のようにしてデーターを 0 にリセットして全消去します。

```
# dd if=/dev/zero of=/dev/sda
```

次のようにしてランダムデーターを上書きして全消去します。

```
# dd if=/dev/urandom of=/dev/sda
```

次のようにしてランダムデーターを非常に効率的に上書きして全消去します。

```
# shred -v -n 1 /dev/sda
```

Debian インストーラ CD 等の多くのブート可能な Linux の CD のシェルから dd(1) が利用可能ですから、”/dev/hda” や”/dev/sda” 等のシステムハードディスクに対して同類のメディアから消去コマンドを実行することでインストールされたシステムを完全に消去することができます。

9.7.9 ハードディスク未使用部分の全消去

データーの消去はファイルシステムからアンリンクされているだけなので、例えば”/dev/sdb1” のようなハードディスク (USB メモリースティック) 上の使用されていない領域には消去されたデーター自身が含まれているかもしれません。これらに上書きすることで綺麗に消去できます。

```
# mount -t auto /dev/sdb1 /mnt/foo
# cd /mnt/foo
# dd if=/dev/zero of=junk
dd: writing to ‘junk’: No space left on device
...
# sync
# umount /dev/sdb1
```

警告
あなたの USB メモリースティックではこれで普通十分です。でもこれは完璧ではありません。消去されたファイル名や属性はファイルシステム中に隠れて残っているかもしれません。

9.7.10 削除されたがまだオープン中のファイルの復活法

ファイルをうっかり消去しても、そのファイルが何らかのアプリケーション (読出しか書込み) によって使われている限り、そのようなファイルを回復出来ます。

例えば、次を試してみて下さい:

```
$ echo foo > bar
$ less bar
$ ps aux | grep ’ less[ ]’
bozo    4775  0.0  0.0  92200   884 pts/8    S+   00:18   0:00 less bar
$ rm bar
$ ls -l /proc/4775/fd | grep bar
lr-x------ 1 bozo bozo 64 2008-05-09 00:19 4 -> /home/bozo/bar (deleted)
```

```
$ cat /proc/4775/fd/4 >bar
$ ls -l
-rw-r--r-- 1 bozo bozo 4 2008-05-09 00:25 bar
$ cat bar
foo
```

この代わりに、(lsof パッケージがインストールされている時) もう一つのターミナルで次のように実行します。

```
$ ls -li bar
2228329 -rw-r--r-- 1 bozo bozo 4 2008-05-11 11:02 bar
$ lsof |grep bar|grep less
less 4775 bozo 4r REG 8,3 4 2228329 /home/bozo/bar
$ rm bar
$ lsof |grep bar|grep less
less 4775 bozo 4r REG 8,3 4 2228329 /home/bozo/bar (deleted)
$ cat /proc/4775/fd/4 >bar
$ ls -li bar
2228302 -rw-r--r-- 1 bozo bozo 4 2008-05-11 11:05 bar
$ cat bar
foo
```

9.7.11 全てのハードリンクを検索

ハードリンクのあるファイルは”ls -li” を使って確認できます、

```
$ ls -li
total 0
2738405 -rw-r--r-- 1 root root 0 2008-09-15 20:21 bar
2738404 -rw-r--r-- 2 root root 0 2008-09-15 20:21 baz
2738404 -rw-r--r-- 2 root root 0 2008-09-15 20:21 foo
```

”baz” も”foo” もリンク数が”2” (>1) でハードリンクがある事を示しています。これらの inode 番号は共通の”2738404” です。これはこれらがハードリンクされた同じファイルということを意味します。ハードリンクされた全てのファイルを偶然うまく見つけられない場合は、それを例えば”2738404” という inode で次のようにして探せます。

```
# find /path/to/mount/point -xdev -inum 2738404
```

9.7.12 見えないディスクスペースの消費

削除されたがオープンされたままのファイルは、通常の du(1) では見えませんが、ディスクスペースを消費します。これらは次のようにすればそのサイズとともにリストできます。

```
# lsof -s -X / |grep deleted
```

9.8 データー暗号化ティップ

あなたの PC への物理的アクセスがあると、誰でも簡単に root 特権を獲得できあなたの PC の全てのファイルにアクセスできます (項4.7.4参照下さい)。これが意味するところは、あなたの PC が盗まれた場合にログインのパスワードではあなたのプライベートでセンシティブなデーターを守れないと言うことです。それを達成するにはデーターの暗号化技術を適用しなければいけません。GNU プライバシーガード (項10.3参照下さい) はファイルを暗号化できますが、少々手間がかかります。

dm-crypt と eCryptfs は最小限のユーザー努力でできる自動的なデーターの暗号化を Linux のカーネルモジュールその物を使って提供します。

パッケージ	ポプコン	サイズ	説明
cryptsetup	V:29, I:78	402	暗号化されたブロックデバイス (dm-crypt / LUKS) のためのユーティリティー
cryptmount	V:4, I:5	228	ノーマルユーザーによるマウント/アンマウントに焦点を当てた暗号化されたブロックデバイス (dm-crypt / LUKS) のためのユーティリティー
ecryptfs-utils	V:3, I:5	460	暗号化されたスタックドファイルシステム (eCryptfs) のためのユーティリティー

Table 9.24: データー暗号化ユーティリティーのリスト

Dm-crypt は device-mapper を使う暗号学的ファイルシステムです。Device-mapper は 1 つのブロックデバイスをもう 1 つのブロックデバイスにマップします。

eCryptfs はスタックドファイルシステムを使うもう 1 つのファイルシステムです。スタックドファイルシステムはマウントされたファイルシステム上の既存のディレクトリーの上に重ね合わせます。

注意
データーの暗号化には CPU 時間等の負担がかかります。その利益と負担の両天秤をします。

注意
debian-installer (lenny 以降) を使うと、dm-crypt/LUKS と initramfs を使って、全 Debian システムを暗号化したディスク上にインストールできます。

ティップ
ユーザー空間での暗号化ユーティリティーに関しては項10.3を参照下さい: GNU プライバシーガード。

9.8.1 dm-crypt/LUKS を使ったリムーバブルディスクの暗号化

例えば”/dev/sdx” にある USB メモリースティックのようなリムーバブルストレージデバイスの内容を dm-crypt/LUKS を使って暗号化できます。それを単に次のようにフォーマットします。

```
# badblocks -c 1024 -s -w -t random -v /dev/sdx
# fdisk /dev/sdx
... ”n” ”p” ”1” ”return” ”return” ”w”
# cryptsetup luksFormat /dev/sdx1
...
# cryptsetup open --type luks /dev/sdx1 sdx1
...
# ls -l /dev/mapper/
total 0
crw-rw---- 1 root root  10, 60 2008-10-04 18:44 control
brw-rw---- 1 root disk 254,  0 2008-10-04 23:55 sdx1
# mkfs.vfat /dev/mapper/sdx1
...
# cryptsetup luksClose sdx1
```

こうすると暗号化されたディスクは、現代的な GNOME のようなデスクトップ環境では gnome-mount(1) を使ってパスワードを聞く以外は通常のディスクと同様に”/media/<disk_label>” にマウントできます (項10.1.7参照下さい)。全て書込まれるデーターが暗号化されている点が相違点です。上記と違い、例えば”mkfs.ext4 /dev/sdx1” を使って ext4 等の異なったフォーマットで記録メディアをフォーマットしても良いです。

注意
もしデーターのセキュリティーが本当に偏執狂的に気になるなら、上記例で複数回の重ね書きをする必要があるかもしれません。でもこの操作は非常に時間がかかります。

9.8.2 dm-crypt を使って swap パーティションを暗号化

オリジナルの”/etc/fstab” が次の内容と仮定します。

```
/dev/sda7 swap sw 0 0
```

dm-crypt を使って swap パーティションの暗号化を次のようにして有効にします

```
# aptitude install cryptsetup
# swapoff -a
# echo ”cswap /dev/sda7 /dev/urandom swap” >> /etc/crypttab
# perl -i -p -e ”s/\/dev\/sda7/\/dev\/mapper\/cswap/” /etc/fstab
# /etc/init.d/cryptdisks restart
 ...
# swapon -a
```

9.8.3 dm-crypt/LUKS で暗号化されたディスクのマウント

dm-crypt/LUKS を用いて”/dev/sdc5” 上に作成された暗号化されたディスクパーティションは以下のようにして”/mnt” マウントできます:

```
$ sudo cryptsetup open /dev/sdc5 ninja --type luks
Enter passphrase for /dev/sdc5: ****
$ sudo lvm
lvm> lvscan
  inactive          ’/dev/ninja-vg/root’ [13.52 GiB] inherit
  inactive          ’/dev/ninja-vg/swap_1’ [640.00 MiB] inherit
  ACTIVE            ’/dev/goofy/root’ [180.00 GiB] inherit
  ACTIVE            ’/dev/goofy/swap’ [9.70 GiB] inherit
lvm> lvchange -a y /dev/ninja-vg/root
lvm> exit
  Exiting.
$ sudo mount /dev/ninja-vg/root /mnt
```

9.8.4 eCryptfs を使って自動的にファイルを暗号化

eCryptfs と ecryptfs-utils パッケージを使うことで、”~/Private/” の下に書かれるファイルを自動的に暗号化できます。

- ecryptfs-setup-private(1) を実行してプロンプトに従って”~/Private/” を設定します。
- ecryptfs-mount-private(1) を実行して”~/Private/” を有効にします。
- センシティブなデーターファイルを”~/Private/” に移動し、必要に応じてシムリンクを作成します。
  - 候補: ”~/.fetchmailrc”、”~/.ssh/identity”、”~/.ssh/id_rsa”、”~/.ssh/id_dsa”、”go-rwx” を持つ他のファイル
- センシティブなデーターディレクトリーを”~/Private/” 中のサブディレクトリーに移動し、必要に応じてシムリンクを作成します。

  – 候補: ”~/.gnupg”、”go-rwx” を持つ他のディレクトリー

- デスクトップ操作がしやすいように”~/Desktop/Private/” から”~/Private/” までのシムリンクを作成します。
- ecryptfs-umount-private(1) を実行して”~/Private/” を無効にします。
- 暗号化されたデーターが必要な際に”ecryptfs-mount-private” を実行して”~/Private/” を有効にします。

ティップ
eCryptfs はセンシティブなファイルのみを選択的に暗号化するので、そのシステムへの負担は dm-crypt を全ルートとか”/home” デバイスとかに使うよりはるかに少ないです。eCryptfs は特段のディスク上のストレージの割り当て努力の必要はありませんが、全てのファイルシステムメタデーターを秘匿することはできません。

9.8.5 eCryptfs を自動的にマウント

もしあなたのログインパスワードを暗号化キーを包むのに使っている場合には、”/etc/pam.d/common-auth” 中の”pam_permit.so” のすぐ前に次に記す行があるようにすることで PAM (プラグ可能な認証モジュール: Pluggable Authentication Modules) を使って eCryptfs を自動的にマウントできます。

”/etc/pam.d/common-auth” の”pam_permit.so” の直前に次の行を挿入します。

```
auth required pam_ecryptfs.so unwrap
```

”/etc/pam.d/common-session” の最後に次の行を挿入します。

```
session optional pam_ecryptfs.so unwrap
```

”/etc/pam.d/common-password” の最初の有効行に次の行を挿入します。

```
password required pam_ecryptfs.so
```

これは非常に便利です。

警告
PAM の設定エラーをすると、あなた自身のシステムからあなたを締め出すかもしれません。第4章を参照下さい。

注意
もしあなたのログインパスワードを暗号化キーを包むのに使っている場合には、あなたの暗号化されたデーターのセキュリティーはあなたのユーザーログインパスワードと同程度です (項4.3参照下さい)。注意深く強力なパスワードを設定していないと、あなたのラップトップを誰かが盗んだ後にパスワード破りソフトを実行すれば、あなたのデーターは危険にさらされます (項4.7.4参照下さい)。

9.9 カーネル

Debian はモジュール化された Linux カーネルをサポートされるアーキテクチャに対してパッケージとしてディストリブートしています。

9.9.1 Linux カーネル 2.6/3.x

Linux カーネル 2.4 と比較して特記すべき Linux カーネル 2.6/3.x の機能がいくつかあります。

- デバイスは udev システムで生成されます (項3.3参照下さい)。
- IDE CD/DVD デバイスへの読出し/書込みアクセスは `ide-scsi` モジュールを使用しません。
- ネットワークのパケットフィルター機能は `iptable` カーネルモジュールを使います。

Linux 2.6.39 から Linux 3.0 へとバージョンが飛んだのは、大きな技術的な変更があったからでは無く 20 周年記念が理由です。

9.9.2 カーネル変数

多くの Linux の機能はカーネル変数を使い次のように設定されます。

- ブートローダーにより初期化されたカーネル変数 (項3.1.2参照下さい)
- 実行時に sysfs によりアクセスできるカーネル変数に関して sysctl(8) を用い変更されたカーネル変数 (項1.2.12参照下さい)
- モジュールが起動された際の modprobe(8) の引数により設定されるモジュール変数 (項9.6.3参照下さい)

`linux-doc-3.*` パッケージで供給される Linux カーネル文書 (”`/usr/share/doc/linux-doc-3.*/Documentation/`” 中の”`kernel-parameters.txt(.gz)`” や関連する他の文書を参照下さい。

9.9.3 カーネルヘッダー

ほとんどの普通のプログラムはカーネルヘッダーを必要としませんし、コンパイルするのにそれらを直接用いるとコンパイルがうまくいかないかもしれません。普通のプログラムは Debian システム上では (`glibc` ソースパッケージから生成される) `libc6-dev` パッケージが提供する”`/usr/include/linux`” や”`/usr/include/asm`” 中のヘッダを使ってコンパイルするべきです。

注意
外部ソースからのカーネルモジュールやオートマウンターデーモン (amd) のようなカーネル固有の一部プログラムをコンパイルする場合、例えば”`-I/usr/src/linux-particular-version/include/`” 等の対応するカーネルヘッダーへのパスをコマンドラインで指定しなければいけません。module-assistant(8) (またはその短縮形 `m-a`) はユーザーが 1 つ以上のカスタムカーネルに関するモジュールパッケージを簡単にビルドとインストールすることを援助します。

9.9.4 カーネルと関連モジュールのコンパイル

Debian にはカーネルと関連モジュールをコンパイルする独自の方法があります。

項3.1.2 中で `initrd` を使う場合、initramfs-tools(8) と update-initramfs(8) と mkinitramfs(8) と initramfs.conf(5) 中の関連情報をしっかり読んで下さい。

警告
Linux カーネルソースをコンパイルする時にソースツリー中のディレクトリー (例えば”`/usr/src/linux*`”) から”`/usr/include/linux`” や”`/usr/include/asm`” へのシムリンクを張ってはいけません。(古くなった一部文書はまだこれをすることを提案しています。)

パッケージ	ポプコン	サイズ	説明
build-essential	I:499	20	Debian パッケージをビルドする上で不可欠なパッケージ: make、gcc、…
bzip2	V:157, I:970	122	bz2 ファイルのための圧縮と解凍ユーティリティー
libncurses5-dev	I:116	6	ncurses のためのデベロッパ用ライブラリーと文書
git	V:305, I:478	35040	git: Linux カーネルによって使われている分散型リビジョンコントロールシステム
fakeroot	V:35, I:521	228	パッケージを非 root としてビルドするための fakeroot 環境を提供
initramfs-tools	V:371, I:989	112	initramfs をビルドするツール (Debian 固有)
dkms	V:70, I:219	294	動的カーネルモジュールサポート (DKMS) (汎用)
devscripts	V:9, I:57	2623	Debian パッケージメンテナ用ヘルパースクリプト (Debian 固有)

Table 9.25: Debian システム上でカーネルの再コンパイルためにインストールする重要パッケージのリスト

注意
Debian の stable (安定版) システム上で最新の Linux カーネルをコンパイルする際には、Debian の unstable (非安定版) システムからバックポートされた最新のツールが必要かもしれません。

注意
動的カーネルモジュールサポート (DKMS) は、カーネル全体を変えること無く個別カーネルモジュールをアップグレードできるようにする新しいディストリビューションに依存しない枠組みです。これはアウトオブツリーのモジュールの管理方法です。これはあなたがカーネルをアップグレードする際のモジュールの再構築を簡単にもします。

9.9.5 カーネルソースのコンパイル: Debian カーネルチーム推奨

アップストリームのカーネルソースからカーネルバイナリーパッケージを作成するには、それが提供するターゲットを用いて”deb-pkg” とします。

```
$ sudo apt-get build-dep linux
$ cd /usr/src
$ wget http://www.kernel.org/pub/linux/kernel/v3.11/linux-<version>.tar.bz2
$ tar -xjvf linux-<version>.tar.bz2
$ cd linux-<version>
$ cp /boot/config-<version> .config
$ make menuconfig
 ...
$ make deb-pkg
```

ティップ
linux-source-<version> パッケージは Debian パッチがあたった Linux カーネルソースを”/usr/src/linux-<version>.tar.bz2” として提供します。

Debian カーネルソースパッケージから特定のバイナリパッケージをビルドするには、”debian/rules.gen” 中の”binary-arch_<architecture>_<featureset>_<flavour>” ターゲットを使います。

```
$ sudo apt-get build-dep linux
$ apt-get source linux
$ cd linux-3.*
$ fakeroot make -f debian/rules.gen binary-arch_i386_none_686
```

詳細は以下参照下さい:

- Debian Wiki: KernelFAQ
- Debian Wiki: Debian カーネル
- Debian Linux カーネルハンドブック: https://kernel-handbook.debian.net

9.9.6 ハードウエアードライバーとファームウエアー

ハードウエアードラーバーとはターゲットシステム上で実行されるコードです。ほとんどのハードウエアードライバーは現在フリーソフトウエアーとして入手可能で通常の `main` エリアにある Debian カーネルパッケージに含まれています。

- GPU ドラーバー
  - Intel GPU ドラーバー (`main`)
  - AMD/ATI GPU ドラーバー (`main`)
  - NVIDIA GPU ドライバー (nouveau ドライバーは `main`、ベンダーにサポートされたバイナリーのみ提供のドライバーは `non-free` 。)
- ソフトモデムドライバー
  - martian-modem や sl-modem-dkms パーケージ (`non-free`)

ファームウエアーとはデバイスにロードされるコードやデーター (例えば CPU マイクロコードや、GPU 上で実行されるレンダリングコードや、FPGA / CPLD データー等々) です。一部のファームウエアーパッケージはフリーソフトウエアーとして入手可能ですが、多くのファームウエアーパッケージはソースの無いバイナリーデーターを含むためにフリーソフトウエアーとして入手不可能です。

- firmware-linux-free (`main`)
- firmware-linux-nonfree (`non-free`)
- firmware-linux-* (`non-free`)
- *-firmware (`non-free`)
- intel-microcode (`non-free`)
- amd64-microcode (`non-free`)

`non-free` や `contrib` パッケージは Debian システムの一部でないことに注意して下さい。`non-free` や `contrib` エリアへのアクセスの有効化や無効化は項2.1.4 に説明されています。項2.1.5 に記載されているように `non-free` や `contrib` パッケージを使用に付帯するマイナスを認識すべきです。

9.10 仮想化システム

仮想化されたシステムを利用すると単一ハード上で同時に複数のシステムのインスタンスを実行することが加能となります。

ティップ
http://wiki.debian.org/SystemVirtualization を参照下さい。

9.10.1 仮想化ツール

Debian には、単純な chroot ではない仮想化やエミュレーション関連のパッケージがあります。一部のパッケージはあなたがそのような環境をセットアップする事を援助します。

パッケージ	ポプコン	サイズ	説明
schroot	V:7, I:10	2708	Debian バイナリーパッケージを chroot 中で実行する専用ツール
sbuild	V:1, I:4	286	Debian ソースから Debian バイナリーパッケージをビルドするツール
pbuilder	V:2, I:16	966	Debian パッケージの個人的なパッケージビルドソフト
debootstrap	V:6, I:63	298	基本的な Debian システムのブートストラップ (sh で書かれている)
cdebootstrap	V:0, I:3	116	Debian システムのブートストラップ (C で書かれている)
virt-manager	V:10, I:42	2298	仮想マシンマネージャー: 仮想マシンを管理するデスクトップアプリケーション
libvirt-clients	V:43, I:62	1167	libvirt ライブラリー用のプログラム
bochs	V:0, I:1	7194	Bochs: IA-32 PC エミュレーター
qemu	I:34	94	QEMU: 高速で汎用のプロセッサエミュレーター
qemu-system	I:21	95	QEMU: フルシステムエミュレーションのバイナリ
qemu-user	V:0, I:13	89671	QEMU: ユーザーモードエミュレーションのバイナリ
qemu-utils	V:11, I:107	6083	QEMU: ユーティリティー
qemu-kvm	V:10, I:61	107	KVM: ハードウエア補助仮想化を利用する x86 ハードウエア上のフル仮想化
virtualbox	V:12, I:16	106495	VirtualBox: i386 と amd64 上での x86 仮想化解決策
xen-tools	V:0, I:4	727	Debian XEN 仮想サーバーの管理ツール
wine	V:19, I:82	192	Wine: Windows API の実装 (標準スイート)
dosbox	V:2, I:18	2742	DOSBox: Tandy/Herc/CGA/EGA/VGA/SVGA グラフィクス、サウンド、DOS 付きの x86 エミュレーター
dosemu	V:0, I:2	4891	DOSEMU: Linux 用 DOS エミュレーター
vzctl	V:0, I:1	1112	OpenVZ サーバー仮想化策 - コントロールツール
vzquota	V:0, I:1	236	OpenVZ サーバー仮想化策 - クオータツール
lxc	V:10, I:15	18761	Linux コンテナ - ユーザースペースツール

Table 9.26: 仮想化ツールのリスト

異なるプラットフォーム仮想化策の詳細な比較は Wikipedia の記事 Comparison of platform virtual machines を参照下さい。

9.10.2 仮想化の業務フロー

注意
ここに記載された機能の一部は squeeze でのみ利用可能です。

注意
lenny 以来の Debian のデフォルトカーネルは KVM をサポートしています

仮想化のための典型的な業務フローにはいくつかの段階があります。

- 空のファイルシステムの作成 (ファイルツリーもしくはディスクイメージ)。

  - ファイルツリーは”`mkdir -p /path/to/chroot`” として作成できる。
  - raw ディスクイメージファイルは dd(1) を使って作れます (項9.6.1と項9.6.5参照下さい)。
  - qemu-img(1) はQEMU によりサポートされたディスクイメージの作成や変換に使えます。
  - raw と VMDK ファイルフォーマットは仮想ツール間の共通フォーマットとして使えます。
- mount(8) を使ってディスクイメージをファイルシステムにマウントする (任意)。
  - raw のディスクイメージファイルに関しては、loop デバイスまたはデバイスマッパーデバイス (項9.6.3参照下さい) としてマウント。
  - QEMU がサポートするディスクイメージファイルに関しては、ネットワークブロックデバイス (項9.10.3参照下さい) としてマウント。
- 必要なシステムデーターを用いて対象のファイルシステムを充足。
  - `debootstrap` や `cdebootstrap` のようなプログラムがこのプロセスを援助します (項9.10.4参照下さい)。
  - OS のインストーラーをフルシステムエミュレーション下で利用。
- 仮想化環境下でプログラムを実行。
  - chroot は、仮想環境の中でプログラムのコンパイルやコンソールアプリケーションの実行やデーモンの実行等をするのに十分な基本的仮想環境を提供します。
  - QEMU: クロスプラットフォームの CPU エミュレーションを提供
  - KVM と共の QEMU はハードウエア補助仮想化によるフルシステムエミュレーションを提供します。
  - VirtualBox はハードウエア補助仮想化の有無によらず i386 と amd64 上でのフルシステムエミュレーションを提供します。

9.10.3 仮想ディスクイメージファイルをマウント。

raw ディスクイメージファイルに関しては、項9.6を参照下さい。

他の仮想ディスクイメージに関しては、qemu-nbd(1) を使ってネットワークブロックデバイスプロトコルを用いてそれらをエクスポートし `nbd` カーネルモジュールを使ってそれらをマウントできます。

qemu-nbd(1) はQEMU がサポートする次のディスクフォーマットをサポートします: raw、qcow2、qcow、vmdk、vdi、bochs、cow (user-mode Linux の copy-on-write)、parallels、dmg、cloop、vpc、vvfat (virtual VFAT)、host_device。

ネットワークブロックデバイスはloop デバイスと同様の方法でパーティションをサポートします (項9.6.3参照下さい)。”`image.img`” の最初のパーティションは次のようにするとマウントできます。

```
# modprobe nbd max_part=16
# qemu-nbd -v -c /dev/nbd0 disk.img
...
# mkdir /mnt/part1
# mount /dev/nbd0p1 /mnt/part1
```

ティップ
qemu-nbd(8) に”`-P 1`” オプションを用いると、”`disk.img`” の最初のパーティションだけをエクスポートできます。

9.10.4 Chroot システム

chroot(8) を使うのは、GNU/Linux 環境の異なったインスタンスをリブートすることなく単一システム上で同時に実行する最も基本的な手法です。

注意
次の例は親システムと chroot システムが同じ CPU アーキテクチャを共有していると仮定しています。

pbuilder(8) プログラムを script(1) の下で次のように実行すると chroot(8) の設定と使い方が学べます。

```
$ sudo mkdir /sid-root
$ sudo pbuilder --create --no-targz --debug --buildplace /sid-root
```

”sid-root” の下に sid 環境のためのシステムデーターをどのようにして充足するかは debootstrap(8) か cdebootstrap(1) を見ると分かります。

ティップ
この debootstrap(8) と cdebootstrap(1) は、Debian インストーラーが Debian をインストールするのに使われています。これらは Debian のインストールディスクを使わず他の GNU/Linux ディストリビューションから Debian をインストールするのにも使えます。

```
$ sudo pbuilder --login --no-targz  --debug --buildplace /sid-root
```

どのようにして sid 環境下で実行されるシステムシェルが作られるかが次で観察できます。

1. ローカル設定のコピー (”/etc/hosts” と ”/etc/hostname” と ”/etc/resolv.conf”)
2. ”/proc” ファイルシステムのマウント
3. ”/dev/pts” ファイルシステムのマウント
4. 常に 101 でプログラム終了する”/usr/sbin/policy-rc.d” を作成
5. ”chroot /sid-root bin/bash -c ’exec -a -bash bin/bash’” を実行

注意
プログラムによっては機能するために chroot の下で pbuilder が提供するより多くの親システムのファイルへのアクセスする必要があります。例えば、”/sys” や”/etc/passwd” や”/etc/group” や”/var/run/utmp” や”/var/log/wtmp” 等が bind マウントもしくはコピーされる必要があるかもしれません。

注意
”/usr/sbin/policy-rc.d” ファイルは、Debian システム上でデーモンプログラムが自動的に起動されることを防ぎます。”/usr/share/doc/sysv-rc/README.policy-rc.d.gz” を参照下さい。

ティップ
pbuilder という特化した chroot パッケージの本来の目的は、chroot システムを作りその chroot 中でパッケージをビルドすることです。それはパッケージのビルド依存関係が正しいことをチェックし、不必要で間違ったビルド依存関係が出来上がったパッケージに混入しないようにする理想的なシステムです。

ティップ
類似の schroot パッケージは i386 の chroot システムを amd64 の親システムの下で実行方法を教えてくれます。

9.10.5 複数のデスクトップシステム

仮想化を使って複数のデスクトップシステムを安全に実行するには、Debian 安定版 (stable) システム上で QEMU か VirtualBox を使うことをお勧めします。これらを使うと通常ありがちなリスクに晒されずに Debian テスト版 (testing) や不安定版 (unstable) システムのデスクトップアプリケーションを実行できるようになります。

純粋な QEMU は非常に遅いので、ホストシステムがサポートする際には KVM を使って加速することをお勧めします。

QEMU 用の Debian システムを含む仮想ディスクイメージ”virtdisk.qcow2” はdebian-installer: 小さな CD を使って次のように作成できます。

```
$ wget http://cdimage.debian.org/debian-cd/5.0.3/amd64/iso-cd/debian-503-amd64-netinst.iso
$ qemu-img create -f qcow2 virtdisk.qcow2 5G
$ qemu -hda virtdisk.qcow2 -cdrom debian-503-amd64-netinst.iso -boot d -m 256
...
```

更なるティップに関しては Debian wiki: QEMU を参照下さい。

VirtualBox はQt の GUI ツールとして提供され非常に直感的に理解できます。その GUI とコマンドラインツールは VirtualBox User Manual と VirtualBox User Manual (PDF) で説明されています。

ティップ
Ubuntu や Fedra 等の GNU/Linux ディストリビューションを仮想化の下で実行するのは設定ティップを学ぶ非常に良い方法です。他のプロプライエタリな OS もこの GNU/Linux の仮想化の下で上手く実行できます。

Chapter 10

データー管理

バイナリーとテキストのデーターを Debian システム上で管理するツールとティップを記します。

10.1 共有とコピーとアーカイブ

警告
競合状態とならないようにするために、アクティブにアクセスされているデバイスやファイルに複数プロセスから調整なく書き込みアクセスをしてはいけません。flock(1) を使ったファイルロック機構がこの回避に使えます。

データーのセキュリティーとそのコントロールされた共有はいくつかの側面があります。

- データーアーカイブの作成
- 遠隔ストレージアクセス
- 複製
- 変更履歴の追跡
- データー共有のアシスト
- 不正なファイルへのアクセスの防止
- 不正なファイルの改変の検出

こういったことは次の組み合わせを使うことで実現できます。

- アーカイブと圧縮ツール
- コピーと同期ツール
- ネットワークファイルシステム
- リムーバブルストレージメディア
- セキュアーシェル
- 認証システム
- バージョンコントロールシステムツール
- ハッシュや暗号学的暗号化ツール

10.1.1 アーカイブと圧縮ツール

Debian システム上で利用可能なアーカイブと圧縮ツールのまとめを以下に記します。

パッケージ	ポプコン	サイズ	拡張子	コマンド	コメント
tar	V:905, I:999	3098	.tar	tar(1)	標準アーカイバー (デファクト標準)
cpio	V:412, I:998	1136	.cpio	cpio(1)	Unix System V スタイルのアーカイバー、find(1) とともに使用
binutils	V:164, I:678	97	.ar	ar(1)	静的ライブラリー生成用のアーカイバー
fastjar	V:2, I:29	183	.jar	fastjar(1)	Java 用のアーカイバー (zip 類似)
pax	V:13, I:26	170	.pax	pax(1)	新規 POSIX 標準アーカイバー、tar と cpio の間の妥協点
gzip	V:883, I:999	245	.gz	gzip(1), zcat(1), …	GNU LZ77 圧縮ユーティリティー (デファクト標準)
bzip2	V:157, I:970	122	.bz2	bzip2(1), bzcat(1), …	gzip(1) より高い圧縮比 (gzip より遅い、類似シンタックス) の Burrows-Wheeler ブロック並び替え圧縮ユーティリティー
lzma	V:2, I:29	149	.lzma	lzma(1)	gzip(1) より高い圧縮比の LZMA 圧縮ユーティリティー (非推奨)
xz-utils	V:454, I:977	612	.xz	xz(1), xzdec(1), …	bzip2(1) より高い圧縮比の XZ 圧縮ユーティリティー (gzip より遅いが bzip2 より早い、LZMA 圧縮ユーティリティーの代替)
p7zip	V:89, I:464	987	.7z	7zr(1), p7zip(1)	高い圧縮比をもつ 7-Zip 圧縮ユーティリティー (LZMA 圧縮)
p7zip-full	V:113, I:486	4664	.7z	7z(1), 7za(1)	高い圧縮比をもつ 7-Zip 圧縮ユーティリティー (LZMA 圧縮、他)
lzop	V:9, I:76	164	.lzo	lzop(1)	gzip(1) より高い圧縮と解凍の速度 (gzip より低い圧縮比、類似シンタックス) の LZO 圧縮ユーティリティー
zip	V:51, I:432	608	.zip	zip(1)	InfoZIP: DOS アーカイブと圧縮ツール
unzip	V:154, I:798	566	.zip	unzip(1)	InfoZIP: DOS アーカイブ解凍と圧縮解凍ツール

Table 10.1: アーカイブと圧縮ツールのリスト

警告
何が起こるかを理解せずに”$TAPE” 変数を設定してはいけません。設定すると tar(1) の挙動が変わります。

注意
gzip 圧縮された tar(1) アーカイブは”.tgz” とか”.tar.gz” といったファイル拡張子を使います。

注意
xz 圧縮された tar(1) アーカイブは”.txz” とか”.tar.xz” といったファイル拡張子を使います。

注意
tar(1) 等の FOSS ツールでのポピュラーな圧縮方法は次のように変遷しています: `gzip` → `bzip2` → `xz`

注意
cp(1) と scp(1) と tar(1) は特殊ファイルに関して一部制約があるかもしれません。cpio(1) は最も汎用性があります。

注意
cpio(1) は find(1) 等のコマンドとともに使うようにできていて、ファイルの選定部分のスクリプトを独立にテストできるのでバックアップスクリプトを作るのに向いています。

注意
Libreoffice データーファイルの内部構造は"`.jar`" ファイルで、`unzip` で開くことができます。

注意
デファクトのクロスプラットフォームのアーカイブツールは `zip` です。最大限のコンパチビリティーのためには"`zip -rX`" として使ってください。もし最大ファイルサイズが問題となる際には"`-s`" オプションも使ってください。

10.1.2 コピーと同期ツール

Debian システム上で利用可能な単純なコピーとバックアップツールのまとめを以下に記します。

パッケージ	ポプコン	サイズ	ツール	機能
`coreutils`	V:891, I:999	17478	GNU cp	ファイルやディレクトリーのローカルコピー ("-a" で再帰的実行)
`openssh-client`	V:803, I:996	4298	scp	ファイルやディレクトリーのリモートコピー (クライアント、"`-r`" で再帰実行)
`openssh-server`	V:690, I:834	1567	sshd	ファイルやディレクトリーのリモートコピー (リモートサーバー)
`rsync`	V:281, I:560	677	-	単方向リモート同期とバックアップ
`unison`	V:4, I:17	14	-	双方向リモート同期とバックアップ

Table 10.2: コピーと同期ツールのリスト

rsync(8) を使ってのファイルのコピーには他の方法より豊かな機能があります。

- 転送元のファイルと転送先の既存ファイル間の相違のみを送信する差分転送アルゴリズム
- サイズか最終変更時間に変更があったファイルのみを探す (デフォルトで採用される) 急速確認アルゴリズム
- tar(1) 類似の"`--exclude`" や"`--exclude-from`" オプション
- 転送先に追加ディレクトリーレベルを作成しなくする「転送元ディレクトリ後スラシュ (/) 付加」文法

ティップ
項10.2.3 に記された `bkup` スクリプトを"`-gl`" オプションとともに cron(8) の下で実行すると静的なデーターアーカイブに関して Plan9 の `dumpfs` と非常に似た機能を実現できます。

ティップ
表 10.11に記されたバージョンコントロールシステム (VCS) ツールは多方向のコピーと同期のツールとして機能します。

10.1.3 アーカイブの慣用句

”./source” ディレクトリー中の全内容を異なるツールを用いてアーカイブしアーカイブ解凍するいくつかの方法を以下に記します。

GNU tar(1):

```
$ tar -cvJf archive.tar.xz ./source
$ tar -xvJf archive.tar.xz
```

この代わりに、次のようにも出来ます。

```
$ find ./source -xdev -print0 | tar -cvJf archive.tar.xz --null -F -
```

cpio(1):

```
$ find ./source -xdev -print0 | cpio -ov --null > archive.cpio; xz archive.cpio
$ zcat archive.cpio.xz | cpio -i
```

10.1.4 コピーの慣用句

”./source” ディレクトリー中の全内容を異なるツールを用いてコピーするいくつかの方法を以下に記します。

- ローカルコピー: ”./source” ディレクトリー → ”/dest” ディレクトリー
- リモートコピー: ローカルホストの”./source” ディレクトリー → ”user@host.dom” ホストの”/dest” ディレクトリー

rsync(8):

```
# cd ./source; rsync -aHAXSv . /dest
# cd ./source; rsync -aHAXSv . user@host.dom:/dest
```

「転送元ディレクトリー後スラシュ付加」文法を上記の代わりに使えます。

```
# rsync -aHAXSv ./source/ /dest
# rsync -aHAXSv ./source/ user@host.dom:/dest
```

この代わりに、次のようにも出来ます。

```
# cd ./source; find . -print0 | rsync -aHAXSv0 --files-from=- . /dest
# cd ./source; find . -print0 | rsync -aHAXSv0 --files-from=- . user@host.dom:/dest
```

GNU cp(1) と openSSH scp(1):

```
# cd ./source; cp -a . /dest
# cd ./source; scp -pr . user@host.dom:/dest
```

GNU tar(1):

```
# (cd ./source && tar cf - . ) | (cd /dest && tar xvfp - )
# (cd ./source && tar cf - . ) | ssh user@host.dom '(cd /dest && tar xvfp - )'
```

cpio(1):

```
# cd ./source; find . -print0 | cpio -pvdm --null --sparse /dest
```

”.” を含むすべての例で”.” は”foo” で代替でき、ファイルを”./source/foo” ディレクトリーから”/dest/foo” ディレクトリにコピーできます。

”.” を含むすべての例で”.” を絶対パスの”/path/to/source/foo” で代替でき、”cd ./source;” を削除することができます。これらは使うツール次第で異なる場所にファイルをコピーします。

- ”/dest/foo”: rsync(8)、GNU cp(1)、scp(1)
- ”/dest/path/to/source/foo”: GNU tar(1) と cpio(1)

ティップ
rsync(8) や GNU cp(1) には転送先のファイルが新しい場合にスキップする”-u” オプションがあります。

10.1.5 ファイル選択の慣用句

アーカイブやコピーコマンド (項10.1.3と項10.1.4参照下さい) のためや xargs(1) (項9.3.9参照下さい) のためにファイルを選択するのに find(1) が使われます。これの操作は find(1) のコマンド引数を使うことで強化できます。

find(1) の基本シンタックスは次のようにまとめられます。

- 条件の引数は左から右へと評価されます。
- 結果が決まった時点で評価は終了します。
- ” 論理 **OR**” (条件間に”-o” で指定) は、” 論理 **AND**” (条件間に”-a” または何もなしで指定) より低い優先順位です。
- ” 論理 **NOT**” (条件前に”!” で指定) は、” 論理 **AND**” より高い優先順位です。
- ”-prune” は常に論理真 **(TRUE)** を返し、ディレクトリーの場合にはこの点以降のファイル探索を停止します。
- ”-name” はシェルのグロブ (項1.5.6参照下さい) を使ってファイル名のベースにマッチし、さらに”*” and ”?” 等のメタ文字で最初の”.” ともマッチします。(新規の POSIX 機能)
- ”-regex” はデフォールトでは emacs スタイルの **BRE** (項1.6.2参照下さい) を用いてフルパスをマッチします。
- ”-size” はファイルサイズ (”+” が前に付いた値はより大きい、”-” が前に付いた値はより小さい) に基づいてファイルをマッチします。
- ”-newer” はその引数に指定されたファイルより新しいファイルとマッチします。
- ”-print0” は常に論理真 **(TRUE)** を返し、フルファイル名を (null 終端処理して) 標準出力へプリントします。

find(1) はしばしば慣用的なスタイルで使われます。

```
# find /path/to \
    -xdev -regextype posix-extended \
    -type f -regex ”.*\.cpio|.*~” -prune -o \
    -type d -regex ”.*/\.git” -prune -o \
    -type f -size +99M -prune -o \
    -type f -newer /path/to/timestamp -print0
```

これは次のアクションをすることを意味します。

1. ”/path/to” からはじまる全ファイルを探索

2. 探索開始したファイルシステムに探索を全体的に制約し、デフォールトの代わりに **ERE** (項1.6.2参照下さい) を使用
3. 正規表現”`.*\.cpio`” か”`.*~`” にマッチするファイルを処理停止をすることで探索から除外
4. 正規表現”`.*/\.git`” にマッチするディレクトリーを処理停止をすることで探索から除外
5. 9MiB(1048576 バイトの単位) より大きいファイルを処理停止をすることで探索から除外
6. 上記の探索条件に合致し”`/path/to/timestamp`” より新しいファイル名をプリントします

上記例中でファイルを検索から除外するときの”`-prune -o`” の慣用的な使い方に注目して下さい。

注意
非 Debian のUnix 的システムでは、一部のオプションは find(1) によってサポートされていないかもしれません。そのような場合には、マッチング方法を調整したり”`-print0`” を”`-print`” で置き換えることを考慮します。これに関連するコマンドも調整する必要があるかもしれません。

10.1.6 アーカイブメディア

重要なデーターアーカイブのためのコンピューターデーターストレージメディアを選ぶ時にはそれらの限界について注意を払うべきです。小さな個人的なバックアップのためには、著者としては名前が知られている会社の CD-R と DVD-R を使い、クールで日陰の乾燥した埃の無い環境に保存しています。(プロ用途ではテープアーカイブメディアに人気があるようです。)

注意
耐火金庫は紙の文書のためのものです。ほとんどのコンピューターデーターストレージメディアは紙よりも耐熱性がありません。著者は通常複数の安全な場所に保管された複数のセキュアーな暗号化されたコピーに頼っています。

ネット上に散見するアーカイブメディアの楽観的なストレージ寿命 (ほとんどベンダー情報由来)。

- 100+ 年: インクと中性紙
- 100 年: オプティカルストレージ (CD/DVD、CD/DVD-R)
- 30 年: 磁気ストレージ (テープ、フロッピー)
- 20 年: 相変化オプティカルストレージ (CD-RW)

これらは取扱いによる機械的故障等は考慮していません。

ネット上に散見するアーカイブメディアの楽観的な書込み回数 (ほとんどベンダー情報由来)。

- 250,000+ 回: ハードディスク
- 10,000+ 回: フラッシュメモリー
- 1,000 回: CD/DVD-RW
- 1 回: CD/DVD-R、紙

注意
ここにあるストレージ寿命や書込み回数の数字はクリチカルなデーターストレージに関する決定に使うべきではありません。製造者によって提供される特定の製品情報を参照下さい。

ティップ
CD/DVD-R や紙は 1 回しか書けないので、本質的に重ね書きで間違ってデーターを消すことを防げます。これは、利点です!

ティップ
もし高速で頻繁な大量のデーターのバックアップをする必要がある場合、高速のネットワーク接続でつながっているリモートホスト上のハードディスクが唯一の現実的なオプションかもしれません。

10.1.7 リムーバブルストレージデバイス

リムーバブルストレージデバイスは次の何れも指します。

- USB フラッシュドライブ
- Hard ディスクドライブ
- 光学ディスクドライブ
- デジタルカメラ
- デジタル音楽プレーヤー

これらは次の何れかで接続できます。

- USB
- IEEE 1394 / FireWire
- PC カード

GNOME や KDE のような最近のデスクトップ環境は、”/etc/fstab” エントリーにマッチが無いリムーバブルデバイスを自動的にマウントする事ができます。

- udisks パッケージは、これらのデバイスをマウントやアンマウントするためのデーモンと関連するユーティリティーを提供します。
- D-bus は、自動的なプロセスを開始するイベントを作成します。
- PolicyKit が必要な特権を提供します。

ティップ
自動的にマウントされたデバイスは、umount(8) によって利用される”uhelper=” マウントオプションが設定されているかもしれません。

ティップ
”/etc/fstab” にリムーバブルメディアデバイスの記載が無い時のみ、現代的なデスクトップ環境下での自動マウントは起こります。

最新のデスクトップ環境下では次のようにしてカスタマイズ可能なマウント点として”/media/<disk_label>” が選ばれます。

- FAT ファイルシステムでは、mlabel(1) を使います。

- ISO9660 ファイルシステムでは、genisoimage(1) を”-V” オプションとともに使います。
- ext2/ext3/ext4 ファイルシステムでは、tune2fs(1) を”-L” オプションとともに使います。

ティップ
符号化方式 (エンコーディング) の選択をマウントオプションとして与える必要があるかもしれません (項8.4.6参照下さい)。

ティップ
ファイルシステムをアンマウンとする際に GUI メニューを使うと、動的に生成された”/dev/sdc” 等のデバイスノード削除するかもしれません。もしそのデバイスノードの削除したくない場合にはシェルのコマンドプロンプトから umount(8) コマンドを使いましょう。

10.1.8 データー共有用のファイルシステム選択

リムーバブルストレージデバイスを使ってデーターを共有する際には、両方のシステムにサポートされた共通のファイルシステムでそれをフォーマットするべきです。ファイルシステム選択のリストを次に示します。

ファイルシステム	典型的使用シナリオの説明
FAT12	フロッピーディスク上のクロスプラットフォームのデーター共有 (<32MiB)
FAT16	小さなハードディスク類似のデバイス上のクロスプラットフォームのデーター共有 (<2GiB)
FAT32	大きなハードディスク類似のデバイス上のクロスプラットフォームのデーター共有 (<8TiB, MS Windows95 OSR2 以降でサポート有り)
NTFS	大きなハードディスク類似のデバイス上のクロスプラットフォームのデーター共有 (MS Windows NT 以降でネイティブにサポート、Linux 上では FUSE 経由の NTFS-3G でサポート)
ISO9660	CD-R and DVD+/-R 上の静的データーのクロスプラットフォームの共有
UDF	CD-R や DVD+/-R 上への増分データーの書込み (新規)
MINIX ファイルシステム	フロッピーディスク上へのスペース効率の良い unix ファイルデーターのストレージ
ext2 ファイルシステム	古い Linux システムとハードディスク類似デバイス上のデーターを共有
ext3 ファイルシステム	古い Linux システムとハードディスク類似デバイス上のデーターを共有
ext3 ファイルシステム	最新の Linux システムとハードディスク類似デバイス上のデーターを共有

Table 10.3: 典型的な使用シナリオに合わせたリムーバブルストレージデバイスのファイルシステムの選択肢のリスト

ティップ
デバイスレベルの暗号化を使ったクロスプラットフォームのデーター共有に関しては、項9.8.1を参照下さい。

FAT ファイルシステムはほとんど全ての現代的なオペレーティングシステムでサポートされていて、ハードディスク類似のメディア経由でのデーター交換目的に非常に有用です。

クロスプラットフォームの FAT ファイルシステムを使ったデーター共有にリムーバブルハードディスク類似デバイスをフォーマットする時の安全な選択肢は次です。

- fdisk(8) か cfdisk(8) か parted(8) (項9.5.2参照下さい) を使ってそれを単一のプライマリパーティションにパーティションしそれを次のようにマークします。
  - 2GB より小さなメディアには FAT16 となるように”6” とタイプします
  - 大きなメディアには FAT32 (LBA) となるように”c” とタイプします
- 第 1 パーティションを mkfs.vfat(8) を使って次のようにフォーマットします。
  - FAT16 となるように”`/dev/sda1`” 等とそのデバイス名だけを使います
  - FAT32 となるように”`-F 32 /dev/sda1`” 等と明示的なオプション指定とそのデバイス名を使います

FAT とか ISO9660 ファイルシステムを使ってデーターを共有する際の安全への配慮を次に記します。

- tar(1) や cpio(1) を使ってアーカイブファイルに最初にファイルをアーカイブすることで長いファイル名やシンボリックリンクやオリジナルの Unix ファイルパーミッションとオーナー情報を保持します。
- split(1) コマンドを使ってアーカイブファイルを 2GiB 以下の塊に分割してファイルサイズの制約から保護します。
- アーカイブファイルを暗号化してその内容を不正アクセスから保護します。

注意
FAT ファイルシステムはその設計上最大ファイルサイズは `(2^32 - 1) bytes = (4GiB - 1 byte)` です。古い 32 ビット OS 上の一部アプリケーションは、最大ファイルサイズはさらに小さく `(2^31 - 1) bytes = (2GiB - 1 byte)` です。Debian は後者の問題に苦しむことはありません。

注意
Microsoft 自身も 200MB を越すドライブやパーティションに FAT を使うことを勧めていません。マイクロソフトは、彼らの”Overview of FAT, HPFS, and NTFS File Systems” で非効率的なディスク領域の使用等の欠点をハイライトしています。もちろん私たちは Linux では通常 ext4 ファイルシステムを使うべきです。

ティップ
ファイルシステムとファイルシステムのアクセスに関しての詳細は、”Filesystems HOWTO” を参照下さい。

10.1.9 ネットワーク経由でのデーター共有

データーをネットワーク経由で他のシステムと共有するときには、共通のサービスを使うべきです。次に一部のヒントを記します。

このようなネットワーク経由でマウントされたファイルシステムやネットワーク経由のファイル転送法はデーター共有のために非常に便利ですが、インセキュアーかもしれませんこれらのネットワーク接続は次に記すようにしてセキュアーにされなければいけません。

- SSL/TLS を使い暗号化
- SSH 経由でそれをトンネル
- VPN 経由でそれをトンネル
- セキュアーファイアウォールの背後に限定

さらに項6.10と項6.11を参照下さい。

ネットワークサービス	典型的使用シナリオの説明
Samba を使う SMB/CIFS ネットワーク経由マウントファイルシステム	”Microsoft Windows Network” 経由でのファイル共有、smb.conf(5) と The Official Samba 3.x.x HOWTO and Reference Guide か `samba-doc` パッケージ参照下さい
Linux カーネルを使う NFS ネットワークマウントファイルシステム	”Unix/Linux Network” 経由のファイル共有、exports(5) と Linux NFS-HOWTO 参照下さい。
HTTP サービス	ウェッブサーバー/クライアント間のファイル共有
HTTPS サービス	暗号化されたセキュアーソケットレイヤー (SSL) もしくは Transport Layer Security (TLS) を使ったウェッブサーバー/クライアント間のファイル共有
FTP サービス	FTP サーバー/クライアント間のファイル共有

Table 10.4: 典型的使用シナリオの場合のネットワークサービスの選択のリスト

10.2 バックアップと復元

コンピューターはいつか壊れるとか、人間によるエラーがシステムやデーターをへの損害を及ぼすことは皆知っています。バックアップと復元の操作は正しいシステム管理の必須構成要素です。考えうる全ての故障モードはいつかの日にやって来ます。

ティップ
バックアップのシステムは簡単にしておき、システムのバックアップは頻繁にします。バックアップデーターが存在することは、あなたのバックアップ方法が技術的に如何に良いかよりも重要です。

実際のバックアップと復元の方針を決める上で 3 つの要素があります。

1. 何をバックアップし復元するかを知っていること
   - あなた自身が作成したデーターファイル: ”~/” 中のデーター
   - あなた自身が使用したアプリケーションが作成したデーターファイル: ”`/var/`” (”`/var/cache/`” と”`/var/run/`” と”`/var/tmp/`” は除外) 中のデーター
   - システム設定ファイル: ”`/etc/`” 中のデーター
   - ローカルデーター: ”`/usr/local/`” とか”`/opt/`” 中のデーター
   - システムインストール情報: 要点 (パーティション、…) をプレーンテキストで書いたメモ
   - 実証済みのデーターセット: 事前に実験的復元操作をして確認済み
2. バックアップと復元の方法を知っていること
   - セキュアーなデーターのストレージ: 上書きやシステム障害の防止
   - 頻繁なバックアップ: スケジュールされたバックアップ
   - 冗長なバックアップ: データーのミラーリング
   - フルプルーフなプロセス: 簡単な単一コマンドバックアップ
3. 関わっているリスクと費用の評価
   - データーがなくなった際の価値
   - バックアップに必要なリソース: 人的、ハードウエアー、ソフトウエアー、…
   - 故障モードとその確率

注意
/proc や /sys や /tmp や /run 上にある擬似ファイルシステム (項1.2.12 と項1.2.13 参照) の内容をバックアップしてはいけません。あなた自身が自分がしていることの意味を余程よく分かっていなければ、これらの内容は巨大で無用なデーターです。

データーのセキュアーなストレージとして、好ましくはファイルシステム破壊に耐えるように異なるディスクや機器上に、少なくとも異なるディスクパーティション上に、データーはあるべきです。重要データーは上書き事故を防ぐために CD/DVD-R のような 1 回書込みメディアに貯蔵するのが好ましいです。(シェルコマンドラインからストレージメディアにどうして書き込むかについては項9.7を参照下さい。GNOME デスクトップの GUI 環境ではメニュー: ”Places → CD/DVD Creator” で簡単に書込みできます。)

注意
データーをバックアップする際には MTA (項6.3参照下さい) 等のアプリケーションデーモンを停止するのも一計です。

注意
”/etc/ssh/ssh_host_dsa_key” や”/etc/ssh/ssh_host_rsa_key” や”~/.gnupg/*” や”~/.ssh/*” や”/etc/passwd” や”/etc/shadow” や”/etc/fetchmailrc” や”popularity-contest.conf” や”/etc/ppp/pap-secrets” や”/etc/exim4/passwd.client” 等のアイデンティティ関連のデーターファイルのバックアップと修復には特に注意が必要です。これらのデーターの一部はシステムに同様な入力をしても再生成できません。

注意
ユーザープロセスで cron ジョブを実行している際には、”/var/spool/cron/crontabs” ディレクトリー中のファイルを回復し cron(8) を再スタートしなければいけません。cron(8) と crontab(1) に関しては項9.3.14を参照下さい。

10.2.1 バックアプユーティリティーのスイート

Debian システム上で利用可能でバックアップユーティリティーのスイートのなかで際立った選ばれたリストを記します。

バックアップツールにはそれぞれの特別な狙いがあります。

- Mondo Rescue を使うと、通常のインストールプロセスを経ずにバックアップ CD/DVD 等から完全なシステムを迅速に復旧できます。
- ユーザーデーターの定期的バックアップ機能は簡単なスクリプト (項10.2.2) と cron(8) で実現できます。
- Bacula と Amanda と BackupPC は、ネットワーク越しの定期的バックアップに焦点のあるフル機能のバックアップスイートです。

項10.1.1や項10.1.2に記された基本的なツールを使うとカスタムスクリプト経由のシステムバックアップができます。そのようなスクリプトは次を使うと強化できます。

- restic パッケージは差分 (遠隔) バックアップを提供します。
- rdiff-backup パッケージは (リモートの) 増分バックアップを可能にします。
- dump パッケージは全ファイルシステムの効率的かつ増分のバックアップと復旧を補助します。

ティップ
dump パッケージに関して学ぶには、”/usr/share/doc/dump/” の中のファイルと ”Is dump really deprecated?” を参照下さい。

パッケージ	ポプコン	サイズ	説明
dump	V:1, I:6	352	ext2/ext3/ext4 ファイルシステム用の 4.4 BSD 由来の dump(8) と restore(8)
xfsdump	V:0, I:9	854	GNU/Linux と IRIX 上の XFS ファイルシステム用の xfsdump(8) と xfsrestore(8) を使う dump と restore
backupninja	V:4, I:5	355	軽量で拡張可のメタバックアップシステム
bacula-common	V:10, I:15	2158	Bacula: ネットワークバックアップ、復元および検証 - 共通のサポートファイル
bacula-client	I:3	183	Bacula: ネットワークバックアップ、復元および検証 - クライアントメタパッケージ
bacula-console	V:1, I:5	107	Bacula: ネットワークバックアップ、復元および検証 - テキストコンソール
bacula-server	I:1	183	Bacula: ネットワークバックアップ、復元および検証 - サーバーメタパッケージ
amanda-common	V:0, I:2	10031	Amanda: Advanced Maryland Automatic Network Disk Archiver (ライブラリー)
amanda-client	V:0, I:2	1089	Amanda: Advanced Maryland Automatic Network Disk Archiver (クライアント)
amanda-server	V:0, I:0	1076	Amanda: Advanced Maryland Automatic Network Disk Archiver (サーバー)
backup-manager	V:1, I:2	572	コマンドラインのバックアップツール
backup2l	V:0, I:1	114	マウントできるメディアのための低メンテナンスのバックアップ/復旧ツール (ディスクベース)
backuppc	V:3, I:3	3182	BackupPC は高性能でエンタープライズ級の、PC をバックアップするためのシステム (ディスクベース)
duplicity	V:7, I:15	1761	(リモート) 増分バックアップ
flexbackup	V:0, I:0	243	(リモート) 増分バックアップ
rdiff-backup	V:7, I:15	733	(リモート) 増分バックアップ
restic	V:1, I:3	20595	(リモート) 増分バックアップ
rsnapshot	V:5, I:11	462	(リモート) 増分バックアップ
slbackup	V:0, I:0	151	(リモート) 増分バックアップ

Table 10.5: バックアップスイートのユーティリティーのリスト

10.2.2 システムバックアップ用スクリプトの例

個人の Debian の unstable (非安定) スイートを実行するデスクトップシステムでは、個人的だったりクリティカルなデーターのみを保護する必要しかありません。いずれにせよ 1 年に 1 回はシステムを再インストールします。だから全システムをバックアップする理由もありませんし、フル機能のバックアップユーティリティーをインストールする理由もありません。

簡単なスクリプトでバックアップ用アーカイブを作成し、GUI を使って CD/DVD にそれを焼きます。次のこのスクリプトの例を示します。

```
#!/bin/sh -e
# Copyright (C) 2007-2008 Osamu Aoki <osamu@debian.org>, Public Domain
BUUID=1000; USER=osamu # UID and name of a user who accesses backup files
BUDIR="/var/backups"
XDIR0=".+/Mail|.+/Desktop"
XDIR1=".+/\.thumbnails|.+/\.?Trash|.+/\.?[cC]ache|.+/\.gvfs|.+/sessions"
XDIR2=".+/CVS|.+/\.git|.+/\.svn|.+/Downloads|.+/Archive|.+/Checkout|.+/tmp"
XSFX=".+\.iso|.+\.tgz|.+\.tar\.gz|.+\.tar\.bz2|.+\.cpio|.+\.tmp|.+\.swp|.+~"
SIZE="+99M"
DATE=$(date --utc +"%Y%m%d-%H%M")
[ -d "$BUDIR" ] || mkdir -p "BUDIR"
umask 077
dpkg --get-selections \* > /var/lib/dpkg/dpkg-selections.list
debconf-get-selections > /var/cache/debconf/debconf-selections

{
find /etc /usr/local /opt /var/lib/dpkg/dpkg-selections.list \
     /var/cache/debconf/debconf-selections -xdev -print0
find /home/$USER /root -xdev -regextype posix-extended \
  -type d -regex "$XDIR0|$XDIR1" -prune -o -type f -regex "$XSFX" -prune -o \
  -type f -size  "$SIZE" -prune -o -print0
find /home/$USER/Mail/Inbox /home/$USER/Mail/Outbox -print0
find /home/$USER/Desktop  -xdev -regextype posix-extended \
  -type d -regex "$XDIR2" -prune -o -type f -regex "$XSFX" -prune -o \
  -type f -size  "$SIZE" -prune -o -print0
} | cpio -ov --null -O $BUDIR/BU$DATE.cpio
chown $BUUID $BUDIR/BU$DATE.cpio
touch $BUDIR/backup.stamp
```

root から実行されるスクリプト断片の例と言う位置づけです。

著者はこれをあなたが次のように変更し実行する事を期待します。

- このスクリプトを編集して全てのあなたの重要なデーターをカバーしましょう (項10.1.5と項10.2参照下さい)。
- 増分バックアップをするには、"`find …-print0`" を、"`find …-newer $BUDIR/backup.stamp -print0`" で置き換えます。
- scp(1) か rsync(1) を使ってリモートホストにバックアップファイルを転送したり、それらを更なるデーターセキュリティーのために CD/DVD に焼きます。(私は CD/DVD に焼くのに GNOME デスクトップの GUI を使っています。更なる冗長性に関しては項12.1.8を参照下さい。)

簡潔にしましょう!

ティップ
debconf の設定データーは"`debconf-set-selections debconf-selections`" で、dpkg の選択データーは"`dpkg --set-selection <dpkg-selections.list`" で復元できます。

10.2.3 データーバックアップ用コピースクリプト

ディレクトリーツリーの下のデーターセットは、”cp -a” としてコピーすると通常のバックアップができます。

”/var/cache/apt/packages/” ディレクトリーの下のデーターのようなディレクトリーの下の大きな上書きされない静的なデーターセットは、”cp -al” を利用するハードリンクを使うと通常のバックアップに代わるディスク空間を効率的に利用するバックアップができます。

私が bkup と名付けたデーターバックアップのためのコピースクリプトを次に示します。このスクリプトは現ディレクトリーの下の全ての (非 VCS) ファイルを親ディレクトリーかリモートホストの日付入りディレクトリーにコピーします。

```
#!/bin/sh -e
# Copyright (C) 2007-2008 Osamu Aoki <osamu@debian.org>, Public Domain
fdot(){ find . -type d \( -iname ”.?*” -o -iname ”CVS” \) -prune -o -print0;}
fall(){ find . -print0;}
mkdircd(){ mkdir -p ”$1”;chmod 700 ”$1”;cd ”$1”>/dev/null;}
FIND=”fdot”;OPT=”-a”;MODE=”CPIOP”;HOST=”localhost”;EXTP=”$(hostname -f)”
BKUP=”$(basename $(pwd)).bkup”;TIME=”$(date  +%Y%m%d-%H%M%S)”;BU=”$BKUP/$TIME”
while getopts gcCsStrlLaAxe:h:T f; do case $f in
g)  MODE=”GNUCP”;; # cp (GNU)
c)  MODE=”CPIOP”;; # cpio -p
C)  MODE=”CPIOI”;; # cpio -i
s)  MODE=”CPIOSSH”;; # cpio/ssh
t)  MODE=”TARSSH”;; # tar/ssh
r)  MODE=”RSYNCSSH”;; # rsync/ssh
l)  OPT=”-alv”;; # hardlink (GNU cp)
L)  OPT=”-av”;;  # copy (GNU cp)
a)  FIND=”fall”;; # find all
A)  FIND=”fdot”;; # find non CVS/ .???/
x)  set -x;; # trace
e)  EXTP=”${OPTARG}”;; # hostname -f
h)  HOST=”${OPTARG}”;; # user@remotehost.example.com
T)  MODE=”TEST”;; # test find mode
\?) echo ”use -x for trace.”
esac; done
shift $(expr $OPTIND - 1)
if [ $# -gt 0 ]; then
  for x in $@; do cp $OPT $x $x.$TIME; done
elif [ $MODE = GNUCP ]; then
  mkdir -p ”../$BU”;chmod 700 ”../$BU”;cp $OPT . ”../$BU/”
elif [ $MODE = CPIOP ]; then
  mkdir -p ”../$BU”;chmod 700 ”../$BU”
  $FIND|cpio --null --sparse -pvd ../$BU
elif [ $MODE = CPIOI ]; then
  $FIND|cpio -ov --null | ( mkdircd ”../$BU”&&cpio -i )
elif [ $MODE = CPIOSSH ]; then
  $FIND|cpio -ov --null|ssh -C $HOST ”( mkdircd \”$EXTP/$BU\”&&cpio -i )”
elif [ $MODE = TARSSH ]; then
  (tar cvf - . )|ssh -C $HOST ”( mkdircd \”$EXTP/$BU\”&& tar xvfp - )”
elif [ $MODE = RSYNCSSH ]; then
  rsync -aHAXSv ./ ”${HOST}:${EXTP}-${BKUP}-${TIME}”
else
  echo ”Any other idea to backup?”
  $FIND |xargs -0 -n 1 echo
fi
```

これはコマンド例の位置付けです。スクリプトを読んで自分自身で編集してからご使用下さい。

ティップ
私はこの bkup を私の”/usr/local/bin/” ディレクトリーに置いています。わたしは一時的スナップショットのバックアップが必要な時に、作業ディレクトリー中で引数無しにこの bkup コマンドを実行します。

ティップ
ソースファイルツリーや設定ファイルツリーのスナップショットの履歴を作成するには、git(7) を使うのが簡単でスペースの効率も良ろしいです (項10.6.5参照下さい)。

10.3 データーセキュリティーのインフラ

データーのセキュリティーのインフラはデーターの暗号化のツールとメッセージダイジェストのツールと署名ツールの組み合わせで提供されます。

パッケージ	ポプコン	サイズ	コマンド	説明
gnupg	V:531, I:950	787	gpg(1)	GNU プライバシーガード - OpenPGP 暗号化ト署名ツール
gpgv	V:880, I:999	859	gpgv(1)	GNU プライバシガード - 署名確認ツール
paperkey	V:1, I:13	58	paperkey(1)	OpenPGP の秘密キーから秘密の情報だけを抜粋
cryptsetup	V:29, I:78	402	cryptsetup(8), …	暗号化されたブロックデバイス (dm-crypt / LUKS) のためのユーティリティー
ecryptfs-utils	V:3, I:5	460	ecryptfs(7), …	ecryptfs スタックドファイルシステム暗号化のためのユーティリティー
coreutils	V:891, I:999	17478	md5sum(1)	MD5 メッセージダイジェストを計算やチェック
coreutils	V:891, I:999	17478	sha1sum(1)	SHA1 メッセージダイジェストを計算やチェック
openssl	V:794, I:993	1465	openssl(1ssl)	”openssl dgst” を使ってメッセージダイジェストを計算やチェック (OpenSSL)

Table 10.6: データーセキュリティーインフラツールのリスト

Linux カーネルモジュール経由で自動的データー暗号化のインフラを実現する dm-crypto と ecryptfs に関しては項9.8を参照下さい。

10.3.1 Gnupg のためのキー管理

基本的なキー管理に関する GNU プライバシガードコマンドを次に記します。

コマンド	説明
gpg --gen-key	新規キーの生成
gpg --gen-revoke my_user_ID	my_user_ID に関するリボークキーを生成
gpg --edit-key user_ID	インタラクティブにキーを編集、ヘルプは”help”
gpg -o file --export	全てのキーをファイルにエクスポート
gpg --import file	全てのキーをファイルからインポート
gpg --send-keys user_ID	user_ID のキーをキーサーバーに送信
gpg --recv-keys user_ID	user_ID のキーをキーサーバーから受信
gpg --list-keys user_ID	user_ID のキーをリスト
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 --refresh-keys	ローカルキーリングをアップデート

Table 10.7: キー管理のための GNU プライバシガードコマンドのリスト

コード	信用の説明
-	所有者への信用未付与/未計算
e	信用計算に失敗
q	計算用の情報不十分
n	このキーを信用不可
m	スレスレの信用
f	フルに信用
u	究極の信用

Table 10.8: トラストコードの意味のリスト

トラストコードの意味を次に記します。

次のようにすると私のキー”1DD8D791” をポピュラーなキーサーバー”hkp://keys.gnupg.net” にアップロード出来ます。

```
$ gpg --keyserver hkp://keys.gnupg.net --send-keys 1DD8D791
```

”~/.gnupg/gpg.conf” (もしくは古い場所”~/.gnupg/options”) 中の良いデフォールトのキーサーバーの設定は次を含みます。

```
keyserver hkp://keys.gnupg.net
```

次によってキーサーバーから知らないキーが獲得できます。

```
$ gpg --list-sigs --with-colons | grep '^sig.*\[User ID not found\]' |\
  cut -d ':' -f 5| sort | uniq | xargs gpg --recv-keys
```

OpenPGP 公開キーサーバー (バージョン 0.9.6 以前) に 2 つ以上サブキーのあるキーを壊すバグがありました。新しい gnupg (>1.2.1-2) パッケージはこのような壊れたサブキーを取り扱えます。gpg(1) の”--repair-pks-subkey-bug” オプションの説明を参照下さい。

10.3.2 GnuPG をファイルに使用

基本的なキー管理に関する GNU プライバシガードコマンドを次に記します。

10.3.3 Mutt で GnuPG を使用

インデックスメニュー上で”S” とすれば GnuPG が使えるようにしておきながら、遅い GnuPG が自動的に起動しないように”~/.muttrc” に次の内容を追加します。

```
macro index S ":toggle pgp_verify_sig\n"
set pgp_verify_sig=no
```

10.3.4 Vim で GnuPG を使用

gnupg のプラグインを使うと”.gpg” や”.asc” や”.ppg” というファイル拡張子のファイルに対して透過的に GnuPG を実行できます。

```
# aptitude install vim-scripts vim-addon-manager
$ vim-addons install gnupg
```

コマンド	説明
`gpg -a -s file`	ファイルを ASCII 文字化した file.asc と署名
`gpg --armor --sign file`	,,
`gpg --clearsign file`	メッセージをクリアサイン
`gpg --clearsign file|mail foo@example.org`	`foo@example.org` にクリアサインされたメッセージをメールする
`gpg --clearsign --not-dash-escaped patchfile`	パッチファイルをクリアサイン
`gpg --verify file`	クリアサインされたファイルを確認
`gpg -o file.sig -b file`	署名を別ファイルで作成
`gpg -o file.sig --detach-sig file`	,,
`gpg --verify file.sig file`	file.sig を使ってファイルを確認
`gpg -o crypt_file.gpg -r name -e file`	file からバイナリー crypt_file.gpg への name 宛公開キー暗号化
`gpg -o crypt_file.gpg --recipient name --encrypt file`	,,
`gpg -o crypt_file.asc -a -r name -e file`	file から ASCII 文字化された crypt_file.asc への name 宛公開キー暗号化
`gpg -o crypt_file.gpg -c file`	file からバイナリー crypt_file.gpg への対称暗号化
`gpg -o crypt_file.gpg --symmetric file`	,,
`gpg -o crypt_file.asc -a -c file`	file から ASCII 文字化された crypt_file.asc への対称暗号化
`gpg -o file -d crypt_file.gpg -r name`	暗号解読
`gpg -o file --decrypt crypt_file.gpg`	,,

Table 10.9: ファイルに使用する GNU プライバシーガードコマンドのリスト

10.3.5 MD5 和

md5sum(1) はrfc1321 の方法を使ってダイジェストファイルを作成し各ファイルをそれで確認するユーティリティーを提供します。

```
$ md5sum foo bar >baz.md5
$ cat baz.md5
d3b07384d113edec49eaa6238ad5ff00  foo
c157a79031e1c40f85931829bc5fc552  bar
$ md5sum -c baz.md5
foo: OK
bar: OK
```

注意
MD5 和の計算は GNU プライバシーガード (GnuPG) による暗号学的署名の計算より CPU への負荷がかかりません。通常、一番上のレベルのダイジェストファイルだけがデーターの整合性のために暗号学的に署名されます。

10.4 ソースコードマージツール

ソースコードをマージする多くのツールがあります。次のコマンドが著者の目に止まりました。

10.4.1 ソースファイル間の相違の抽出

ふたつのソースファイル間の相違を抽出したユニファイド差分ファイルは、以下の要領でファイル位置に対応し”`file.patch0`” か”`file.patch1`” として作成されます。

```
$ diff -u file.old file.new > file.patch0
$ diff -u old/file new/file > file.patch1
```

10.4.2 ソースファイルに更新をマージ

差分ファイル (別名、パッチファイル) はプログラム更新を送るのに使われます。受け取った側はこの更新を別のファイルに次のようにして適用します。

```
$ patch -p0 file < file.patch0
$ patch -p1 file < file.patch1
```

10.4.3 3 方マージによる更新

3 つのバージョンのソースコードがある場合、diff3(1) を使って効率的に 3 方マージを次のように実行できます。

```
$ diff3 -m file.mine file.old file.yours > file
```

パッケージ	ポプコン	サイズ	コマンド	説明
diffutils	V:871, I:991	1598	diff(1)	1 行ごとにファイルを比較
diffutils	V:871, I:991	1598	diff3(1)	1 行ごとにファイルを比較やマージ
vim	V:106, I:398	3231	vimdiff(1)	vim で 2 つのファイルを並べて比較
patch	V:99, I:725	248	patch(1)	差分ファイルをオリジナルに適用
dpatch	V:0, I:11	191	dpatch(1)	Debian パッケージのためにパッチのシリーズを管理
diffstat	V:16, I:154	73	diffstat(1)	差分ファイルによる変化のヒストグラム作成
patchutils	V:18, I:150	232	combinediff(1)	2 つの積み重ねパッチから 1 つの合計パッチを生成
patchutils	V:18, I:150	232	dehtmldiff(1)	HTML ページから差分ファイルを抽出
patchutils	V:18, I:150	232	filterdiff(1)	差分ファイルから差分ファイルを抽出や削除
patchutils	V:18, I:150	232	fixcvsdiff(1)	CVS により作成された patch(1) が誤解する差分ファイルを修正
patchutils	V:18, I:150	232	flipdiff(1)	古い 2 つのパッチを交換
patchutils	V:18, I:150	232	grepdiff(1)	正規表現にマッチするパッチによって変更されるファイルを表示
patchutils	V:18, I:150	232	interdiff(1)	2 つのユニファイド差分ファイル間の違いを表示
patchutils	V:18, I:150	232	lsdiff(1)	どのファイルがパッチによって変更されるかを表示
patchutils	V:18, I:150	232	recountdiff(1)	ユニファイドコンテキスト差分ファイルのカウントやオフセットを再計算
patchutils	V:18, I:150	232	rediff(1)	手編集された差分ファイルのカウントやオフセットを再計算
patchutils	V:18, I:150	232	splitdiff(1)	増分パッチの分離
patchutils	V:18, I:150	232	unwrapdiff(1)	ワードラップされたパッチを復元
wiggle	V:0, I:0	174	wiggle(1)	リジェクトされたパッチを適用
quilt	V:3, I:33	788	quilt(1)	パッチのシリーズを管理
meld	V:14, I:39	2972	meld(1)	ファイルを比較やマージ (GTK)
dirdiff	V:0, I:2	166	dirdiff(1)	ディレクトリーツリー間で相違点の表示と変更のマージ
docdiff	V:0, I:0	555	docdiff(1)	2 つのファイルをワード毎/文字毎に比較
imediff	V:0, I:0	157	imediff(1)	対話型フルスクリーンの 2 方/3 方マージツール
makepatch	V:0, I:0	102	makepatch(1)	拡張パッチファイルの生成
makepatch	V:0, I:0	102	applypatch(1)	拡張パッチファイルの適用
wdiff	V:9, I:72	644	wdiff(1)	テキストファイル間のワードの相違表示

Table 10.10: ソースコードマージツールのリスト

パッケージ	ポプコン	サイズ	ツール	VCS タイプ	コメント
cssc	V:0, I:2	2044	CSSC	ローカル	Unix SCCS のクローン (非推奨)
rcs	V:3, I:19	562	RCS	ローカル	”Unix SCCS の本来あるべき姿”
cvs	V:5, I:41	4609	CVS	リモート	リモート VCS の過去の標準
subversion	V:20, I:109	4858	Subversion	リモート	” 良く出来た CVS”、新しいリモート VCS のデファクト標準
git	V:305, I:478	35040	Git	分散型	C で書かれた高速 DVCS (Linux カーネル他が利用)
mercurial	V:8, I:48	1053	Mercurial	分散型	Python と一部 C で書かれた DVCS
bzr	V:2, I:16	28	Bazaar	分散型	tla に影響され Python で書かれた DVCS (Ubuntu が使用)
darcs	V:0, I:7	23159	Darcs	分散型	パッチに関して賢い計算をする DVCS (遅い)
tla	V:0, I:2	1011	GNU arch	分散型	主に Tom Lord による DVCS (歴史的)
monotone	V:0, I:0	5815	Monotone	分散型	C++ で書かれた DVCS
tkcvs	V:0, I:1	1498	CVS, …	リモート	VCS (CVS, Subversion, RCS) レポジトリーツリーの GUI 表示
gitk	V:6, I:42	1723	Git	分散型	VCS (Git) レポジトリーツリーの GUI 表示

Table 10.11: バージョンコントロールシステムツールのリスト

10.5 バージョンコントロールシステム

Debian システム上のバージョンコントロールシステム (VCS) のまとめを次に記します。

注意
VCS システムを今始めて学ぶ場合には、人気が出て来ている **Git** から学び出すことをお薦めします。

VCS はリビジョンコントロールシステム (RCS) とかソフトウエアー設定管理 (SCM) という別名もあります。

Git のような分散 VCS が最近一番人気のツールです。CVS や Subversion は既存のオープンソースプログラム活動に参加するのにまだ有用かもしれません。

Debian は Debian Salsa サービス経由でフリーの Git サービスを提供します。その説明文書は https://wiki.debian.org/Salsa にあります。

注意
Debian は既にその旧 alioth サービスを終了し、旧 alioth サービスのデーターは alioth-archive にターボールとしてあります。

VCS アーカイブへの共有アクセスを作るための基本が数点あります。

- ”`umask 002`” を使用 (項1.2.4参照下さい)
- すべての VCS アーカイブ中のファイルを適切なグループに所属させる
- すべての VCS アーカイブ中のディレクトリーのセットグループ ID を有効にする (BSD 式のファイル生成手順、項1.2.3参照下さい)
- VCS アーカイブを共有しているユーザーをグループに所属させる

10.5.1 VCS コマンドの比較

次に全体像を提供するべく元来の VCS コマンドを極端に簡略化した比較を示します。典型的なコマンド文字列はオプションや引数が必要となるかもしれません。

Git	CVS	Subversion	機能
`git init`	`cvs init`	`svn create`	(ローカル) レポジトリーを作成
-	`cvs login`	-	リモートレポジトリーにログイン
`git clone`	`cvs co`	`svn co`	リモートレポジトリーをワーキングツリーとしてチェックアウト
`git pull`	`cvs up`	`svn up`	リモートレポジトリーをマージしてワーキングツリーを更新
`git add .`	`cvs add`	`svn add`	VCS にワーキングツリー中のファイルを追加
`git rm`	`cvs rm`	`svn rm`	VCS からワーキングツリー中のファイルを削除
-	`cvs ci`	`svn ci`	リモートレポジトリーに変更をコミット
`git commit -a`	-	-	ローカルレポジトリーに変更をコミット
`git push`	-	-	ローカルレポジトリーでリモートレポジトリーを更新
`git status`	`cvs status`	`svn status`	VCS に対するワーキングツリーの状態を表示
`git diff`	`cvs diff`	`svn diff`	diff < 参照レポジトリー > < ワーキングツリー >
`git repack -a -d; git prune`	-	-	ローカルレポジトリーを単一パックにリパック
`gitk`	`tkcvs`	`tkcvs`	VCS レポジトリーツリーの GUI 表示

Table 10.12: 本来の VCS コマンドの比較

注意
`git` サブコマンドを直接"`git-xyz`" としてコマンドラインから起動するのは 2006 年初以来推奨されません。

ティップ
$PATH で指定されたパス中に実行可能ファイル `git-foo` が存在する場合、ハイフォン無しの"`git foo`" をコマンドラインに入力するとこの `git-foo` が起動されます。これは `git` コマンドの機能です。

ティップ
tkcvs(1) や gitk(1) 等の GUI ツールはファイルの変更履歴を追跡するのに本当に役立ちます。多くの公開アーカイブが彼らのレポジトリーの中を閲覧するための提供しているウェッブインターフェースもまた非常に有用です。

ティップ
Git は `git-cvs` や `git-svn` パッケージを使って CVS や Subversion によって提供されるレポジトリー等の他の VCS レポジトリーを直接処理したり、ローカル変更のためのローカルレポジトリーを提供します。git for CVS users や項10.6.4を参照下さい。

ティップ
CVS や Subversion に等価コマンドが無いコマンドが Git にあります: ”fetch”、”rebase”、”cherry-pick”、…

10.6 Git

Git はローカルとリモートのソースコード管理の全機能があります。これはリモートレポジトリーとのネットワーク接続なしにソースコードへの変更を記録できるといことです。

10.6.1 Git クライアントの設定

Git は使うあなたの名前や email アドレス等を”~/.gitconfig” 中のいくつかのグローバル設定に設定したいなら次のようにします。

```
$ git config --global user.name ”Name Surname”
$ git config --global user.email yourname@example.com
```

もしあなたが CVS や Subversion コマンドに慣れ過ぎている場合には、いくつかのコマンドエリアスの設定を次のようにするのも一計です。

```
$ git config --global alias.ci ”commit -a”
$ git config --global alias.co checkout
```

あなたのグローバル設定は次のようにするとチェックできます。

```
$ git config --global --list
```

10.6.2 Git リファレンス

次を参照下さい。

- マンページ: git(1) (/usr/share/doc/git-doc/git.html)
- Git ユーザーマニュアル (/usr/share/doc/git-doc/user-manual.html)
- git へのチュートリアル導入 (/usr/share/doc/git-doc/gittutorial.html)
- git へのチュートリアル導入: 第 2 部 (/usr/share/doc/git-doc/gittutorial-2.html)
- 約 20 のコマンドを使って毎日 GIT (/usr/share/doc/git-doc/everyday.html)
- CVS ユーザーのための git (/usr/share/doc/git-doc/gitcvs-migration.html)
  - これは CVS 同様のサーバーのセットアップ法や CVS から Git への古いデーターの抽出法も説明します。
- ウェッブ上で利用可能な他の Git に関するリソース
  - Git - SVN Crash Course
  - Git マジック (/usr/share/doc/gitmagic/html/index.html)

git-gui(1) と gitk(1) コマンドは簡単に Git が利用出来るようにします。

警告
たとえ gitk(1) 等の一部ツールが受け付けるからといって、タグ文字列中にスペースを使ってはいけません。他の git コマンドで支障が起こるかもしれません。

10.6.3 Git コマンド

アップストリームが異なる VCS を使っていようと、アップストリームへのネットワーク接続無しにローカルコピーを管理できるので、ローカル活動に git(1) を使うのは良い考えかもしれません。次は git(1) とともに使われるパッケージとコマンドのリストです。

パッケージ	ポプコン	サイズ	コマンド	説明
git-doc	I:15	11762	N/A	正式 Git 文書
gitmagic	I:1	721	N/A	"Git マジック"、Git に関する分かり易いガイド
git	V:305, I:478	35040	git(7)	git: 高速、スケーラブル、分散型リビジョンコントロールシステム
gitk	V:6, I:42	1723	gitk(1)	GUI による履歴付き Git レポジトリーブラウザー
git-gui	V:2, I:24	2317	git-gui(1)	Git 用の GUI (履歴無し)
git-svn	V:1, I:22	1144	git-svnimport(1)	Subversion から Git へのデーターのインポート
git-svn	V:1, I:22	1144	git-svn(1)	Subversion と Git 間での双方向操作を提供
git-cvs	V:0, I:10	1279	git-cvsimport(1)	CVS から Git へのデーターのインポート
git-cvs	V:0, I:10	1279	git-cvsexportcommit(1)	Git から CVS チェックアウトに 1 コミットエキスポート
git-cvs	V:0, I:10	1279	git-cvsserver(1)	Git 用 CVS サーバーエミュレーター
git-email	V:0, I:11	966	git-send-email(1)	Git からパッチの集合の email として送信
stgit	V:0, I:0	603	stg(1)	Git 上の quilt (Python)
git-buildpackage	V:2, I:12	4193	git-buildpackage(1)	Git を使って Debian パッケージ化を自動化
guilt	V:0, I:0	146	guilt(7)	Git 上の quilt (SH/AWK/SED/…)

Table 10.13: git 関連のパッケージとコマンドのリスト

ティップ
git(1) では、多くのコミットをしながらローカルブランチ上で仕事をして"`git rebase -i master`" 等を使って後から変更履歴を整理します。こうすると綺麗な変更履歴が作れます。git-rebase(1) と git-cherry-pick(1) を参照下さい。

ティップ
作業中のディレクトリーの現状を失わずにクリーンな作業ディレクトリを取り戻すには"`git stash`" を使えばできます。git-stash(1) を参照ください。

10.6.4 Subversion レポジトリー用の Git

"`svn+ssh://svn.example.org/project/module/trunk`" にある Subversion のレポジトリを"`./dest`" にあるローカルの Git レポジトリにチェックアウトでき、また Subversion レポジトリに戻すコミットができます。例えば:

```
$ git svn clone -s -rHEAD svn+ssh://svn.example.org/project dest
$ cd dest
... make changes
$ git commit -a
... keep working locally with git
$ git svn dcommit
```

ティップ
”-rHEAD” を使うことにより Subversion のレポジトリから過去の全コンテントを複製することが避けられます。

10.6.5 設定履歴記録のための Git

Git ツールを使い設定の経時履歴をマニュアルで記録できます。次は”/etc/apt/” の内容を記録する単純な練習例です。

```
$ cd /etc/apt/
$ sudo git init
$ sudo chmod 700 .git
$ sudo git add .
$ sudo git commit -a
```

設定を説明とともにコミットします。

設定ファイルへの変更をします。

```
$ cd /etc/apt/
$ sudo git commit -a
```

設定を説明とともにコミットしそのままいろいろ作業をしていきます。

```
$ cd /etc/apt/
$ sudo gitk --all
```

フルの履歴を手中しています。

注意
設定データーのどのファイルパーミッションでも機能するためには sudo(8) が必要です。ユーザーの設定データーの場合 sudo をスキップ出来るかもしれません。

注意
上記例での”chmod 700 .git” コマンドはアーカイブデーターを不正読出しアクセスから守るために必要です。

ティップ
設定履歴の記録に関するより完全なセットアップは、etckeeper パッケージを見て下さい: 項9.2.10。

10.7 CVS

CVS は Subversion や Git より古いバージョン管理システムです。

注意
以下の CVS のための例中にある多くの URL は既に存在しません。

次を参照下さい。

- cvs(1)
- ”/usr/share/doc/cvs/html-cvsclient”
- ”/usr/share/doc/cvs/html-info”
- ”/usr/share/doc/cvsbook”
- ”info cvs”

10.7.1 CVS レポジトリーの設定

CVS レポジトリーへのコミットを”src” グループメンバに限定し許可し、CVS の管理を”staff” グループメンバに限定し許可するように次の設定をすることでうっかりミスを予防できます。

```
# cd /var/lib; umask 002; mkdir cvs
# export CVSROOT=/srv/cvs/project
# cd $CVSROOT
# chown root:src .
# chmod 2775 .
# cvs -d $CVSROOT init
# cd CVSROOT
# chown -R root:staff .
# chmod 2775 .
# touch val-tags
# chmod 664 history val-tags
# chown root:src history val-tags
```

ティップ
”$CVSROOT” ディレクトリーの所有者を”root:staff” とし、そのパーミッションを”3775” と変更すると新規プロジェクトの作成を制限できます。

10.7.2 CVS へのローカルアクセス

デフォールトの CVS レポジトリーは”$CVSROOT” で指示されます。次はローカルアクセスのための”$CVSROOT” を設定します。

```
$ export CVSROOT=/srv/cvs/project
```

10.7.3 pserver を使った CVS へのリモートアクセス

多くの公開 CVS サーバーはアカウント名”anonymous” で pserver サービス経由の読出しのみアクセスを提供します。例えば、Debian ウェッブサイト内容は webwml プロジェクトは Debian alioth サービスでの CVS を使ってメンテされていました。以下はこの CVS レポジトリーへのリモートアクセスのための”$CVSROOT” を設定していました。

```
$ export CVSROOT=:pserver:anonymous@anonscm.debian.org:/cvs/webwml
$ cvs login
```

注意
pserver は盗聴攻撃されやすくインセキュアーなので、書込みアクセスはサーバーの管理者によって通常無効にされています。

10.7.4 ssh を使った CVS へのリモートアクセス

SSH を使い webwml プロジェクトの CVS レポジトリーにリモートアクセスためには、以下のように”$CVS_RSH” と”$CVSROOT” を設定していました。

```
$ export CVS_RSH=ssh
$ export CVSROOT=:ext:account@cvs.alioth.debian.org:/cvs/webwml
```

リモートパスワードプロンプトを無くすのに SSH を使ったパブリックキー認証を使えます。

10.7.5 新規ソースを CVS にインポート

”~/path/to/module1” で新規のローカルソースツリーの場所を作成。

```
$ mkdir -p ~/path/to/module1; cd ~/path/to/module1
```

”~/path/to/module1” の下の新規のローカルソースツリーを埋める。

次のパラメーターを使って CVS にインポートします。

- モジュール名: ”module1”
- ベンダータグ: Main-branch (全ブランチへのタグ)
- リリースタグ: Release-initial (特定リリースタグ)

```
$ cd ~/path/to/module1
$ cvs import -m ”Start module1” module1 Main-branch Release-initial
$ rm -Rf . # optional
```

10.7.6 CVS レポジトリーのファイルパーミッション

CVS は現レポジトリーファイルを上書きせず他のファイルで置き換えます。だからレポジトリーディレクトリーへの書込みパーミッションはクリチカルです。”module1” という新規レポジトリーを”/srv/cvs/project” に作成の際毎に、必要ならこの条件を満たすために次を実行します。

```
# cd /srv/cvs/project
# chown -R root:src module1
# chmod -R ug+rwX   module1
# chmod    2775     module1
```

10.7.7 CVS のワークフロー

次に CVS を使う典型的なワークフローを記します。

”$CVSROOT” で指される CVS プロジェクトから得られる全てのモジュールを次のようにしてチェックします。

```
$ cvs rls
CVSROOT
module1
module2
...
```

”module1” をそのデフォールトディレクトリーの”./module1” に次のようにしてチェックアウトします。

```
$ cd ~/path/to
$ cvs co module1
$ cd module1
```

必要に応じ内容に変更を加える。

次のようにして”diff -u [repository] [local]” と等価を作成し変更をチェックします。

```
$ cvs diff -u
```

あるファイル”file_to_undo” をひどく壊したが他のファイルは大丈夫な事に気づきます。

次のようにして”file_to_undo” ファイルを CVS からのクリーンコピーで上書きします。

```
$ cvs up -C file_to_undo
```

次のようにして更新されたローカルソースツリーを CVS に保存します。

```
$ cvs ci -m ”Describe change”
```

次のようにして”file_to_add” ファイルを作成し CVS に追加します。

```
$ vi file_to_add
$ cvs add file_to_add
$ cvs ci -m ”Added file_to_add”
```

次のようにして CVS から最新バージョンをマージします。

```
$ cvs up -d
```

コンフリクトある変更を意味する”C filename” で始まる行に注意します。

”.#filename.version” 中の変更されていないコードを探します。

コンフリクトある変更はファイル中の”<<<<<<<” や”>>>>>>>” を探索します。

必要に応じコンフリクト解消のためにファイルを編集します。

次のようにしてリリースタグ”Release-1” を追加します。

```
$ cvs ci -m ”last commit for Release-1”
$ cvs tag Release-1
```

さらに編集します。

次のようにリリースタグ”Release-1” を削除します。

```
$ cvs tag -d Release-1
```

次のように CVS へ変更をチェックインします。

```
$ cvs ci -m ”real last commit for Release-1”
```

リリースタグ”Release-1” を更新した CVS の main の HEAD に次のようにして再付加します。

```
$ cvs tag Release-1
```

次のようにして”Release-initial” タグによってで指定されるオリジナルのバージョンから、スティッキーブランチタグが”Release-initial-bugfixes” のブランチを作成し、”~/path/to/old” ディレクトリーにチェックアウトします。

```
$ cvs rtag -b -r Release-initial Release-initial-bugfixes module1
$ cd ~/path/to
$ cvs co -r Release-initial-bugfixes -d old module1
$ cd old
```

ティップ
ブランチポイントとして特定の日付を指定するには”-r Release-initial” の代わりに”-D 2005-12-20” (ISO 8601 日付フォーマット) を使います。

オリジナルバージョンに基づいたスティッキータグ”Release-initial-bugfixes” が付いているこのローカルのソースツリーで作業をします。

このブランチで一人で作業をします…誰か他の人がこの”Release-initial-bugfixes” ブランチに合流するまで。

次のようにして必要に応じて新規ディレクトリーを作りながらこのブランチ上の他の人が変更したファイルと同期します。

```
$ cvs up -d
```

必要に応じコンフリクト解消のためにファイルを編集します。

次のように CVS へ変更をチェックインします。

```
$ cvs ci -m ”checked into this branch”
```

次のようにしてスティッキータグを削除し (”-A”)、キーワード展開せずに (”-kk”)、main の HEAD (最新版) をつかってローカルのツリーを更新します。

```
$ cvs up -d -kk -A
```

次のようにしてキーワード展開せずに (”-kk”)、”Release-initial-bugfixes” ブランチをマージしてローカルのツリー (内容 = main の HEAD) を更新します。

```
$ cvs up -d -kk -j Release-initial-bugfixes
```

エディターを使ってコンフリクト解消します。

次のように CVS へ変更をチェックインします。

```
$ cvs ci -m ”merged Release-initial-bugfixes”
```

次のようにしてアーカイブを作成します。

```
$ cd ..
$ mv old old-module1-bugfixes
$ tar -cvzf old-module1-bugfixes.tar.gz old-module1-bugfixes
$ rm -rf old-module1-bugfixes
```

ティップ
”cvs up” コマンドには、新規ディレクトリー作成には”-d” オプションを、また空ディレクトリーの摘み取りには”-P” オプションを指定できます。

ティップ
”cvs co module1/subdir” とサブディレクトリーの名前を提供して、”module1” のサブディレクトリーだけをチェックアウトすることが出来ます。

10.7.8 CVS から最新ファイル

CVS から最新ファイルを取り込むには、次のように”tomorrow” (明日) を使います:

```
$ cvs ex -D tomorrow module_name
```

オプション	意味
-n	空実行、効果無し
-t	各段階の CVS 活動が分かるようにメッセージを表示

Table 10.14: CVS コマンドの特記すべきオプション (cvs(1) の最初の引数として使用)

10.7.9 CVS の管理運営

モジュールのエリアス”mx” を CVS プロジェクト (ローカルサーバー) に追加します。

```
$ export CVSROOT=/srv/cvs/project
$ cvs co CVSROOT/modules
$ cd CVSROOT
$ echo ”mx -a module1” >>modules
$ cvs ci -m ”Now mx is an alias for module1”
$ cvs release -d .
```

さて、CVS から”new” ディレクトリーに”module1” (エリアス: ”mx”) を次のようにしてチェックアウトします。

```
$ cvs co -d new mx
$ cd new
```

注意
上記プロシージャを行うには、適切なファイルパーミッションが設定されているべきです。

10.7.10 CVS チェックアウトの実行ビット

CVS からファイルをチェックアウトする時に、その実行パーミッションビットは保持されます。

チェックアウトされた例えば”filename” ファイルで実行ビットの問題にあった際には、次のコマンドを使い CVS レポジトリー中のそのパーミッションを変更します。

```
# chmod ugo-x filename
```

10.8 Subversion

Subversion は Git 以前で CVS 以降の少し古いバージョン管理システムです。Subversion にはタグとブランチを除く CVS と Git のほとんどの機能をあります。

Subversion サーバーをセットアップするには、subversion と libapache2-mod-svn と subversion-tools パッケージをインストールする必要があります。

10.8.1 Subversion レポジトリーの設定

現状の subversion パッケージはレポジトリーの設定はしないので手動で設定を行う必要があります。レポジトリーを置く可能な場所の 1 つは”/srv/svn/project” 中です。

ディレクトリーを次のように作成します。

```
# mkdir -p        /srv/svn/project
```

レポジトリーデーターベースを次のように作成します。

```
# svnadmin create /srv/svn/project
```

10.8.2 Apach2 サーバーの経由の Subversion アクセス

Apache2 サーバー経由で Subversion レポジトリーにアクセスするだけなら、レポジトリーを次に記すように WWW サーバーのみによって書込み可とするだけが必要です。

```
# chown -R www-data:www-data /srv/svn/project
```

ユーザー認証経由でレポジトリーにアクセス許可するために”/etc/apache2/mods-available/dav_svn.conf” 中の次の部分を追加 (もしくはアンコメント) します。

```
<Location /project>
  DAV svn
  SVNPath /srv/svn/project
  AuthType Basic
  AuthName ”Subversion repository”
  AuthUserFile /etc/subversion/passwd
<LimitExcept GET PROPFIND OPTIONS REPORT>
    Require valid-user
</LimitExcept>
</Location>
```

次のコマンドでユーザー認証ファイルを作成します。

```
# htpasswd2 -c /etc/subversion/passwd some-username
```

Apach2 を再スタート

あなたの新規の Subversion レポジトリーは”http://localhost/project” と (ウェッブサーバーの URL が”http://ex
と仮定すると) ”http://example.com/project” の URL にてアクセスできます。

10.8.3 グループによる Subversion へのローカルアクセス

次は Subversion レポジトリーを例えば project というグループによってローカルアクセス出きるようにする設定です。

```
# chmod  2775     /srv/svn/project
# chown -R root:src /srv/svn/project
# chmod -R ug+rwX   /srv/svn/project
```

あなたの新しい Subversion レポジトリーは、project グループに属するローカルユーザーにとって svn(1) から”file:///localhost/srv/svn/project” か”file:///srv/svn/project” の URL からグループアクセスできます。グループアクセスを確実にするために、svn や svnserve や svnlook や svnadmin 等のコマンドを”umask 002” の下で実行しなければいけません。

10.8.4 グループによる Subversion への SSH 経由のリモートアクセス

SSH から”example.com:/srv/svn/project” の URL にあるグループアクセス可能な Subversion レポジトリーは、svn(1) を使って”svn+ssh://example.com:/srv/svn/project” にてアクセスできます。

10.8.5 Subversion ディレクトリー構造

Subversion のブランチやタグの機能を欠くことを補うべく、多くのプロジェクトは次と似たようなディレクトリーツリーを使っています。

```
----- module1
  |   |-- branches
  |   |-- tags
  |   |   |-- release-1.0
  |   |   `-- release-2.0
  |   |
  |   `-- trunk
  |       |-- file1
  |       |-- file2
  |       `-- file3
  |
  `-- module2
```

ティップ
ブランチやタグをマークするためには”svn copy …” コマンドを使わなければいけません。こうすることで Subversion がファイルの変更履歴を適正な記録を確実にするとともに記録容量を節約できます。

10.8.6 新規ソースを Subversion にインポート

”~/path/to/module1” で新規のローカルソースツリーの場所を作成。

```
$ mkdir -p ~/path/to/module1; cd ~/path/to/module1
```

”~/path/to/module1” の下の新規のローカルソースツリーを埋める。

ソースを次のパラメーターを使って Subversion にインポートします。

- モジュール名: ”module1”
- Subversion サイトの URL: ”file:///srv/svn/project”,
- Subversion ディレクトリー: ”module1/trunk”:
- Subversion タグ: ”module1/tags/Release-initial”

```
$ cd ~/path/to/module1
$ svn import file:///srv/svn/project/module1/trunk -m ”Start module1”
$ svn cp file:///srv/svn/project/module1/trunk file:///srv/svn/project/module1/tags/Release ↩
    -initial
```

この代わりに、次のようにも出来ます。

```
$ svn import ~/path/to/module1 file:///srv/svn/project/module1/trunk -m ”Start module1”
$ svn cp file:///srv/svn/project/module1/trunk file:///srv/svn/project/module1/tags/Release ↩
    -initial
```

ティップ
”file:///…” といった URL は、”http://…” や”svn+ssh://…” といったいかなる他形式の URL ででも置き換えられます。

10.8.7 Subversion のワークフロー

次に Subversion をその本来のクライアントとともに使う典型的なワークフローを記します。

ティップ
`git-svn` パッケージによって提供されるクライアントコマンドは `git` コマンドを使う Subversion の代替ワークフローを提供します。項10.6.4を参照下さい。

”`file:///srv/svn/project`” の URL で指定される Subversion プロジェクトから得られる全ての利用可能なモジュールを次のようにしてチェックします。

```
$ svn list file:///srv/svn/project
module1
module2
...
```

次のようにして”`module1/trunk`” を”`module1`” ディレクトリーにチェックアウトします。

```
$ cd ~/path/to
$ svn co file:///srv/svn/project/module1/trunk module1
$ cd module1
```

必要に応じ内容に変更を加える。

次のようにして”`diff -u [repository] [local]`” と等価を作成し変更をチェックします。

```
$ svn diff
```

あるファイル”`file_to_undo`” をひどく壊したが他のファイルは大丈夫な事に気づきます。

次のようにして”`file_to_undo`” ファイルを Subversion からのクリーンコピーで上書きします。

```
$ svn revert file_to_undo
```

更新されたローカルソースツリーを次のようにして Subversion に保存します。

```
$ svn ci -m ”Describe change”
```

次のようにして”`file_to_add`” ファイルを作成し Subversion に追加します。

```
$ vi file_to_add
$ svn add file_to_add
$ svn ci -m ”Added file_to_add”
```

次のようにして Subversion から最新バージョンをマージします。

```
$ svn up
```

コンフリクトある変更を意味する”`C filename`” で始まる行に注意します。

”`filename.r6`” と”`filename.r9`” と”`filename.mine`” 等中の変更されていないコードを探します。

コンフリクトある変更はファイル中の”`<<<<<<<`” や”`>>>>>>>`” を探索します。

必要に応じコンフリクト解消のためにファイルを編集します。

次のようにしてリリースタグ”`Release-1`” を追加します。

```
$ svn ci -m ”last commit for Release-1”
$ svn cp file:///srv/svn/project/module1/trunk file:///srv/svn/project/module1/tags/Release ↩
    -1
```

さらに編集します。

次のようにリリースタグ”Release-1” を削除します。

```
$ svn rm file:///srv/svn/project/module1/tags/Release-1
```

次のようにして変更を Subversion にチェックインします。

```
$ svn ci -m ”real last commit for Release-1”
```

リリースタグ”Release-1” を更新した Subversion の trunk の HEAD に再付加します。

```
$ svn cp file:///srv/svn/project/module1/trunk file:///srv/svn/project/module1/tags/Release ↩
    -1
```

次のようにして”module1/tags/Release-initial” というパスで指定されるオリジナルのバージョンから、パスが”module1/branches/Release-initial-bugfixes” のブランチを作成し、”~/path/to/old” ディレクトリーにチェックアウトします。

```
$ svn cp file:///srv/svn/project/module1/tags/Release-initial file:///srv/svn/project/ ↩
    module1/branches/Release-initial-bugfixes
$ cd ~/path/to
$ svn co file:///srv/svn/project/module1/branches/Release-initial-bugfixes old
$ cd old
```

ティップ
ブランチポイントとして特定の日付を指定するには”module1/tags/Release-initial” の代わりに”module1/trunk@{2005-12-20}” (ISO 8601 日付フォーマット) を使います。

オリジナルバージョンに基づいたブランチ”Release-initial-bugfixes” を指定しているローカルのソースツリーで作業します。

このブランチで一人で作業をします…誰か他の人がこの”Release-initial-bugfixes” ブランチに合流するまで。

次のようにしてこのブランチ上の他の人が変更したファイルと同期します。

```
$ svn up
```

必要に応じコンフリクト解消のためにファイルを編集します。

次のようにして変更を Subversion にチェックインします。

```
$ svn ci -m ”checked into this branch”
```

trunk の HEAD を使って次のようにローカルツリーを更新します。

```
$ svn switch file:///srv/svn/project/module1/trunk
```

次のようにして”Release-initial-bugfixes” ブランチをマージしてローカルのツリー (内容 = trunk の HEAD) を更新します。

```
$ svn merge file:///srv/svn/project/module1/branches/Release-initial-bugfixes
```

エディターを使ってコンフリクト解消します。

次のようにして変更を Subversion にチェックインします。

```
$ svn ci -m ”merged Release-initial-bugfixes”
```

次のようにしてアーカイブを作成します。

```
$ cd ..
$ mv old old-module1-bugfixes
$ tar -cvzf old-module1-bugfixes.tar.gz old-module1-bugfixes
$ rm -rf old-module1-bugfixes
```

ティップ
”file:///…” といった URL は、”http://…” や”svn+ssh://…” といったいかなる他形式の URL ででも置き換えられます。

ティップ
”svn co file:///srv/svn/project/module1/trunk/subdir module1/subdir” とサブディレクトリーの名前を提供して、”module1” のサブディレクトリーだけをチェックアウトすることが出来ます。

オプション	意味
--dry-run	空実行、効果無し
-v	svn 活動の詳細なメッセージを表示

Table 10.15: Subversion コマンドの特記すべきオプション (svn(1) の最初の引数として使用)

Chapter 11

データー変換

Debian システム上のデーターフォーマット変換のツールとティップを記します。

標準に準拠したツールは非常に良い状態ですが、プロプライエタリデーターフォーマットのサポートは限定的です。

11.1 テキストデーター変換ツール

テキストデーター変換のための次のパッケージが著者の目に止まりました。

パッケージ	ポプコン	サイズ	キーワード	説明
libc6	V:935, I:999	12771	文字セット	iconv(1) によるロケール間のテキスト符号化方式変換ソフト (基本的)
recode	V:3, I:25	603	文字セット + 行末文字	ロケール間のテキスト符号化方式変換ソフト (機能豊富、より多いエリアスと機能)
konwert	V:1, I:54	134	文字セット	ロケール間のテキスト符号化方式変換ソフト (高級機能)
nkf	V:0, I:11	358	文字セット	日本語のための文字セット翻訳ソフト
tcs	V:0, I:0	518	文字セット	文字セット翻訳ソフト
unaccent	V:0, I:0	29	文字セット	アクセント付き文字をアクセントの無しの等価文字に置換
tofrodos	V:1, I:25	55	行末文字	DOS と Unix 間のテキストフォーマット変換ソフト: fromdos(1) と todos(1)
macutils	V:0, I:1	298	行末文字	Macintosh と Unix 間のテキストフォーマット変換ソフト: frommac(1) and tomac(1)

Table 11.1: テキストデーター変換ツールのリスト

11.1.1 テキストファイルを iconv を使って変換

ティップ
iconv(1) は libc6 パッケージの一部として提供されていて、文字の符号化方式変換のために実質的に全ての Unix 的システムで常に利用可能です。

次のようにするとテキストファイルを iconv(1) を使って変換できます。

```
$ iconv -f encoding1 -t encoding2 input.txt >output.txt
```

符号化方式 (エンコーディング) 値をマッチングする際には、大文字小文字の区別は無く、”-” や”_” を無視します。”iconv -l” コマンドにより、サポートされている符号化方法が確認できます。

符号化方式値	使い方
ASCII	情報交換用米国標準コード (ASCII); アクセント文字無しの 7 ビットコード
UTF-8	全現代的 OS のための現行多言語標準
ISO-8859-1	西欧州言語用の旧標準、ASCII + アクセント文字
ISO-8859-2	東欧州言語用の旧標準、ASCII + アクセント文字
ISO-8859-15	西欧州言語用の旧標準、ユーロ文字付き ISO-8859-1
CP850	コードページ 850、西欧州言語用グラフィック文字付き Microsoft DOS 文字、ISO-8859-1 の変種
CP932	コードページ 932、日本語用 Microsoft Windows スタイル Shift-JIS の変種
CP936	コードページ 936、簡体中国語用 Microsoft Windows スタイル GB2312 か GBK か GB18030 の変種
CP949	コードページ 949、韓国語用 Microsoft Windows スタイル EUC-KR か統一ハングルコードの変種
CP950	コードページ 950、繁体中国語用 Microsoft Windows スタイル Big5 の変種
CP1251	コードページ 1251、キリル文字用 Microsoft Windows スタイル符号化方式
CP1252	コードページ 1252、西欧州言語用 Microsoft Windows スタイル ISO-8859-15 の変種
KOI8-R	キリル文字用の旧ロシアの UNIX 標準
ISO-2022-JP	7 ビットコードのみを用いる日本語 email の標準符号化方式
eucJP	Shift-JIS とはまったく違う、旧日本の UNIX 標準 8 ビットコード
Shift-JIS	日本語のための JIS X 0208 Appendix 1 標準 (CP932 参照下さい)

Table 11.2: 符号化方式値とその使い方リスト

注意
一部の符号化方式 (エンコーディング) はデーター変換のみサポートされており、ロケール値としては使われません (項8.4.1参照下さい)。

ASCII や ISO-8859 文字セットのような 1 バイトに収まる文字セットに付いては、文字の符号化方式 (エンコーディング) とは文字セットとほとんど同じ事を意味します。

日本語のための JIS X 0213 や実質的に全ての言語のためのユニコード文字セット (UCS, Unicode, ISO-10646-1) のような多くの文字を含む文字セットの場合には、バイトデーター列に落とし込む多くの符号化手法があります。

- 日本語用には、EUC と ISO/IEC 2022 (別名 JIS X 0202)
- ユニコード用には、UTF-8 と UTF-16/UCS-2 と UTF-32/UCS-4

これらに関しては、文字セットと文字符号化方式の間にはっきりとした区別があります。

コードページは、一部のベンダー固有のコードページで文字符号化テーブルと同義語として使用されています。

注意
ほとんどの符号化システムが 7 ビット文字に関して ASCII と同じコードを共有している事を覚えておいて下さい。もちろん例外はありますが。もし古い日本語の C プログラムや URL のデーターをカジュアルにシフト JIS と呼ばれている符号化フォーマットから UTF-8 フォーマットに変換する際には、期待される結果を得るために”shift-JIS” ではなく”CP932” を使いましょう: 0x5C → ”\” と 0x7E → ”~”。こうしないと、これらが間違った文字に変換されます。

ティップ
recode(1) は、十分使えますし、iconv(1) と fromdos(1) と todos(1) と frommac(1) と tomac(1) を組み合わせ以上の機能を提供します。詳しくは”info recode” を参照下さい。

11.1.2 ファイルが UTF-8 であると iconv を使い確認

次のようにするとテキストファイルが UTF-8 でエンコードされていると iconv(1) を使って確認できます。

```
$ iconv -f utf8 -t utf8 input.txt >/dev/null || echo ”non-UTF-8 found”
```

ティップ
最初の非 UTF-8 文字を見つけるには上記例中で”--verbose” オプションを使います。

11.1.3 iconv を使ってファイル名変換

次に、単一ディレクトリー中の旧 OS 下で作成されたファイル名から現代的な UTF-8 のファイル名に符号化方式を変換するスクリプト例を示します。

```
#!/bin/sh
ENCDN=iso-8859-1
for x in *;
 do
 mv ”$x” ”$(echo ”$x” | iconv -f $ENCDN -t utf-8)”
done
```

”$ENCDN” 変数値には、旧 OS 下で用いられたファイル名に用いられた元となる表 11.2中にあるエンコーディングを指定します。

もっと複雑な場合にはそのようなファイル名を含有するファイルシステム (ディスクドライブ上のパーティション等) を mount(8) オプションに適正な符号化方式 (エンコーディング) (項8.4.6参照下さい) を指定してマウントし、その全内容を他の UTF-8 でマウントされたファイルシステムに”cp -a” コマンドを使ってコピーします。

11.1.4 行末変換

テキストファイルのフォーマット、特に行末 (EOL) コード、はプラットフォーム依存です。

行末 (EOL) フォーマット変換プログラムに関して、fromdos(1) と todos(1) と frommac(1) と tomac(1) は非常に便利です。recode(1) もまた役に立ちます。

注意
python-moinmoin パッケージ用の wiki のデーター等の Debian システム上の一部データーは、MSDOS スタイルの CR-LF を行末コードとして用います。あくまで上記は一般則と言うだけです。

プラットフォーム	行末コード	コントロール	**10** 進数	**16** 進数
Debian (unix)	LF	^J	10	0A
MSDOS と Windows	CR-LF	^M^J	13 10	0D 0A
Apple の Macintosh	CR	^M	13	0D

Table 11.3: 異なるプラットフォーム上での行末スタイルのリスト

注意
ほとんどのエディター (例えば vim や emacs や gedit 等) は MSDOS スタイルの行末を透過的に取り扱えます。

ティップ
MSDOS と Unix スタイルが混在する行末スタイルを MSDOS スタイルに統一するには、todos(1) を使う代わりに”sed -e '/\r$/!s/$/\r/'” を使う方がより好ましいです。(例えば、2 つの MSDOS スタイルファイルを diff3(1) を使ってマージした後。) todos は全ての行に CR を追加するというのがこの理由です。

11.1.5 タブ変換

タブコードを変換するための良く使われる専用プログラムがいくつかあります。

機能	**bsdmainutils**	**coreutils**
タブからスペースに展開する	”col -x”	expand
スペースからタブに逆展開する	”col -h”	unexpand

Table 11.4: bsdmainutils と coreutils パッケージ中のタブ変換コマンドのリスト

indent パッケージにある indent(1) コマンドは C プログラム中のホワイトスペースを完全にリフォーマットします。

vim や emacs 等のエディタープログラムもまたタブ変換に使えます。例えば vim を使うと、”:set expandtab” として”:%retab” するコマンドシーケンスでタブ変換が出来ます。これを元に戻すのは、”:set noexpandtab” として”:%retab!” とするコマンドシーケンスです。

11.1.6 自動変換付きエディター

vim プログラムなどのインテリジェントな現代的なエディターは大変良く出来ていていかなる符号化方式やいかなるファイルフォーマットでも機能します。これらのエディターを UTF-8 ロケール下で UTF-8 を扱えるコンソール中で使用することで最良の互換性が得られます。

latin1 (iso-8859-1) 符号化方式で保存された古い西欧州の Unix テキストファイル”u-file.txt” は、単純に vim を使って次のようにして編集出来ます。

```
$ vim u-file.txt
```

vim 中の符号化方式自動判定機構が、最初は UTF-8 符号化方式を仮定し、それが上手く行かなかった際に latin1 を仮定するから可能です。

latin2 (iso-8859-2) 符号化方式で保存された古いポーランドの Unix テキストファイル”pu-file.txt” は、vim を使って次のようにして編集出来ます。

```
$ vim '+e ++enc=latin2 pu-file.txt'
```

eucJP 符号化方式で保存された古い日本の Unix テキストファイル"ju-file.txt" は、vim を使って次のようにして編集出来ます。

```
$ vim '+e ++enc=eucJP ju-file.txt'
```

shift-JIS 符号化方式 (より正確には: CP932) で保存された古い日本の MS-Windows テキストファイル"jw-file.txt" は、vim を使って次のようにして編集出来ます。

```
$ vim '+e ++enc=CP932 ++ff=dos jw-file.txt'
```

"++enc" や"++ff" オプションを使ってファイルが開かれた時は、Vim コマンドライン中の":w" がオリジナルのファイルフォーマットでオリジナルのファイルを上書きします。例えば":w ++enc=utf8 new.txt" 等と Vim コマンドライン中で保存フォーマットや保存ファイル名を指定することも出来ます。

vim オンラインヘルプ中の mbyte.txt "multi-byte text support" と、"++enc" に使われるロケール値に関する表 11.2を参照下さい。

emacs ファミリーのプログラムもまた同様の機能の実行ができます。

11.1.7 プレーンテキスト抽出

以下はウェッブページを読みテキストファイルに落とします。ウェッブから設定を取ってくる時や grep(1) 等の基本的な Unix テキストツールをウェッブページに適用するときに非常に有用です。

```
$ w3m -dump http://www.remote-site.com/help-info.html >textfile
```

同様に、次を用いることで他のフォーマットからプレーンテキストデーターを抽出出来ます。

パッケージ	ポプコン	サイズ	キーワード	機能
w3m	V:31, I:284	2289	html→text	"w3m -dump" コマンドを使う HTML からテキストへの変換ソフト
html2text	V:3, I:33	274	html→text	先進的 HTML からテキストへの変換ソフト (ISO 8859-1)
lynx	V:13, I:98	1948	html→text	"lynx -dump" コマンドを使う HTML からテキストへの変換ソフト
elinks	V:6, I:28	1767	html→text	"elinks -dump" コマンドを使う HTML からテキストへの変換ソフト
links	V:6, I:39	2249	html→text	"links -dump" コマンドを使う HTML からテキストへの変換ソフト
links2	V:1, I:15	5417	html→text	"links2 -dump" コマンドを使う HTML からテキストへの変換ソフト
antiword	V:2, I:10	589	MSWord→text,ps	MSWord ファイルをプレーンテキストか ps に変換
catdoc	V:27, I:127	675	MSWord→text,TeX	MSWord ファイルをプレーンテキストか TeX に変換
pstotext	V:1, I:3	126	ps/pdf→text	PostScript と PDF ファイルからテキストを抽出
unhtml	V:0, I:0	43	html→text	HTML ファイルからマークアップタグを削除
odt2txt	V:1, I:7	60	odt→text	OpenDocument テキストからテキストへの変換ソフト

Table 11.5: プレーンテキストデーター抽出ツールのリスト

11.1.8 プレーンテキストデーターをハイライトとフォーマット

次のようにしてプレーンテキストデーターをハイライトとフォーマット出来ます。

パッケージ	ポプコン	サイズ	キーワード	説明
vim-runtime	V:19, I:435	31723	ハイライト	":source $VIMRUNTIME/syntax/html.vim" を使ってソースコードを HTML に変換するための Vim MACRO
cxref	V:0, I:0	1193	c → html	C プログラムから latex か HTML への変換ソフト (C 言語)
src2tex	V:0, I:0	622	ハイライト	多くのソースコードの TeX への変換ソフト (C 言語)
source-highlight	V:0, I:7	1992	ハイライト	多くのソースコードを HTML と XHTML と LaTeX と Texinfo と ANSI カラーエスケープシーケンスと DocBook にハイライト付きで変換 (C++)
highlight	V:1, I:12	1083	ハイライト	多くのソースコードを HTML と XHTML と LaTeX と Tex と AXSL-FO にハイライト付きで変換 (C++)
grc	V:0, I:3	190	text → color	汎用着色化ソフト (Python)
txt2html	V:0, I:3	259	text → html	テキストから HTML への変換ソフト (Perl)
markdown	V:0, I:8	57	text → html	markdown テキスト文書の (X)HTML へのフォーマット化ソフト (Perl)
asciidoc	I:13	81	text → any	AsciiDoc テキスト文書の XML/HTML へのフォーマット化ソフト (Python)
pandoc	V:8, I:47	151714	text → any	汎用マークアップコンバーター (Haskel)
python-docutils	V:12, I:133	1771	text → any	ReStructured テキスト文書の XML へのフォーマット化ソフト (Python)
txt2tags	V:0, I:1	342	text → any	テキストから HTML と SGML と LaTeX と man page と MoinMoin と Magic Point と PageMaker への文書変換 (Python)
udo	V:0, I:0	583	text → any	汎用文書 - テキスト処理ユーティリティー (C 言語)
stx2any	V:0, I:0	264	text → any	構造化プレーンテキストからたのフォーマットへの文書変換ソフト (m4)
rest2web	V:0, I:0	527	text → html	ReStructured Text から html への文書変換ソフト (Python)
aft	V:0, I:0	235	text → any	" 自由フォーム" 文書準備システム (Perl)
yodl	V:0, I:0	610	text → any	プリ文書言語とその処理用のツール (C 言語)
sdf	V:0, I:0	1445	text → any	単純文書処理ソフト (Perl)
sisu	V:0, I:0	5344	text → any	文書の構造化と出版と探索の枠組み (Ruby)

Table 11.6: プレーンテキストデーターをハイライトするツールのリスト

11.2 XML データー

Extensible Markup Language (XML) は構造化情報を含む文書のためのマークアップ言語です。

XML.COM にある入門情報を参照下さい。

- "What is XML?"
- "What Is XSLT?"
- "What Is XSL-FO?"

- ”What Is XLink?”

11.2.1 XML に関する基本ヒント

XML テキストはちょっと HTML のようにも見えます。これを使うと一つの文書から複数のフォーマットのアウトプット管理できるようになります。簡単な XML システムの一つはここで使っている `docbook-xsl` パッケージです。

各 XML ファイルは次のような標準的な XML 宣言でスタートします。

```
<?xml version="1.0" encoding="UTF-8"?>
```

XML 要素の基本的シンタックスは次のようにマークアップされます。

```
<name attribute="value">content</name>
```

空の XML 要素は次の短縮形を使ってマークアップされます。

```
<name attribute="value"/>
```

上記例中の”`attribute="value"`” はオプションです。

XML 中のコメントセクションは次のようにマークアップされます。

```
<!-- comment -->
```

マークアップを追加する以外に、XML は次の文字に関して事前定義されたエンティティを使い内容を少し改変する必要があります。

事前定義されたエンティティ	変換先の文字
`&quot;`	”: 引用符
`&apos;`	’: アポストロフィ
`<`	<: 以下
`>`	>: 以上
`&`	&: アンパサンド

Table 11.7: XML で事前定義されているエントリーのリスト

注意
”<” と”&” はアトリビュートやエレメントには使えません。

注意
例えば”`&some-tag:`” 等の SGML スタイルのユーザー定義エンティティが使われた場合、他の定義は無効で最初の定義が有効です。エンティティ定義は”`<!ENTITY 何らかのタグ" エンティティ値">`” と表現されます。

注意
XML のマークアップがタグ名の何らかの組み合わせで (あるデーターを内容としてであれアトリビュート値としてであれ) 整合性を持ってされている限り、他の XML の変換は拡張可能スタイルシート言語変換 (XSLT) を使うととっても簡単な作業です。

11.2.2 XML 処理

拡張可能スタイルシート言語 (XSL) のような XML ファイルを処理に利用可能なツールは沢山あります。

基本的に、良くできた XML ファイルを一度作ると、いかなるフォーマットへも拡張可能なスタイルシート言語変換 (XSLT) を使って変換できます。

フォーマットオブジェクト用拡張可能スタイルシート言語 (XSL-FO) がフォーマットのための答えとなるはずです。`fop` パッケージは Java プログラム言語 に依存するため Debian の `main` アーカイブでは新規です。このため、LaTeX コードが XML から XSLT を使って通常作成され、DVI や PostScript や PDF 等のプリンタブルなファイルが LaTeX システムを使って作成されます。

パッケージ	ポプコン	サイズ	キーワード	説明
`docbook-xml`	V:15, I:280	2133	xml	DocBook 用 XML ドキュメントタイプ定義 (DTD)
`xsltproc`	V:15, I:104	160	xslt	XSLT コマンドラインプロセスソフト (XML→ XML, HTML, plain text, 他)
`docbook-xsl`	V:13, I:165	14870	xml/xslt	DocBook XML を XSLT を使って各種アウトプットへ処理する XSL スタイルシート
`xmlto`	V:1, I:23	130	xml/xslt	XSLT を用いて XML から全てへの変換ソフト
`dbtoepub`	V:0, I:0	37	xml/xslt	DocBook XML から.epub へのコンバーター
`dblatex`	V:3, I:16	4643	xml/xslt	XSLT を使って Docbook ファイルを DVI, PostScript, PDF 文書へ変換
`fop`	V:1, I:24	291	xml/xsl-fo	Docbook XML ファイルを PDF に変換

Table 11.8: XML ツールのリスト

XML は標準一般化マークアップ言語 (SGML) のサブセットなので、ドキュメントスタイル構文規程言語 (DSSSL) 等の SGML 用として利用可能な広範なツールで処理できます。

パッケージ	ポプコン	サイズ	キーワード	説明
`openjade`	V:2, I:38	1019	dsssl	ISO/IEC 10179: 1996 標準 DSSSL プロセッサ (最新版)
`docbook-dsssl`	V:1, I:23	2604	xml/dsssl	DocBook XML を各種出力フォーマットに DSSSL を使って処理するための DSSSL スタイルシート
`docbook-utils`	V:0, I:16	281	xml/dsssl	`docbook2*` コマンドで DSSSL を使って DocBook ファイルを他のフォーマットに (HTML, RTF, PS, man, PDF) 変換するなどのユーティリティー
`sgml2x`	V:0, I:0	90	SGML/dsssl	DSSSL スタイルシートを使う SGML や XML からの変換ソフト

Table 11.9: DSSL ツールのリスト

ティップ
GNOME の `yelp` は DocBook XML ファイルを X 上に体裁良く表示するので時々便利です。

11.2.3 XML データー抽出

他のフォーマットから以下を使うと HTML とか XML のデーターを抽出出来ます。

非 XML の HTML ファイルの場合は、これらを整合性ある XML である XHTML に変換できます。XHTML は XML ツールで処理できます。

パッケージ	ポプコン	サイズ	キーワード	説明
wv	V:0, I:7	717	MSWord → any	Microsoft Word から HTML や LaTeX 等への文書変換ソフト
texi2html	V:0, I:8	1833	texi → html	Texinfo から HTML への変換ソフト
man2html	V:0, I:2	138	manpage → html	manpage から HTML への変換ソフト (CGI サポート)
unrtf	V:0, I:4	148	rtf → html	RTF から HTML 等への文書変換ソフト
info2www	V:1, I:3	76	info → html	GNU info から HTML への変換ソフト (CGI サポート)
ooo2dbk	V:0, I:0	217	sxw → xml	OpenOffice.org SXW 文書から DocBook XML への変換ソフト
wp2x	V:0, I:0	202	WordPerfect → any	WordPerfect 5.0 と 5.1 ファイルから TeX と LaTeX と troff と GML と HTML への変換ソフト
doclifter	V:0, I:0	451	troff → xml	troff から DocBook XML への変換ソフト

Table 11.10: テキストデーター変換ツールのリスト

パッケージ	ポプコン	サイズ	キーワード	説明
libxml2-utils	V:22, I:246	182	xml ↔ html ↔ xhtml	xmllint(1) (シンタクスチェック、リフォーマット、整形、他) を含むコマンドライン XML ツール
tidy	V:2, I:14	84	xml ↔ html ↔ xhtml	HTML シンタックスチェックソフトとリフォーマットソフト

Table 11.11: XML 整形印刷ツールのリスト

一度適正な XML が生成されれば、XSLT 技術を使ってマークアップコンテキスト等に基づいてデーターを抽出出来ます。

11.3 タイプセッティング

Unix のtroff プログラムは最初 AT&T で開発されました。それはマンページを作成するのに通常使われます。

Donald Knuth 氏によって作成された TeX は非常に強力な組版ツールでデファクト標準です。最初 Leslie Lamport 氏によって書かれた LaTeX は TeX の力への高レベルアクセスを可能にします。

パッケージ	ポプコン	サイズ	キーワード	説明
texlive	V:3, I:50	71	(La)TeX	組版、校正、印刷のための TeX システム
groff	V:3, I:64	11838	troff	GNU troff テキストフォーマティングシステム

Table 11.12: タイプ設定ツールのリスト

11.3.1 roff タイプセッティング

伝統的には、roff が主な Unix テキスト処理システムです。roff(7) と groff(7) と groff(1) と grotty(1) と troff(1) と groff_mdoc(7) と groff_man(7) と groff_ms(7) と groff_me(7) と groff_mm(7) と"`info groff`" を参照下さい。

`groff` パッケージをインストールすると"`/usr/share/doc/groff/`" 中に"`-me`"マクロに関する良い入門書や参考書が読めます。

ティップ
”groff -Tascii -me -” はANSI エスケープコードを含むプレーンテキストを生成します。もしマンページのような多くの”^H” や”_” を含む出力が欲しい場合には、この代わりに”GROFF_NO_SGR=1 groff -Tascii -me -” を使います。

ティップ
groff が生成した”^H” や”_” をテキストから削除するには、それを”col -b -x” でフィルターします。

11.3.2 TeX/LaTeX

TeX Live ソフトウエアーディストリビューションは完全な TeX システムを提供します。texlive メタパッケージは、ほとんどの一般的タスクに十分な TeX Live パッケージのまともな選択を提供します。

TeX と LaTeX に関する多くの参考書が利用可能です。

- The teTeX HOWTO: The Linux-teTeX Local Guide
- tex(1)
- latex(1)
- texdoc(1)
- texdoctk(1)
- ”The TeXbook”、Donald E. Knuth 著 (Addison-Wesley)
- ”LaTeX - A Document Preparation System”、Leslie Lamport 著 (Addison-Wesley)
- ”The LaTeX Companion”、Goossens と Mittelbach と Samarin 著 (Addison-Wesley)

これはもっとも強力な組版環境です。多くの SGML 処理ソフトはこれをバックエンドのテキスト処理ソフトとしています。多くの人が Emacs や Vim をソースのエディターとして使う一方、lyx パッケージが提供する Lyx と texmacs パッケージが提供する GNU TeXmacs は洒落た LaTeX のWYSIWYG 編集環境を提供します。

多くのオンラインリソースが利用可能です。

- TEX Live ガイド - TEX Live 2007 (”/usr/share/doc/texlive-doc-base/english/texlive-en/live.html”) (texlive-doc-base パッケージ)
- A Simple Guide to Latex/Lyx
- Word Processing Using LaTeX
- Local User Guide to teTeX/LaTeX

文書が大きくなると、TeX はエラーを発生する事があります。この問題の解決には (正しくは”/etc/texmf/texmf.d/95No を編集し update-texmf(8) を実行することで) ”/etc/texmf/texmf.cnf” 中のプールの数を増やし修正しなければいけません。

注意
このファイルには必要なマクロのほとんど全てが含まれます。この文書は 7 から 10 行をコメントして”\input manmac \proofmodefalse” を追加すると tex(1) で処理できると聞いた事があります。オンラインバージョンを使うのではなくこの本 (さらに Donald E. Knuth 氏による全ての本) を購入される事を強く勧めます。しかし、そのソースは TeX の入力の非常に良い例です！

11.3.3 マニュアルページを綺麗に印刷

次のコマンドでマンページを PostScript で上手く印刷できます。

```
$ man -Tps some_manpage | lpr
```

11.3.4 マニュアルページの作成

プレーンな troff フォーマットでマンページ (マニュアルページ) を書く事は可能ですが、それを作成するヘルパーパッケージがあります。

パッケージ	ポプコン	サイズ	キーワード	説明
docbook-to-man	V:0, I:13	191	SGML → manpage	DocBook SGML から roff man マクロへの変換ソフト
help2man	V:0, I:10	498	text → manpage	--help からの自動マンページ生成ソフト
info2man	V:0, I:0	134	info → manpage	GNU info から POD かマンページへの変換ソフト
txt2man	V:0, I:1	114	text → manpage	ベタの ASCII テキストからマンページ形式へ変換

Table 11.13: マンページ作成を補助するパッケージのリスト

11.4 印刷可能データー

Debian システム上では印刷可能なデーターは PostScript フォーマットで表現されます。共通 Unix 印刷システム (CUPS) は非 PostScript プリンタ用のラスタ化のバックエンドプログラムとして Ghostscript を使用します。

11.4.1 Ghostscript

印刷データー処理の核心はラスタ画像を生成する Ghostscript という PostScript (PS) インタープリタです。

最新の Artifex からのアップストリーム版 Ghostscript は統合リリースである 8.60 リリースにて AFPL から GPL にライセンス変更され最新の ESP バージョンによる CUPS 関連等の変更をマージしました。

パッケージ	ポプコン	サイズ	説明
ghostscript	V:252, I:598	231	GPL 版 Ghostscript PostScript/PDF インタープリタ
ghostscript-x	V:15, I:65	223	GPL 版 Ghostscript PostScript/PDF インタープリタ - X ディスプレーサポート
libpoppler95	I:3	4172	xpdf PDF ビューワー派生 PDF レンダリングライブラリー
libpoppler-glib8	V:217, I:481	449	PDF レンダリングライブラリー (GLib 準拠共有ライブラリー)
poppler-data	V:111, I:637	13090	PDF レンダリングライブラリー用 CMaps (CJK サポート: Adobe-*)

Table 11.14: Ghostscript PostScript インタープリタのリスト

ティップ
”gs -h” とすると Ghostscript の設定が表示されます。

11.4.2 2 つの PS や PDF ファイルをマージ

2 つの PostScript (PS) や Portable Document Format (PDF) ファイルは Ghostscript の gs(1) をつかってマージできます。

```
$ 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
```

注意
PDF は、クロスプラットフォームの印刷可能フォーマットとして広範に使われていて、本質的にいくつかの追加機能と拡張がされている、圧縮 PS フォーマットです。

ティップ
コマンドラインの場合、`psutils` パッケージ中の psmerge(1) 等のコマンドは PostScript 文書を操作するのに便利です。`pdftk` パッケージの pdftk(1) も PDF 文書を操作するのに便利です。

11.4.3 印刷可能データーユーティリティー

印刷可能なデーターに用いる次のパッケージが著者の目に止まりました。

パッケージ	ポプコン	サイズ	キーワード	説明
poppler-utils	V:241, I:434	689	pdf → ps,text, …	PDF ユーティリティー: `pdftops`, `pdfinfo`, `pdfimages`, `pdftotext`, `pdffonts`
psutils	V:6, I:105	219	ps → ps	PostScript 文書変換ツール
poster	V:0, I:5	58	ps → ps	PostScript ページから大きなポスターを作る
enscript	V:1, I:20	2132	text → ps, html, rtf	ASCII テキストから PostScript か HTML か RTF か Pretty-Print への変換
a2ps	V:1, I:15	3651	text → ps	全てを PostScript に変換するソフトと綺麗印刷ソフト
pdftk	I:51	28	pdf → pdf	PDF 文書変換ツール: `pdftk`
html2ps	V:0, I:3	249	html → ps	HTML から PostScript への変換ソフト
gnuhtml2latex	V:0, I:1	27	html → latex	html から latex への変換ソフト
latex2rtf	V:0, I:6	480	latex → rtf	LaTeX から MS Word で読める RTF へと文書変換
ps2eps	V:3, I:68	98	ps → eps	PostScript から EPS (カプセル化済み PostScript) への変換ソフト
e2ps	V:0, I:0	109	text → ps	日本語符号化サポート付きの Text から PostScript への変換ソフト
impose+	V:0, I:0	119	ps → ps	PostScript ユーティリティー
trueprint	V:0, I:0	146	text → ps	多くのソースコード (C, C++, Java, Pascal, Perl, Pike, Sh, Verilog) の PostScript への綺麗印刷 (C 言語)
pdf2svg	V:0, I:4	30	ps → svg	PDF からスケール可のベクトルグラフィクス (SVG) フォーマットへの変換ソフト
pdftoipe	V:0, I:0	71	ps → ipe	PDF から IPE の XML フォーマットへの変換ソフト

Table 11.15: プリントできるデーターのユーティリティーのリスト

11.4.4 CUPS を使って印刷

Common Unix Printing System (CUPS) が提供する、lp(1) と lpr(1) コマンドの両方が印刷可能なデーターの印刷をカスタム化するオプションを提供します。

以下のコマンドの内のひとつを使い一つのファイルに対し 3 部の印刷をページ順に揃えてできます。

```
$ lp -n 3 -o Collate=True filename
```

```
$ lpr -#3 -o Collate=True filename
```

さらに、コマンドライン印刷とオプションに書かれているように”`-o number-up=2`” や”`-o page-set=even`”, ”`-o page-set=odd`” や”`-o scaling=200`” や”`-o natural-scaling=200`” 等の印刷オプションを使ってカスタム化できます。

11.5 メールデーター変換

テキストデーター変換のための次のパッケージが著者の目に止まりました。

パッケージ	ポプコン	サイズ	キーワード	説明
`sharutils`	V:4, I:55	1421	メール	shar(1) と unshar(1) と uuencode(1) と uudecode(1)
`mpack`	V:1, I:18	106	MIME	MIME メッセージの符号化と逆符号化のソフト: mpack(1) と munpack(1)
`tnef`	V:1, I:10	110	ms-tnef	Microsoft のみのフォーマットの”application/ms-tnef” タイプの MIME アタッチメントを開梱
`uudeview`	V:0, I:5	109	メール	次のフォーマットのエンコーダーとデコーダー: uuencode, xxencode, BASE64, quoted printable, BinHex

Table 11.16: メールデーター変換を補助するパッケージのリスト

ティップ
インターネットメッセージアクセスプロトコル バージョン 4 (IMAP4) サーバー (項6.7参照下さい) は、プロプライエタリメールシステムのクライアントソフトが IMAP4 サーバーも使えるように設定できる場合、プロプライエタリメールシステムからメールを取り出すのに利用できるかもしれません。

11.5.1 メールデーターの基本

メイル (SMTP) データーは 7 ビットデーター列に限定されるべきです。だからバイナリーデーターや 8 ビットテキストデーターはMultipurpose Internet Mail Extensions (MIME) を用いたり文字セット (項8.4.1参照下さい) を選択して 7 ビットのフォーマットにエンコードされます。

標準のメールストレージフォーマットは RFC2822 (RFC822 の更新版) により定義される mbox フォーマットです。mbox(5) (`mutt` パッケージが提供) を参照下さい。

欧州言語の場合､ほとんど 8 ビット文字が無いので ISO-8859-1 文字セットとともに”`Content-Transfer-Encoding: quoted-printable`” が通常メールに使われます。欧州のテキストが UTF-8 符号化された場合、ほとんどが 7 ビット文字なので”`Content-Transfer-Encoding: quoted-printable`” が大体使われます。

日本語には､テキストを 7 ビットにしておくために伝統的に”`Content-Type: text/plain; charset=ISO-2022-JP`” がメールに使われます。しかし、古い Microsoft システムは適正な宣言無しに Shift-JIS でメールデーターを送るかもしれません。日本語のテキストが UTF-8 で符号化される場合、多くの 8 ビットデーターを含むので Base64 が大体使われます。他のアジアの言語でも状況は同様です。

注意
もし IMAP4 サーバー (項6.7参照下さい) と話せる非 Debian クライアントからあなたの非 Unix メールデーターがアクセス出きるなら、あなた自身の IMAP4 サーバーを実行することでメールデーターを引き出せるかもしれません。

注意
もし他のメールストレージフォーマットを使っている場合、mbox フォーマットに移動するのが良い第一歩です。mutt(1) のような汎用クライアントプログラムはこれに非常に便利です。

メールボックスの内容は procmail(1) と formail(1) を使って各メッセージに分割できます。

各メールメッセージは `mpack` パッケージにある munpack(1) (または他の専用ツール) を使って開梱して MIME 符号化された内容を取り出せます。

11.6 グラフィクスデーターツール

印刷可能なデーターに用いる次のパッケージが著者の目に止まりました。

ティップ
aptitude(8) の正規表現"~`Gworks-with::image`" (項2.2.6参照下さい) を使ってさらなる画像ツールを探しましょう。

gimp(1) のような GUI プログラムは非常に強力ですが、imagemagick(1) 等のコマンドラインツールはスクリプトでイメージ操作を自動化するのに非常に便利です。

デジタルカメラのファイルフォーマットのデファクト標準は、追加のメタデーター付きの JPEG 画像ファイルフォーマットである交換可能な画像ファイルフォーマット (EXIF) です。EXIF は日付や時間やカメラ設定等の情報を保持できます。

Lempel-Ziv-Welch (LZW) ロス無しデーター圧縮特許の期限は切れました。LZW データー圧縮を使う画像交換フォーマット (GIF) ユーティリティーは Debian システム上で自由に利用可能となりました。

ティップ
リムーバブル記録メディア付きのどのデジタルカメラやスキャナーも、カメラファイルシステム用デザインルール に準拠し FAT ファイルシステムを使っているので USB ストレージ読取り機を経由すれば Linux で必ず機能します。項10.1.7参照下さい。

11.7 その他のデーター変換

多くのデーター変換プログラムがあります。aptitude(8) で"~`Guse::converting`" という正規表現 (項2.2.6参照下さい) を使い次のプログラムが私の目に止まりました。

RPM フォーマットからのデーター抽出もまた次のようにするとできます。

```
$ rpm2cpio file.src.rpm | cpio --extract
```

パッケージ	ポプコン	サイズ	キーワード	説明
gimp	V:68, I:341	22313	画像 (bitmap)	GNU イメージ操作プログラム
imagemagick	I:400	218	画像 (bitmap)	画像操作プログラム
graphicsmagick	V:3, I:17	5224	画像 (bitmap)	画像操作プログラム (imagemagick のフォーク)
xsane	V:17, I:173	2346	画像 (bitmap)	GTK+ に基づく SANE (Scanner Access Now Easy) 用の X11 フロントエンド
netpbm	V:32, I:409	4302	画像 (bitmap)	画像変換ツール
icoutils	V:21, I:127	221	png↔ico(bitmap)	MS Windows のアイコンやカーソールと PNG フォーマット間の変換 (favicon.ico)
scribus	V:2, I:23	30375	ps/pdf/SVG/…	Scribus DTP エディター
libreoffice-draw	V:177, I:434	14600	画像 (vector)	LibreOffice office スイート - ドロー
inkscape	V:55, I:209	84823	画像 (vector)	SVG (スケーラブルベクトルグラフィクス) エディター
dia	V:5, I:31	3727	画像 (vector)	ダイアグラムエディター (Gtk)
xfig	V:2, I:15	1793	画像 (vector)	X11 下でインテラクティブ生成するソフト
pstoedit	V:4, I:98	988	ps/pdf→ 画像 (vector)	PostScript と PDF ファイルから編集可能なベクトルグラフィクスへの変換ソフト (SVG)
libwmf-bin	V:10, I:211	113	Windows/画像 (vector)	Windows メタファイル (ベクトル画像データー) 変換ツール
fig2sxd	V:0, I:0	149	fig→sxd(vector)	XFig ファイルを OpenOffice.org Draw フォーマットに変換
unpaper	V:2, I:19	460	画像 → 画像	OCR 用のスキャンしたページの後処理ツール
tesseract-ocr	V:8, I:37	1500	画像 → テキスト	HP の商用 OCR エンジンの基づくフリーの OCR ソフトウエアー
tesseract-ocr-eng	V:7, I:37	4032	画像 → テキスト	OCR エンジンデーター: tesseract-ocr の英文用言語ファイル
gocr	V:1, I:13	531	画像 → テキスト	フリー OCR ソフト
ocrad	V:0, I:5	303	画像 → テキスト	フリー OCR ソフト
eog	V:71, I:264	10189	画像 (Exif)	画像ビューアープログラム Eye of GNOME
gthumb	V:5, I:22	5475	画像 (Exif)	画像ビューアー兼ブラウザー (GNOME)
geeqie	V:6, I:21	14643	画像 (Exif)	GTK+ を用いた画像ビューアー
shotwell	V:19, I:223	6451	画像 (Exif)	デジタル写真オーガナイザー (GNOME)
gtkam	V:0, I:6	1154	画像 (Exif)	デジタルカメラからメディアを回収するアプリケーション (GTK+)
gphoto2	V:1, I:12	955	画像 (Exif)	gphoto2 デジタルカメラコマンドライン版クライアント
gwenview	V:28, I:97	10570	画像 (Exif)	画像ビューア (KDE)
kamera	I:97	798	画像 (Exif)	KDE アプリケーション用デジタルカメラサポート
digikam	V:2, I:13	2646	画像 (Exif)	デジタル写真管理アプリケーション
exiv2	V:3, I:44	321	画像 (Exif)	EXIF/IPTC メタデーター操作ツール
exiftran	V:1, I:21	70	画像 (Exif)	デジタルカメラの jpeg 画像を変換
jhead	V:1, I:11	109	画像 (Exif)	Exif に準拠の JPEG (デジタルカメラ写真) ファイルの非画像部を操作
exif	V:1, I:12	339	画像 (Exif)	JPEG ファイル中の EXIF 情報を表示するコマンドラインユーティリティー
exiftags	V:0, I:5	292	画像 (Exif)	デジタルカメラの JPEG ファイルから Exif タグを読むユーティリティー
exifprobe	V:0, I:4	499	画像 (Exif)	デジタル写真からメタデーターを読み出す
dcraw	V:2, I:19	535	画像 (Raw)→ppm	生のデジタルカメラ画像のデコード
			画像	

パッケージ	ポプコン	サイズ	キーワード	説明
alien	V:2, I:34	161	rpm/tgz→deb	外来のパッケージの Debian パッケージへの変換ソフト
freepwing	V:0, I:0	421	EB→EPWING	”Electric Book” (日本で人気) から単一の JIS X 4081 フォーマット (EPWING V1 のサブセット) への変換ソフト
calibre	V:9, I:36	54876	any→EPUB	e-book コンバーターとライブラリーの管理

Table 11.18: その他のデーター変換ツールのリスト

Chapter 12

プログラミング

Debian システム上でプログラミングを学ぶ人がパッケージ化されたソースコードを読み込めるようになるための指針を示します。以下はプログラムに関する特記すべきパッケージと対応する文書パッケージです。

パッケージ	ポプコン	サイズ	文書
autoconf	V:41, I:282	1846	autoconf-doc が提供する”info autoconf
automake	V:42, I:279	1830	automake1.10-doc が提供する”info automake”
bash	V:791, I:999	6469	bash-doc が提供する”info bash”
bison	V:9, I:103	2815	bison-doc が提供する”info bison”
cpp	V:319, I:770	42	cpp-doc が提供する”info cpp”
ddd	V:0, I:10	4184	ddd-doc が提供する”info ddd”
exuberant-ctags	V:5, I:37	341	exuberant-ctags(1)
flex	V:9, I:93	1279	flex-doc が提供する”info flex”
gawk	V:368, I:454	2558	gawk-doc が提供する”info gawk”
gcc	V:165, I:604	45	gcc-doc が提供する”info gcc”
gdb	V:13, I:114	9789	gdb-doc が提供する”info gdb”
gettext	V:48, I:312	5843	gettext-doc が提供する”info gettext”
gfortran	V:11, I:98	16	gfortran-doc が提供する”info gfortran” (Fortran 95)
fpc	I:3	121	python-doc が提供する python(1) と html ページ (Pascal)
glade	V:0, I:8	1730	メニューが提供するヘルプ (UI Builder)
libc6	V:935, I:999	12771	glibc-doc と glibc-doc-reference が提供する”info libc”
make	V:157, I:609	1592	make-doc が提供する”info make”
xutils-dev	V:1, I:12	1466	imake(1), xmkmf(1), 他
mawk	V:372, I:997	242	mawk(1)
perl	V:610, I:992	705	perl-doc と perl-doc-html が提供する perl(1) と html
python	V:293, I:923	68	python-doc が提供する python(1) と html ページ
tcl	V:31, I:414	22	tcl-doc が提供する tcl(3) と詳細なマンページ
tk	V:30, I:406	22	tk-doc が提供する tk(3) と詳細なマンページ
ruby	V:137, I:318	35	ri が提供する ruby(1) と詳細なマンページ
vim	V:106, I:398	3231	vim-doc が提供するヘルプ (F1) メニュー
susv2	I:0	16	”The Single UNIX Specifications v2” を取得
susv3	I:0	16	”The Single UNIX Specifications v3” を取得

Table 12.1: プログラミングを補助するパッケージのリスト

オンラインリファレンスは manpages と manpages-dev パッケージをインストールした後で”man name” とタイプすると利用可能です。GNU ツールのオンラインリファレンスは該当する文書パッケージをインストールした後

で”info program_name” とタイプすると使えます。一部の GFDL 文書は DFSG に準拠していないと考えられているので main アーカイブに加えて contrib や non-free アーカイブを含める必要があるかもしれません。

警告
”test” を実行可能なテストファイルの名前に用いてはいけません。”test” はシェルのビルトインです。

注意
ソースから直接コンパイルしたソフトウエアープログラムは、システムプログラムとかち合わないように、”/usr/local” か”/opt” の中にインストールします。

ティップ
”99 ボトルのビールの歌” 作成のコード例はほとんど全てのプログラム言語に関する理解のための非常に好適です。

12.1 シェルスクリプト

シェルスクリプトは実行ビットがセットされたテキストファイルで、以下に示すフォーマットのコマンドを含んでいます。

```
#!/bin/sh
 ... b'' コ b''b'' マ b''b'' ン b''b'' ド b''b'' 行 b''
```

最初の行はこのファイル内容を読み実行するシェルインタープリタを指定します。

シェルスクリプトを読むのは Unix 的なシステムがどのように機能しているのかを理解する最良の方法です。ここでは、シェルプログラムに関する指針や心がけを記します。失敗から学ぶために” シェルの失敗” (http://www.greenend.org.uk/rjk/2001/04/shell.html) を参照下さい。

シェル対話モード (項1.5と項1.6参照下さい) と異なり、シェルスクリプトは変数や条件文やループを繁用します。

12.1.1 POSIX シェル互換性

多くのシステムスクリプトは POSIX シェル (表 1.13参照下さい) のどれで解釈されるか分かりません。システムのデフォールトシェルは実際のプログラムをさしているシムリンクである”/bin/sh” です。

- bash(1)、lenny 以前の場合
- dash(1)、squeeze 以降の場合

全ての POSIX シェル間でポータブルとするために **bashisms** や **zshisms** を使うシェルスクリプトを書くのを避けましょう。checkbashisms(1) を使うとこれがチェックできます。

”echo” コマンドはその実装がシェルビルトインや外部コマンド間で相違しているので次の注意点を守って使わなければいけません。

- ”-n” 以外のどのコマンドオプション使用も避けます。
- 文字列中にエスケープシーケンスはその取扱いに相違があるので使用を避けます。

推薦: **POSIX**	回避すべき: **bashism**
`if [ "$foo" = "$bar" ] ; then …`	`if [ "$foo" == "$bar" ] ; then …`
`diff -u file.c.orig file.c`	`diff -u file.c{.orig,}`
`mkdir /foobar /foobaz`	`mkdir /foo{bar,baz}`
`funcname() { …}`	`function funcname() { …}`
8 進表記: "`\377`"	16 進表記: "`\xff`"

Table 12.2: 典型的 bashizms のリスト

注意
"`-n`" オプションは実は POSIX シンタックスではありませんが、一般的に許容されています。

ティップ
出力文字列にエスケープシーケンスを埋め込む必要がある場合には、"`echo`" コマンドの代わりに"`printf`" コマンドを使います。

12.1.2 シェル変数

特別なシェルパラメーターがシェルスクリプト中ではよく使われます。

シェル変数	変数値
`$0`	シェルまたはシェルスクリプトの名前
`$1`	最初 (1 番目) のシェル引数
`$9`	9 番目のシェル引数
`$#`	シェル引数の数
`"$*"`	`"$1 $2 $3 $4 …"`
`"$@"`	`"$1" "$2" "$3" "$4" …`
`$?`	最近実行のコマンドの終了状態
`$$`	このシェルスクリプトの PID
`$!`	最近スタートしたバックグラウンドジョブの PID

Table 12.3: シェル変数のリスト

覚えておくべき基本的なパラメーター展開を次に記します。

パラメーター式形	**var** が設定されていればの値	**var** が設定されていなければの値
`${var:-string}`	"`$var`"	"`string`"
`${var:+string}`	"`string`"	"`null`"
`${var:=string}`	"`$var`"	"`string`" (合わせて"`var=string`" を実行)
`${var:?string}`	"`$var`"	"`string`" を **stderr** に出力 (エラーとともに exit する)

Table 12.4: シェル変数展開のリスト

ここで、これら全てのオペレーターのコロン"`:`" は実際はオプションです。

- "`:`" 付き = 存在と非ヌル文字列をテストするオペレータ
- "`:`" 無し = 存在のみをテストするオペレータ

パラメーター置換形	結果
`${var%suffix}`	最短のサフィクスパターンを削除
`${var%%suffix}`	最長のサフィクスパターンを削除
`${var#prefix}`	最短のプリフィクスパターンを削除
`${var##prefix}`	最長のプリフィクスパターンを削除

Table 12.5: 重要なシェル変数置換のリスト

12.1.3 シェル条件式

各コマンドは条件式に使えるエグジットステイタスを返します。

- 成功: 0 (” 真”)
- エラー: 非 0 (” 偽”)

注意
シェル条件文の文脈中の”0” は” 真” を意味します、一方 C 条件文の文脈中の”0” は” 偽” を意味します。

注意
”[” は、”]” までの引数を条件式として評価する、`test` コマンドと等価です。

覚えておくべき基本的な条件文の慣用句は次です。

- ”`<command> && < 成功したらこの command も実行 > || true`”
- ”`<command> || < もし command が成功しないとこのコマンドも実行 > || true`”
- 次のようなマルチラインスクリプト断片

```
if [ <b’’ 条 b’’b’’ 件 b’’b’’ 式 b’’> ]; then
 <b’’ 成 b’’b’’ 功 b’’b’’ な b’’b’’ ら b’’b’’ こ b’’b’’ の b’’b’’ コ b’’b’’ マ b’’b’’ ン b’’b’’ ド
     b’’b’’ を b’’b’’ 実 b’’b’’ 行 b’’>
else
 <b’’ 成 b’’b’’ 功 b’’b’’ で b’’b’’ な b’’b’’ い b’’b’’ な b’’b’’ ら b’’b’’ こ b’’b’’ の b’’b’’ コ
     b’’b’’ マ b’’b’’ ン b’’b’’ ド b’’b’’ を b’’b’’ 実 b’’b’’ 行 b’’>
fi
```

ここで、シェルスクリプトが”-e” フラグ付きで起動された際にシェルスクリプトがこの行で exit しないようにするために末尾の”`|| true`” が必要です。

式	論理真を返す条件
`-e <file>`	<file> 存在する
`-d <file>`	<file> 存在しディレクトリーである
`-f <file>`	<file> 存在し通常ファイルである
`-w <file>`	<file> 存在し書込み可
`-x <file>`	<file> 存在し実行可
`<file1> -nt <file2>`	<file1> は <file2> より新しい (変更)
`<file1> -ot <file2>`	<file1> は <file2> より古い (変更)
`<file1> -ef <file2>`	<file1> と <file2> は同デバイス上の同 inode 番号

Table 12.6: 条件式中のファイル比較オペレーター

条件式中の算術整数比較演算子は”`-eq`” と”`-ne`” と”`-lt`” と”`-le`” と”`-gt`” と”`-ge`” です。

式	論理真を返す条件
`-z <str>`	<str> の長さがゼロ
`-n <str>`	<str> の長さが非ゼロ
`<str1> = <str2>`	<str1> と <str2> は等しい
`<str1> != <str2>`	<str1> と <str2> は等しく無い
`<str1> < <str2>`	<str1> は <str2> より前 (ロケール依存)
`<str1> > <str2>`	<str1> は <str2> より後 (ロケール依存)

Table 12.7: 条件式中での文字列比較オペレータのリスト

12.1.4 シェルループ

POSIX シェル中で使われるループの慣用句があります。

- ”`for x in foo1 foo2 …; do` コマンド; `done`” は”`foo1 foo2 …`” リストの項目を変数”x” にアサインし” コマンド” を実行してループします。
- ”`while` 条件; `do` コマンド; `done`” は” 条件” が真の場合” コマンド” を繰り返します。
- ”`until` 条件; `do` コマンド; `done`” は” 条件” が真でない場合” コマンド” を繰り返します。
- ”`break`” に出会うと、ループからの脱出が出来ます。
- ”`continue`” に出会うと、次のループ初めに戻りループを再開します。

ティップ
C 言語のような数字の繰り返しは”`foo1 foo2 ...`” 生成に seq(1) 使って実現します。

ティップ
項9.3.9を参照下さい。

12.1.5 シェルコマンドライン処理シーケンス

シェルはおおよそ次のシーケンスでスクリプトを処理します。

- シェルは 1 行読み込みます。
- シェルは、もし ”…” や ’…’ の中なら、行の一部を **1** つのトークンとしてグループします。
- シェルは 1 行を次のによってトークンに分割します。
  - 空白: `<space> <tab> <newline>`
  - メタ文字: `< > | ; & ( )`
- シェルは、もし ”…” や ’…’ の中でないなら、各トークンを予約語に対してチェックしその挙動を調整します。
  - 予約語: `if then elif else fi for in while unless do done case esac`
- シェルは、もし ”…” や ’…’ の中でないなら、エリアスを展開します。
- シェルは、もし”…” や’…’ の中でないなら、ティルドを展開します。
  - ”~” → 現ユーザーのホームディレクトリー
  - ”~<user>” → <user> のホームディレクトリー

- シェルは、もし '…' の中でないなら、パラメーター" をその値に展開します。
  - パラメーター: "$PARAMETER" or "${PARAMETER}"
- シェルは、もし '…' の中でないなら、コマンドの置き換えを展開します。
  - "$(command)" → "command" の出力
  - "` command `" → "command" の出力
- シェルは、もし "…" や '…' の中でないなら、パス名のグロブを展開します。
  - * → いかなる文字
  - ? → 1 文字
  - […] → "…" 中の 1 つ
- シェルはコマンドを次から検索して実行します。
  - 関数定義
  - ビルトインコマンド
  - "$PATH" 中の実行ファイル
- シェルは次行に進みこのプロセスを一番上から順に反復します。

ダブルクォートの中のシングルクォートは特段の効果はありません。

シェル環境中で"set -x" を実行したり、シェルを"-x" オプションで起動すると、シェルは実行するコマンドを全てプリントするようになります。これはデバグをするのに非常に便利です。

12.1.6 シェルスクリプトのためのユーティリティープログラム

Debian システム上でできるだけポータブルなシェルプログラムとするには、ユーティリティープログラムを **essential** パッケージで提供されるプログラムだけに制約するのが賢明です。

- "aptitude search ~E" は **essential** (必須) パッケージをリストします。
- "dpkg -L < パッケージ名 > |grep '/man/man.*/'" は < パッケージ名 > パッケージによって提供されるコマンドのマンページをリストします。

パッケージ	ポプコン	サイズ	説明
coreutils	V:891, I:999	17478	GNU コアユーティリティー
debianutils	V:925, I:999	230	Debian 限定の雑ユーティリティー
bsdmainutils	V:60, I:996	26	FreeBSD 由来の追加ユーティリティー集
bsdutils	V:673, I:999	393	4.4BSD-Lite 由来の基本ユーティリティー
moreutils	V:11, I:35	237	追加の Unix ユーティリティー

Table 12.8: シェルスクリプト用の小さなユーティリティープログラムを含むパッケージのリスト

ティップ
moreutils は Debian の外では存在しないかも知れませんが、興味深い小さなプログラムを提供します。もっとも特記すべきは、オリジナルファイルを上書きしたいときに非常に有効な sponge(8) です。

パッケージ	ポプコン	サイズ	説明
x11-utils	V:180, I:599	712	xmessage(1): window 中にメッセージや質問を表示 (X)
whiptail	V:87, I:995	71	シェルスクリプトからユーザーフレンリーなダイアログボックスを表示 (newt)
dialog	V:15, I:123	1222	シェルスクリプトからユーザーフレンリーなダイアログボックスを表示 (ncurses)
zenity	V:87, I:409	384	シェルスクリプトからグラフィカルなダイアログボックスを表示 (gtk2.0)
ssft	V:0, I:0	75	シェルスクリプトフロントエンドツール (gettext を使った zenity や kdialog や dialog のラッパー)
gettext	V:48, I:312	5843	"/usr/bin/gettext.sh": メッセージ翻訳

Table 12.9: ユーザーインターフェースプログラムのリスト

12.1.7 シェルスクリプトダイアログ

簡単なシェルプログラムのユーザーインターフェースは、echo や read コマンドを使った退屈な相互作用からいわゆる対話 (dialog) プログラム等の一つを使うことでよりよい相互作用になります。

12.1.8 zenity を使うシェルスクリプト例

dvdisaster(1) によって RS02 データーを補足した ISO イメージを生成する簡単なスクリプトの例を次に示します。

```
#!/bin/sh -e
# gmkrs02 : Copyright (C) 2007 Osamu Aoki <osamu@debian.org>, Public Domain
#set -x
error_exit()
{
  echo "$1" >&2
  exit 1
}
# Initialize variables
DATA_ISO="$HOME/Desktop/iso-$$.img"
LABEL=$(date +%Y%m%d-%H%M%S-%Z)
if [ $# != 0 ] && [ -d "$1" ]; then
  DATA_SRC="$1"
else
  # Select directory for creating ISO image from folder on desktop
  DATA_SRC=$(zenity --file-selection --directory  \
    --title="Select the directory tree root to create ISO image") \
    || error_exit "Exit on directory selection"
fi
# Check size of archive
xterm -T "Check size $DATA_SRC" -e du -s $DATA_SRC/*
SIZE=$(($(du -s $DATA_SRC | awk '{print $1}')/1024))
if [ $SIZE -le 520 ] ; then
  zenity --info --title="Dvdisaster RS02" --width 640  --height 400 \
    --text="The data size is good for CD backup:\\n $SIZE MB"
elif [ $SIZE -le 3500 ]; then
  zenity --info --title="Dvdisaster RS02" --width 640  --height 400 \
    --text="The data size is good for DVD backup :\\n $SIZE MB"
else
  zenity --info --title="Dvdisaster RS02" --width 640  --height 400 \
    --text="The data size is too big to backup : $SIZE MB"
  error_exit "The data size is too big to backup :\\n $SIZE MB"
fi
# only xterm is sure to have working -e option
```

```
# Create raw ISO image
rm -f ”$DATA_ISO” || true
xterm -T ”genisoimage $DATA_ISO” \
  -e genisoimage -r -J -V ”$LABEL” -o ”$DATA_ISO” ”$DATA_SRC”
# Create RS02 supplemental redundancy
xterm -T ”dvdisaster $DATA_ISO” -e  dvdisaster -i ”$DATA_ISO” -mRS02 -c
zenity --info --title=”Dvdisaster RS02” --width 640  --height 400 \
  --text=”ISO/RS02 data ($SIZE MB) \\n created at: $DATA_ISO”
# EOF
```

デスクトップに”/usr/local/bin/gmkrs02 %d” のようなコマンド設定をしたローンチャを作るのも面白いかもしれません。

12.2 Make

Make はプログラムのグループを管理するためのユーティリティーです。make(1) を実行すると、make は”Makefile” というルールファイルを読み、ターゲットが最後に変更された後で変更された前提ファイルにターゲットが依存している場合やターゲットが存在しない場合にはターゲットを更新します。このような更新は同時並行的にされるかもしれません。

ルールファイルのシンタックスは次です。

```
b’’ タ b’’b’’ ー b’’b’’ ゲ b’’b’’ ッ b’’b’’ ト b’’: [ b’’ 前 b’’b’’ 提 b’’ ... ]
 [TAB]  command1
 [TAB]  -command2 # b’’ エ b’’b’’ ラ b’’b’’ ー b’’b’’ 無 b’’b’’ 視 b’’
 [TAB]  @command3 # b’’ エ b’’b’’ コ b’’b’’ ー b’’b’’ 抑 b’’b’’ 制 b’’
```

上記で、”[TAB]” は TAB コードです。各行は make による変数置換後シェルによって解釈されます。スクリプトを継続する行末には”\” を使います。シェルスクリプトの環境変数のための”$” を入力するためには”$$” を使います。

ターゲットや前提に関するインプリシット (暗黙) ルールは、例えば次のように書けます。

```
%.o: %.c header.h
```

上記で、ターゲットは”%” という文字を (1 つだけ) 含んでいます。”%” は実際のターゲットファイル名の空でないいかなる部分文字列ともマッチします。前提もまた同様にそれらの名前が実際のターゲットファイル名にどう関連するかを示すために”%” を用いることができます。

自動変数	変数値
$@	ターゲット
$<	最初の前提条件
$?	全ての新規の前提条件
$^	全ての前提条件
$*	”%” はターゲットパターンの軸にマッチします

Table 12.10: make の自動変数のリスト

変数展開	説明
foo1 := bar	一回だけの展開
foo2 = bar	再帰的展開
foo3 += bar	後ろに追加

Table 12.11: make 変数の展開のリスト

”make -p -f/dev/null” を実行して自動的な内部ルールを確認下さい。

12.3 C

C プログラム言語で書かれたプログラムをコンパイルする適正な環境を次のようにして設定できます。

```
# apt-get install glibc-doc manpages-dev libc6-dev gcc build-essential
```

GNU C ライブラリーパッケージである libc6-dev パッケージは、C プログラム言語で使われるヘッダーファイルやライブラリールーチンの集合である C 標準ライブラリーを提供します。

C のリファレンスは以下を参照下さい。

- ”info libc” (C ライブラリー関数リファレンス)
- gcc(1) と”info gcc”
- 各 C ライブラリー関数名 (3)
- Kernighan & Ritchie 著, ”The C Programming Language”, 第 2 版 (Prentice Hall)

12.3.1 単純な C プログラム (gcc)

簡単な例の”example.c” は”libm” ライブラリーを使って実行プログラム”run_example” に次のようにしてコンパイル出来ます。

```
$ 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、”-lm” は sqrt(3) のために libc6 パッケージで提供されるライブラリー”/usr/lib/libm.so” をリンクするのに必要です。実際のライブラリーは”/lib/” 中にあるファイル名”libm.so.6” で、それは”libm-2.7.so” にシムリンクされています。

出力テキスト中の最後のパラメーターを良く見ましょう。”%10s” が指定されているにもかかわらず 10 文字以上あります。

上記のオーバーラン効果を悪用するバッファーオーバーフロー攻撃を防止のために、sprintf(3) や strcpy(3) 等の境界チェック無しのポインターメモリー操作関数の使用は推奨できません。これに代えて snprintf(3) や strncpy(3) を使います。

12.4 デバグ

デバグは重要なプログラム活動です。プログラムのデバグをどうしてするかを知っていることで、あなたも意味あるバグリポートを作成できる良い Debian ユーザーになれます。

12.4.1 基本的な gdb 実行

Debian 上の第一義的デバッガは、実行中のプログラムを検査できるようにする gdb(1) です。

gdb と関連プログラムを次のようにインストールしましょう。

```
# apt-get install gdb gdb-doc build-essential devscripts
```

gdb の良い入門書は”info gdb” とかネット上に色々あります。次は gdb(1) を”-g” を使ってデバグ情報を付けてコンパイルされた”program” に使う簡単な例です。

```
$ gdb program
(gdb) b 1                # set break point at line 1
(gdb) run args           # run program with args
(gdb) next               # next line
...
(gdb) step               # step forward
...
(gdb) p parm             # print parm
...
(gdb) p parm=12          # set value to 12
...
(gdb) quit
```

ティップ
多くの gdb(1) コマンドは省略できます。タブ展開はシェル同様に機能します。

12.4.2 Debian パッケージのデバグ

Debian システムではインストールされたバイナリーはデフォールトでストリップされているべきなので、通常のパッケージではほとんどのデバグシンボルが削除されています。gdb(1) を使って Debian パッケージをデバグするには、対応する *-dbg か *-dbgsym パッケージをインストールする必要があります (例えば libc6 の場合 libc6-dbg、coreutils の場合 coreutils-dbgsym)。

旧スタイルのパッケージは、対応する *-dbg パッケージを提供します。それは Debian の main アーカイブとの中に通常のパッケージと一緒に置かれます。新スタイルのパッケージは場合によってはビルド時に *-dbgsym パッケージを自動生成し、それらのデバグパッケージは別にして debian-debug アーカイブに置きます。詳細は Debian Wiki の記事を参照ください。

デバグしようとしているパッケージに *-dbg パッケージもその *-dbgsym パッケージも無い場合は、次のようにしてリビルドした後でインストールする必要があります。

```
$ mkdir /path/new ; cd /path/new
$ sudo apt-get update
$ sudo apt-get dist-upgrade
$ sudo apt-get install fakeroot devscripts build-essential
$ apt-get source package_name
$ cd package_name*
$ sudo apt-get build-dep ./
```

必要に応じてバグを修正します。

例えば次のように、既存パッケージを再コンパイルする時は”+debug1” を後ろに付けたり、リリース前のパッケージをコンパイルする時は”~pre1” を後ろに付けたりと、正規の Debian バージョンとかち合わないようにパッケージバージョンを増やします。

```
$ dch -i
```

次のようにしてデバグシンボル付きでパッケージをコンパイルしてインストールします。

```
$ export DEB_BUILD_OPTIONS="nostrip noopt"
$ debuild
$ cd ..
$ sudo debi package_name*.changes
```

パッケージのビルドスクリプトを確認して、バイナリーのコンパイルに確実に"CFLAGS=-g -Wall" が使われているようにします。

12.4.3 バックトレースの収集

プログラムがクラッシュするのに出会った場合に、バックトレース情報をバグレポートに切り貼りして報告するのは良い考えです。

バックトレースは次のような段取りで得られます。

- gdb(1) の下でプログラム実行します。
- クラッシュを再現します。
  - gdb プロンプトに落ちて戻るようになります。
- gdb プロンプトで"bt" とタイプします。

プログラムがフリーズした場合には、gdb を実行しているターミナルで Ctrl-C を押すことでプログラムをクラッシュさせて gdb プロンプトが得られます。

ティップ
しばしば、一番上数行が"malloc()" か"g_malloc()" 中にあるバックトレースを見かけます。こういったことが起こる場合は、大体あまりあなたのバックトレースは役に立ちません。有用な情報を見つけるもっとも簡単な方法は環境変数"$MALLOC_CHECK_" の値を 2 と設定することです (malloc(3))。gdb を実行しながらこれを実行するには次のようにします。

```
$ MALLOC_CHECK_=2 gdb hello
```

12.4.4 上級 gdb コマンド

コマンド	コマンド目的の説明
(gdb) thread apply all bt	マルチスレッドプログラムの全てのスレッドのバックトレースを取得
(gdb) bt full	関数コールのスタック上に来たパラメーターを取得
(gdb) thread apply all bt full	異常のオプションの組み合わせでバックトレースとパラメーターを取得
(gdb) thread apply all bt full 10	無関係の出力を切り最後の 10 のコールに関するバックトレースとパラメーターを取得
(gdb) set logging on	gdb アウトプットをファイルに書き出す (デフォールトは"gdb.txt")

Table 12.12: 上級 gdb コマンドのリスト

12.4.5 X エラーのデバグ

GNOME プログラム `preview1` が X エラーを受けると、次のようなメッセージが見つかります。

```
The program 'preview1' received an X Window System error.
```

このような場合には、プログラムを"`--sync`" 付きで実行して、バックトレースを得るために"`gdk_x_error`" 関数上で停止するようにしてみましょう。

12.4.6 ライブラリーへの依存の確認

次のように ldd(1) を使ってプログラムのライブラリーへの依存関係をみつけだします。

```
$ 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)
```

`chroot` された環境下で ls(1) が機能するには、上記ライブラリーがあなたの `chroot` された環境内で利用可能である必要があります。

項9.3.6を参照下さい。

12.4.7 メモリーリーク検出ツール

Debian にはメモリーリークを検出するプログラムがいくつか利用可能です。

パッケージ	ポプコン	サイズ	説明
`libc6-dev`	V:249, I:620	14357	mtrace(1): glibc 中の malloc デバッグ機能
`valgrind`	V:6, I:46	80378	メモリーデバッガとプロファイラ
`electric-fence`	V:0, I:5	70	malloc(3) デバッガ
`leaktracer`	V:0, I:3	57	C++ プログラム用のメモリーリーク追跡ソフト
`libdmalloc5`	V:0, I:3	393	メモリーアロケーションのデバグ用ライブラリー

Table 12.13: メモリーリーク検出ツールのリスト

12.4.8 静的コード分析ツール

静的コード分析用の lint のようなツールがあります。

12.4.9 バイナリーのディスアッセンブリー

次のように objdump(1) を使ってバイナリーコードをディスアッセンブルできます。

```
$  objdump -m i386 -b binary -D /usr/lib/grub/x86_64-pc/stage1
```

注意
gdb(1) は対話的にコードをディスアッセンブルするのに使えます。

パッケージ	ポプコン	サイズ	説明
splint	V:0, I:4	2315	C プログラムを静的にバグのチェックするためのツール
flawfinder	V:0, I:0	181	C/C++ ソースコードを検査してセキュリティーの脆弱性を探すツール
perl	V:610, I:992	705	静的コードチェックソフト付きのインタープリタ: B::Lint(3perl)
pylint	V:2, I:13	1371	Python コード静的チェックソフト
weblint-perl	V:0, I:1	32	HTML 用のシンタックス最小限の文体チェックソフト
linklint	V:0, I:0	344	高速リンクチェックソフトとウェッブサイトメンテツール
libxml2-utils	V:22, I:246	182	XML ファイルを検証する xmllint(1) を含むユーティリティー

Table 12.14: 静的コード分析ツールのリスト

12.5 Flex —改良版 Lex

Flex はLex 互換の高速字句解析生成ソフトです。

flex(1) の入門書は”`info flex`” の中にあります。

自分で作った”`main()`” と”`yywrap()`” を供給する必要があります。そうでない場合にはあなたの flex プログラムは次のようでなければライブラリー無しにコンパイル出来ません。これというのは”`yywrap`” はマクロで、”`%option main`” とすると”`%option noyywrap`” が暗示的に有効になるからです。

```
%option main
%%
.|\n    ECHO ;
%%
```

上記の代わりにとして、cc(1) のコマンドラインの最後に (ちょうど AT&T-Lex が”`-ll`” 付きであるように) ”`-lfl`” リンカーオプションを使いコンパイルすることが出来ます。この場合、”`%option`” は必要なくなります。

12.6 Bison —改良版 Yacc

Yacc 互換の前方参照可能な LR パーサーとか LALR パーサー生成ソフトは、いくつかのパッケージによって Debian 上で提供されています。

パッケージ	ポプコン	サイズ	説明
bison	V:9, I:103	2815	GNU LALR パーサー生成ソフト
byacc	V:0, I:6	160	Berkeley LALR パーサー生成ソフト
btyacc	V:0, I:0	243	byacc に基づいたバックトラッキング機能付きパーサー生成ソフト

Table 12.15: Yacc 互換の LALR パーサー生成ソフトのリスト

bison(1) の入門書は”`info bison`” の中にあります。

あなた自身の”`main()`” と”`yyerror()`” を供給する必要があります。”`main()`” は、しばしば Flex によって提供される”`yylex()`” を呼び出す”`yyparse()`” を呼び出します。

```
%%

%%
```

12.7 Autoconf

Autoconf は自動的にソフトウエアーのソースコードパッケージを GNU のビルドシステムを使って種々様々な Unix 的システムに適応させるためのシェルスクリプトを作成するツールです。

autoconf(1) は”`configure`” という設定プログラムを作成します。”`configure`” は”`Makefile.in`” を雛形として使って自動的にカスタム化した”`Makefile`” を作成します。

12.7.1 プログラムをコンパイルとインストール

警告
システムファイルをあなたがコンパイルしたプログラムでインストールする時に上書きしてはいけません。

Debian は”`/usr/local/`” とか”`/opt`” 中のファイルに触れません。プログラムをソースからコンパイルする場合、Debian とかち合わないようにそれを”`/usr/local/`” の中にインストールします。

```
$ cd src
$ ./configure --prefix=/usr/local
$ make
$ make install # this puts the files in the system
```

12.7.2 プログラムのアンインストール

オリジナルのソースを保有し、それが autoconf(1)/automake(1) と使用しあなたがそれをどう設定したかを覚えているなら、次のように実行してソフトウエアーをアンイストールします。

```
$ ./configure ''all-of-the-options-you-gave-it''
# make uninstall
```

この代わりに、”`/usr/local/`” の下にだけインストールプロセスがファイルを置いたことが絶対に確実でそこに重要なものが無いなら、次のようにしてその内容を消すことが出来ます。

```
# find /usr/local -type f -print0 | xargs -0 rm -f
```

どこにファイルがインストールされるか良く分からない場合には、`checkinstall` パッケージにある checkinstall(8) を使いアンインストールする場合クリーンなパスとなるようにすることを考えましょう。これは”`-D`” オプションを使うと Debian パッケージを作成できます。

12.8 究極の短い Perl スクリプト

どんな AWK スクリプトでも a2p(1) を使えば自動的に Perl に書き換えられますが、1 行 AWK スクリプトから 1 行 Perl スクリプトへの変換は手動変換するのが最良です。

次の AWK スクリプト断片を考えます。

```
awk '($2==”1957”) { print $3 }' |
```

これは次の数行のどれとも等価です。

```
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"' |
```

```
perl -lane 'print$F[2]if$F[1]eq+1957' |
```

最後のスクリプトは謎々状態です。Perl の次の機能を利用しています。

- ホワイトスペースはオプション。
- 数字から文字列への自動変換が存在します。

コマンドラインオプションに関しては perlrun(1) を参照下さい。もっとクレージーな Perl スクリプトに関しては、Perl ゴルフが面白いです。

12.9 ウェッブ

基本的な対話式動的ウェッブページは次のようにして作られます。

- 質問 (クエリー) はブラウザーのユーザーに HTML フォームを使って提示されます。
- フォームのエントリーを埋めたりクリックすることによって次の符号化されたパラメーター付きの URL 文字列をブラウザーからウェッブサーバーに送信します。
  - "`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`"
- URL 中の"%nn" は 16 進数で nn の値の文字と置き換えられます。
- 環境変通は次のように設定されます: "`QUERY_STRING="VAR1=VAL1 VAR2=VAL2 VAR3=VAL3"`".
- ウェッブサーバー上の CGI プログラム ("`program.*`" のいずれでも) が環境変数"`$QUERY_STRING`" とともに起動されます。
- CGI プログラムの `STDOUT` (標準出力) がウエブブラウザーに送られ対話式の動的なウェッブページとして表示されます。

セキュリティー上、CGI パラメーターを解釈する手作りの急ごしらえのプログラムは作らない方が賢明です。Perl や Python にはこのための確立したモジュールが存在します。PHP はこの様な機能とともに提供されます。クライアントでのデーターのストレージの必要がある場合、HTTP クッキーが使われます。クライアントサイドのデーター処理が必要な場合、Javascript が良く使われます。

詳しくは、Common Gateway Interface や The Apache Software Foundation や JavaScript を参照下さい。

http://www.google.com/search?hl=en&ie=UTF-8&q=CGI+tutorial を URL として直接ブラウザーのアドレスに入れ Google で"CGI tutorial" を検索するとグーグルサーバー上の CGI スクリプトが動いているのを観察する良い例です。

12.10 ソースコード変換

ソースコード変換するプログラムがあります。

パッケージ	ポプコン	サイズ	キーワード	説明
perl	V:610, I:992	705	AWK → PERL	AWK から PERL へのソースコード変換シフト: a2p(1)
f2c	V:0, I:6	442	FORTRAN → C	FORTRAN 77 から C/C++ へのソースコード変換ソフト: f2c(1)
intel2gas	V:0, I:0	178	intel → gas	NASM (Intel フォーマット) から GNU Assembler (GAS) への変換ソフト

Table 12.16: ソースコード変換ツールのリスト

12.11 Debian パッケージ作成

Debian パッケージを作りたい場合には、次を読みましょう。

- 基本的なパッケージシステムの理解には第2章
- 基本的なポーティングプロセスの理解のために、項2.7.13
- 基本的な chroot 技術の理解のために、項9.10.4
- debuild(1) と pbuilder(1) と pdebuild(1)
- リコンパイルとデバグは項12.4.2
- チュートリアルとして Debian New Maintainers' Guide (maint-guide パッケージ)
- Debian Developer's Reference (developers-reference パッケージ)
- Debian ポリシーマニュアル (debian-policy パッケージ)
- Guide for Debian Maintainers (debmake-doc パッケージ)

debmake や dh-make や dh-make-perl 等のパッケージングを補助するパッケージがあります。

Appendix A

補遺

以下が本文書の背景です。

A.1 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ーのための非常にパワフルなコンピューター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。非 PostScript プリンタが接続された LPR プリンタの設定がこんなつまずく点の良い例でした。(最近のインストレーションでは CUPS システムが使われるのでもうこの様な問題はありません。)

” 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することが難しいものです。また、HOWTO や mini-HOWTO と呼ばれるリファレンスもあります。これらは理解はしやすいのですが、詳細過ぎて全体像を失いがちです。ちょっと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は骨が折れることが時々あります。

この”Debian リファレンス (第 2.77 版)” が Debian 迷路の真っ只中にいる皆様にとって解決の糸口となることを望みます。

A.2 著作権の経緯

Debian リファレンスは青木修 <osamu at debian dot org> が個人用システム管理メモとして書き始められました。多くの内容が debian-user メーリングリストや他の Debian のリソースから得られた知識由来です。

当時 Debian Documentation Project で非常にアクティブであった、Josip Rodin 氏の助言に従い、DDP 文書の一部として”Debian リファレンス (第 1 版、2001-2007)” を作りました。

6 年経った時点で、青木はオリジナルの”Debian リファレンス (第 1 版)” が時代遅れとなっている事に気づき多くの内容を書き換え始めました。新たな”Debian リファレンス (第 2 版)” が 2008 年にリリースされました。

チュートリアルの内容はその内容とインスピレーションを次から得ました。

- ”Linux User’s Guide” Larry Greenfield 著 (1996 年 12 月)
  - ”Debian Tutorial” によって陳腐化
- ”Debian Tutorial” Havoc Pennington 著。(1998 年 12 年 11 日)
  - Oliver Elphick と Ole Tetlie と James Treacy と Craig Sawyer と Ivan E. Moore II による一部著作
  - ”Debian GNU/Linux: Guide to Installation and Usage” によって陳腐化
- ”Debian GNU/Linux: Guide to Installation and Usage” John Goerzen and Ossama Othman 著 (1999 年)

– ”Debian リファレンス (第 1 版)” によって陳腐化

パッケージやアーカイブに関する記述はそのオリジンやインスピレーションの一部を次に遡ることができます。

- ”Debian FAQ” (Josip Rodin が維持していた 2002 年 3 月版)

他の内容はそのオリジンやインスピレーションを次に遡ることができます。

- ”Debian リファレンス (第 1 版)” 青木修著 (2001 年〜2007 年)
  – 2018 年のより新しい”Debian リファレンス (第 2 版)” によって陳腐化

以前の”Debian リファレンス (第 1 版)” は次によって作られました。

- ネットワーク設定に関する大部分の内容は Thomas Hood が寄稿
- X と VCS に関連するかなりの内容は Brian Nelson が寄稿
- ビルドスクリプトや多くの内容に関する訂正で Jens Seidel が寄与
- David Sewell による徹底的な校正
- 翻訳者やコントリビューターやバグ報告者達による多くの寄与

Debian システム上の多くのマニュアルページや info ページが本文書を書く上での第一義的参照情報として使われました。青木修が公正な使用と考える範囲内で、それらの多くの部分、特にコマンドの定義が、本文書の文体と目的に合うように注意深い編集をした後、断片的文言として使われました。

gdb デバッガーに関する記述は Arii Pollak と Loïc Minier と Dafydd Harries の了承のもと backtrace に関する Debian wiki の内容を拡張して使いました。

既に上記で触れた項目を除く現在の”Debian リファレンス (第 2.77 版)” (2021-01-10 06:32:51 UTC) の内容はほとんど私自身の仕事です。これらはコントリビュータ – によっても更新されています。

”Debian リファレンス (第 1 版)” は、角田慎一さんがすべて日本語訳しました。

”Debian リファレンス (第 2 版)” は、英文原著者の青木修自身がすべてを日本語訳しました。その際に”Debian リファレンス (第 1 版)” から内容が比較的変更されていない「第 1 章 GNU/Linux チュートリアル」等では、角田さんの旧訳文を青木が文体や内容を調整した上で一部再利用させて頂きました。

著者である青木修は本文書を世に送ることにご助力戴いた皆様に感謝いたします。

A.3 文書のフォーマット

英語のオリジナル文書のソースは AsciiDoc のテキストファイルを用いて書かれます。AsciiDoc は直接 XML を書くより手間がかからずテーブルを分かりやすいフォーマットで入力出きるので便利という理由で使われています。XML と PO ファイルを真のソースファイルと考えて下さい。ビルドスクリプトによって DocBook XML ソースに変換され、更に自動的に生成されるデーターを埋め込み最終的な DocBook XML ソースとされます。この最終的な DocBook XML ソースは HTML と epub とプレーンテキストと PostScript と PDF に変換できます。(配布時には一部フォーマットが無効化されているかもしれません。)