Debian クイックリファレンス

Osamu Aoki (青木 修) <osamu\#at\#debian.org>
翻訳: 角田 慎一 <tsuno\#at\#ngy.1st.ne.jp>
'著者' on page 27

CVS, 2007年 1 月 18日 木曜日 11時54分39秒 UTC時間

概要

本Debian クイックリファレンス (http://qref.sourceforge.net/)はクイック参考書として、Debian システムへの簡潔な導入を提供します。本書はDebian リファレンス (http://qref.sourceforge.net/)からの抜粋です。

著作権表示

Copyright (c) 2001–2005 by Osamu Aoki <osamu#at#debian.org>.

This document may be used under the terms of the GNU General Public License version 2 or higher. (http://www.gnu.org/copyleft/gpl.html)

Permission is granted to make and distribute verbatim copies of this document provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this document under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this document into another language, under the above conditions for modified versions, except that this permission notice may be included in translations approved by the Free Software Foundation instead of in the original English.

目次

章 1

序文

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

1.1 本文書の慣習

この “Debian クイックリファレンス” は短い `bash` シェルコマンドを通じて 情報を供給します。

- Unix スタイルな **マニュアルページ** への参照は `bash(1)` のような形式で与えられます。
- GNU の **TEXINFO ページ** への参照は `info libc` のような形式で与えられます。

1.2 Debian ディストリビューションの基礎

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

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

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

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

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

現在使用しているディストリビューションがリリースしたバージョンよりも新しいバージョンのパッケージを利用したい場合、‘`stable`, `testing`, 又は `unstable` ディストリビューションへのアップグレード’ on page 3 に記述しているようにより新しいディストリビューションに更新することもできますし、選択したパッケージのみを更新することもできます。そのパッケージが簡単にはアップグレードできない場合には、‘`stable` システムへのパッケージ移植’ on page 20 に記述しているようにバックポートしたくなるかもしれません。

章 2

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

2.1 Potato から Woody へのアップグレード

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

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

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

次にシステムの残りの部分を Woody にアップグレードします。

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

2.2 アップグレードの準備

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

まず stable 用のリポジトリのクリーンなリストを取得します。

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

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

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

/etc/apt/sources.list と /etc/apt/preferences のチューニング技術については 'Debian パッケージ管理の基礎' on page 8 をご覧ください。

2.3 アップグレード

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

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

基本的なことは 'Debian パッケージ管理' on page 7 をご覧ください。そして、問題に 直面した場合は 'APT アップグレードのトラブルシュート' on page 13 をご覧ください。

2.3.1 dselect を使用する

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

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

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

```
# dselect install
```

dselect の install 作業中、パッケージの設定に関する質問に いくつか答える必要があるでしょう。ですから、この過程のために時間と ノート を用意しておきましょう。‘dselect’ on page 10 をご覧ください。

dselect を使用してください。 **常に動作します :)**

章 3

Debian パッケージ管理

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

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

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

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

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

chroot を用いる手法は、 システムの安定性と最新版のソフトウエアへのアクセスの両方を確保できる ために、望ましい手法です。

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

3.1 イントロダクション

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

3.1.1 主要なパッケージ管理ツール

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

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

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

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

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

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

3.1.2 便利なツール

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

3.2 Debian パッケージ管理の基礎

'アップグレードの準備' on page 3 に記述されているように sources.list を設定しましょう。 [1]

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

3.2.1 task のインストール

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

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

```
dselect update
```

を起動させる必要がある ことに注意してください。

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

3.2.2 aptitude

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

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

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

キーストローク	アクション
F10	メニュ
?	キーストロークヘルプを表示
u	パッケージアーカイブ情報を更新
+	パッケージをアップグレードするか新たにインストールするとしてマークする
-	パッケージを削除するとしてマークする (設定はそのまま)
_	パッケージを purge するとしてマークする (設定も削除)
=	パッケージを hold する
U	全てのアップグレード可能なパッケージをアップグレードするとしてマークする
g	選択パッケージのダウンロードおよびインストール
q	現在のスクリーンを終了し、変更点を保存する
x	現在のスクリーンを終了し、変更点を無視する
Enter	パッケージの情報を閲覧
C	パッケージの変更履歴を閲覧

l	表示されたパッケージの制限を変更
/	単語検索
\	最後の検索を繰り返す

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

aptitude の他の利点には次があります。

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

3.2.3 dselect

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

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

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

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

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

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

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

3.2.4 APT を用いてディストリビューションを追いかける

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

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

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

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

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

新しいリリースの **package** を古いリリースのものに ダウングレードすることは Debian では公式にサポートされていません。 しかしながら、新しいバージョンがうまく動かないので、うまく動くバージョンの パッケージを再インストールする目的で特定のパッケージをダウン グレード する必要があることが判明するかもしれません。 これらの以前 インストールしていたパッケージファイルはローカルの /var/cache/apt/archives/ 又は リモートでは http://snapshot.debian.net/ に見つかるかもしれません。 ‘dpkg を用いたレスキュー’ on page 14 もご覧下さい。

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

3.2.5 aptitude, apt-get と apt-cache コマンド

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

- aptitude update (又は apt-get update)
  この作業により、リポジトリで利用可能なパッケージリストが更新されます。

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

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

- `apt-get dselect-upgrade`

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

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

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

- `aptitude install -t unstable` *package*

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

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

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

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

  パッケージ *foo bar ...* に関する情報をチェックします。

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

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

- `aptitude install` *foo bar-*

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

- `aptitude remove` *bar*

  bar パッケージを削除しますが、設定ファイルは削除しません。

- `aptitude purge` *bar*

  全ての設定ファイルと一緒に *bar* パッケージを削除します。

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

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

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

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

3.3 Debian で生き残るためのコマンド

本章の知識により、**永遠の** upgrade 生活をすごせます。 :)

3.3.1 Debian のバグをチェックし、助けを求める

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

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

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

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

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

そして次を実行してください。

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

3.3.2 APT アップグレードのトラブルシュート

パッケージの依存性問題は ‘アップグレード’ on page 4 に記述したように `unstable` 又は `testing` にアップグレードする場合に発生する可能性があります。 ほとんどの場合、これはそのパッケージがまだ得られないパッケージに Depends しているためです。 これらの問題は次の手順を用いて解決できます。

```
# aptitude dist-upgrade
```

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

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

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

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

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

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

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

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

3.3.3 dpkg を用いたレスキュー

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

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

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

3.3.4 パッケージ選択データの回復

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

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

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

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

3.3.5 /var のクラッシュ後のシステム回復

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

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

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

を実行します。これによりシステムが機能するようになるはずです。'パッケージ選択データの回復' on page 15 に記述している技術を用いてパッケージ選択データ の回復をはかどらせることができます。 ([FIXME]: 本手順はさらなる検証が必要。)

3.3.6 ブート不能なシステムにパッケージをインストール

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

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

そして設定を行い、問題を修正します。

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

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

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

3.3.7 dpkg コマンドが壊れた場合どうするか

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

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

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

3.4 Debian 涅槃コマンド

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

3.4.1 ファイルに関する情報

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

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

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

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

もしくは特別なパッケージコマンドを使います。

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

3.4.2 パッケージ情報

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

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

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

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

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

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

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

3.4.3 APT によりキーボードに触らずにインストール

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

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

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

'インストール済みパッケージの再設定' on page 18 に従い、特定のパッケージを後で設定することもできます。

3.4.4 インストール済みパッケージの再設定

次のコマンドを使ってインストール済みパッケージの再設定を行います。

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

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

特別な設定スクリプトを持つプログラムがいくつかあります。[3]

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

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

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

3.4.5 パッケージの削除及びパージ

設定ファイルを維持したままパッケージを削除します。

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

設定ファイルを含め、パッケージを削除します。

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

3.4.6 古いパッケージを hold する

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

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

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

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

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

次のコマンドにより hold されたパッケージのリストを表示できます。

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

3.4.7 stable/testing/unstable システムの混在

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

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

3.4.8 キャッシュされたパッケージファイルを取り除く

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

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

3.4.9 システム設定の記録/コピー

パッケージ選択ステータスのローカルコピーを取るには、次を実行します。

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

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

このファイルを他のコンピュータに転送し、これを用いてインストール可能です。

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

3.4.10 stable システムへのパッケージ移植

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

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

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

そしてソースを取得し、ローカルパッケージを作成します。

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

$ dpkg-buildpackage -rfakeroot

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

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

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

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

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

だけです。

3.4.11 ローカルパッケージのアーカイブ

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

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

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

代わりに、手早いけど汚いローカル deb リポジトリを次のように作れます。

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

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

3.4.12 alien バイナリパッケージへの変換又はインストール

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

3.4.13 自動でコマンドをインストールする

auto-apt はオンデマンドのパッケージインストールツールです。

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

3.4.14 インストールされたパッケージファイルを検証する

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

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

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

3.5 他の Debian 独特な点

3.5.1 **dpkg-divert** コマンド

Diversions というファイルにより、dpkg はファイルをインストールする時に本来意図していた場所ではなく、**退避した** 場所にインストールするようになります。**Diversions** は Debian パッケージスクリプトで衝突が起こりうるファイルを移動させるために使うことができます。システム管理者はパッケージの設定ファイルや他のファイルを (**conffiles** としてマークされていなければ) dpkg が新しいバージョンのパッケージをインストールする時にそれらのファイルを上書きしてしまわないようにするために diversion を使うことができます。.

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

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

3.5.2 `equivs` パッケージ

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

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

3.5.3 Alternative コマンド

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

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

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

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

好みの X Window 環境を設定するには、`update-alternatives` を `/usr/bin/x-session-manager` と `/usr/bin/x-window-manager` に適用します。

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

3.5.4 ランレベルの使い方

インストール直後は、ほとんどの Debian パッケージはサービスを ランレベル 2 から 5 まで起動するように設定します。 これゆえに、カスタマイズされていない Debian システムではランレベル 2, 3, 4, と 5 には何の違いがありません。Debian はローカルの管理者に任せてい

ます。 ランレベルのカスタマイズを ローカルの管理者に任せています。 これは他の有名な GNU/Linux ディストリビューションにより用いられている ランレベルの方法とは異なります。 ブートシーケンスの終わりで X ディスプレイマネージャが起動されないように ランレベル 2 で `xdm` や `gdm` を無効にしたい時が 来るかもしれません。 その時はランレベル 3 から起動するように変更できます。

3.5.5 デーモンサービスを無効にする

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

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

最近の Debian において、標準では X11 は TCP/IP 経由の遠隔接続を 許可していません。 SSH での X フォワードも無効にされています。

付 録 A

付録

A.1 著者

Debian クイックリファレンス はまず Osamu Aoki <osamu\#at\#debian.org> により 個人的なインストール メモとして書かれ、結局 “クイックリファレンス” と呼ばれました。コンテンツの多くは “debian-users” メーリングリストのアーカイブ から取られました。又、“Debian Installation Manual” および “Debian Release Notes” も参照されました。

Debian Documentation Project (http://www.debian.org/doc/ddp) (DDP) に関して 非常にアクティブな人物であり、現在の “The Debian FAQ” のメンテナである Josip Rodin の助言に従い、本文書は “Debian リファレンス” と改名し、 “The Debian FAQ” からリファレンス形式の内容としていくつかの章をマージしています。

本文書は次に挙げる QREF チームメンバーにより編集、翻訳と 拡張が行われています。

- 原作者
  - Osamu Aoki (青木 修) <osamu\#at\#debian.org> (leader: all contents)
- 英語版の校正及び追加の貢献
  - Esko Araj.rvi <edu\#at\#iki.fi> (etch アップデート)
  - Thomas Hood <jdthood\#at\#yahoo.co.uk> (ネットワーク関連)
  - Brian Nelson <nelson\#at\#bignachos.com> (特に X 関連)
  - David Sewell <dsewell\#at\#virginia.edu> (退任)
  - Jan Michael C Alonzo <jmalonzo\#at\#spaceants.net>
  - Daniel Webb <webb\#at\#robust.colorado.edu>
  - および全翻訳者からのフィードバック
- フランス語訳
  - Guillaume Erbs <gerbs\#at\#free.fr> (leader: fr)
  - Renald Casagraude <rcasagraude\#at\#interfaces.fr>
  - Jean-Pierre Delange <adeimantos\#at\#free.fr>
  - Daniel Desages <daniel\#at\#desages.com>
- イタリア語訳
  - Davide Di Lazzaro <mc0315\#at\#mclink.it> (leader: it)
- ポルトガル (ブラジル) 語訳
  - Paulo Rogerio Ormenese <pormenese\#at\#uol.com.br> (leader: pt-br)

  - Andre Luis Lopes <andrelop\#at\#ig.com.br>
  - Marcio Roberto Teixeira <marciotex\#at\#pop.com.br>
  - Rildo Taveira de Oliveira <to_rei\#at\#yahoo.com>
  - Raphael Bittencourt Simoes Costa <raphael-bsc\#at\#bol.com.br>
  - Gustavo Noronha Silva <kov\#at\#debian.org> (coordinator)
- スペイン語訳
  - Walter Echarri <wecharri\#at\#infovia.com.ar> (leader: es)
  - Jose Carreiro <ffx\#at\#urbanet.ch>
- ドイツ語訳
  - Jens Seidel <tux-master\#at\#web.de> (leader: de)
  - Willi Dyck <wdyck\#at\#gmx.net>
  - Stefan Schroeder <stefan\#at\#fkp.uni-hannover.de>
  - Agon S. Buchholz <asb\#at\#kefk.net>
- ポーランド語訳 PDDP (http://debian.linux.org.pl):
  - Marcin Andruszkiewicz
  - Mariusz Centka <mariusz.centka\#at\#debian.linux.org.pl>
  - Bartosz Feński <fenio\#at\#debian.linux.org.pl> (leader: pl)
  - Radosław Grzanka <radekg\#at\#debian.linux.org.pl>
  - Bartosz 'Xebord' Janowski
  - Jacek Lachowicz
  - Rafał Michaluk
  - Leonard Milcin, Jr.
  - Tomasz Z. Napierała <zen\#at\#debian.linux.org.pl>
  - Oskar Ostafin <cx\#at\#debian.linux.org.pl>
  - Tomasz Piękoś
  - Jacek Politowski
  - Mateusz Prichacz <mateusz\#at\#debian.linux.org.pl>
  - Marcin Rogowski
  - Paweł Różański
  - Mariusz Strzelecki
  - Krzysztof Ścierski
  - Przemysław Adam Śmiejek <tristan\#at\#debian.linux.org.pl>
  - Krzysztof Szynter
  - Mateusz Tryka <uszek\#at\#debian.linux.org.pl>
  - Cezary Uchto
  - Krzysztof Witkowski <tjup\#at\#debian.linux.org.pl>
  - Bartosz Zapałowski <zapal\#at\#debian.linux.org.pl>
- 中国語 (簡体字) 訳
  - Hao "Lyoo" LIU <iamlyoo\#at\#163.net>
  - Ming Hua <minghua\#at\#rice.edu>
  - Xiao Sheng Wen <atzlinux\#at\#163.com> (leader: zh-cn)
  - Haifeng Chen <optical.dlz\#at\#gmail.com>
  - Xie Yanbo <xieyanbo\#at\#gmail.com>
  - easthero <easthero\#at\#gmail.com>
- 中国語 (繁体字) 訳
  - Asho Yeh <asho\#at\#debian.org.tw> (leader: zh-tw)

  - Tang Wei Ching <wctang\#at\#csie.nctu.edu.tw> (ex-leader: zh-tw)
- 日本語訳
  - Shinichi Tsunoda (角田 慎一) <tsuno\#at\#ngy.1st.ne.jp> (leader: ja)
  - Osamu Aoki (青木 修) <osamu\#at\#debian.org>
- フィンランド語訳
  - Esko Araj.rvi <edu\#at\#iki.fi> (leader: fi)

A.2 保証

私は専門家では無いので、一般に Debian 又は Linux に関する完全な知識を 持っているふりをするつもりはありません。私が考慮するセキュリティは ホームユースにだけ適合しているかもしれません。

本文書は権威のあるガイドの置き換えにはなり得ません。

全ての保証を放棄します。全ての商標はそれぞれの商標保有者が所有しています。

A.3 フィードバック

本文書に対するコメントや追加は大歓迎です。 debian-reference パッケージあるいは翻訳版パッケージに 対して Debian BTS system (http://bugs.debian.org/) にメールを 送ってください。 reportbug を使うと簡単に詳細な bug report を提出できます。 また、<osamu\#at\#debian.org> (Osamu Aoki (http://people.debian.org/~osamu/)) に英語でメールを送るか、翻訳者に日本語でメールを送ることもできます。