# APT HOWTO (Obsolete Documentation)

Gustavo Noronha Silva <kov@debian.org>

1.7.6 - 2002 年 1 月

## 概要

この文書は、Debian のパッケージ管理ユーティリティである APT の働きについて、 ユーザに深く理解してもらうことを意図しています。その目的は、新しい Debian ユーザの生活を楽にしたり、シ ステム管理について理解を深めたいと願う人の手助けと なることです。Debian ユーザが得られるサポートを改善する目的で、Debian プロ ジェクトのために作成されました。

# 著作権表示

# 目次

# 章 1

# はじめに

初めに .tar.gz ありき。ユーザたちは、自分の GNU/Linux システムで使いたい プログラム毎にコンパイルしなければなりませんでした。Debian が作られた時、 マシンにインストールされたパッケージの管理機能を備えたシステムが必要だと 考えられました。`dpkg` というのが、そのシステムに与えられた名前です。 こうして有名な 'パッケージ' というものが、Debian に最初にもたらされたのです。 それは Red Hat が、独自の rpm システムを作ろうと決断した少し前のことでした。

すぐに新たなジレンマが、GNU/Linux 製作者たちの心中に生まれました。 彼らが必要としているのは、パッケージ間の依存関係を自動的に管理し、 アップグレート中でも設定ファイルに注意を払ってくれる、高速で、実用的な、 効率の高い方法だったのです。ここでもまた、Debian が道を切り開いて APT - Advanced Packaging Tool - を産み出しました。APT はそれから Conective によって rpm でも使えるよう移植され、それ以外のディストリビューションにも適合するように なりました。

本マニュアルでは、Conectiva による APT の移植である apt-rpm には触れません。 しかしその目的の為の本文書への “パッチ” は歓迎します。

# 章 2

# 基本的な設定

## 2.1 /etc/apt/sources.list ファイル

APT は操作の一部に、利用可能なパッケージの一覧である 'ソース' ファイルを 使用します。このファイルが /etc/apt/sources.list です。

このファイルには通常、以下のようなエントリがあります。

```
deb http://host/debian distribution section1 section2 section3
deb-src http://host/debian distribution section1 section2 section3
```

もちろん、上記は架空のものであって、そのままでは使えません。 各行の冒頭にある deb と deb-src は、アーカイブの種類を 示しています。つまり、通常使うコンパイル済のパッケージであるバイナリ パッケージ (deb)。それから、オリジナルのプログラムソースと Debian の コントロールファイル (.dsc) 、およびプログラムを 'Debian 化' するのに必要な変更点を含んだ diff.gz からなるソースパッケージ (deb-src) です。

Debian のデフォルトの sources.list は次のようになっています。

```
# See sources.list(5) for more information, especialy
# Remember that you can only use http, ftp or file URIs
# CDROMs are managed through the apt-cdrom tool.
deb http://http.us.debian.org/debian stable main contrib non-free
deb http://non-us.debian.org/debian-non-US stable/non-US main contrib non-fre
deb http://security.debian.org stable/updates main contrib non-free

# Uncomment if you want the apt-get source function to work
#deb-src http://http.us.debian.org/debian stable main contrib non-free
#deb-src http://non-us.debian.org/debian-non-US stable/non-US main contrib no
```

以上は Debian の基本インストールに必要な行です。最初の deb 行は、 オフィシャルのアーカイブを示しています。2 行目は non-US アーカイブ、3 行目は Debian セキュリティアップグレードのアーカイブです。

最後の 2 行はコメントアウトされて ('#' で始まって) いるので、apt-get は無視します。 これらは deb-src 行で、Debian のソースパッケージを指しています。 テストや再コンパイル目的でプログラムソースを頻繁にダウンロードするのなら、 これらの行のコメントを外してください。

/etc/apt/sources.list ファイルには、各種の行を混在させることができます。APT は http, ftp, file (ローカル ファイル、たとえばマウントされた ISO9660 ファイルシステム中のディレクトリ など), ssh など、異なる種類のアーカイブにも対応できます。

## 2.2 ローカルでの APT の使い方

多数の .deb パッケージをインストールする際に、依存関係を自動的に解決 するために APT を使いたくなる時があるでしょう。

そのためには .deb ファイルを収めるディレクトリを作ってください。 例えば:

```
mkdir /root/debs
```

次に /root ディレクトリ内に入って、空のファイルを作ってください。 名前は何でも構いません。これは、APT のレポジトリが "override" という 名で知られるファイルを必要とするからです。これは空でも構いませんが、 存在はしていなければなりません。この種のファイルを作るのには、次の コマンドが利用できます。

```
touch file
```

このファイルの中で、パッケージに付いてくるオプションを上書きするための オプションを定義することもできます。例は以下の通り:

```
package priority section
```

例の中で、package はパッケージ名、priority は low, medium, high の いずれか、section はそのパッケージが属するジャンルです。そのファイルは 空のままでも構いません。

さらに /root ディレクトリ内で次のコマンドを実行してください:

```
dpkg-scanpackages debs file | gzip > debs/Packages.gz
```

上記の例中、file は "override" ファイルを指定してください。 上記コマンドは、debs/Packages.gz という名のパッケージに関する 各種情報を含んだファイルを生成します。このファイルが APT に 使われます。最後に、パッケージを使用するため sources.list に 以下の行を追加してください。

```
deb file:/root debs/
```

この後は、普段通りに APT のコマンドを使うだけです。ソースのレポジトリも生成することができます。そのための手順は同じですが、.orig.tar.gz と .diff.gz ファイルがなければならず、生成されるファイルも Packages.gz の代わりに Sources.gz となります。生成用のプログラムも異なります。dpkg-scansources です。コマンドラインは次のようになります。

```
dpkg-scansources debs | gzip > debs/Sources.gz
```

dpkg-scansources には “override” ファイルが不要であることに注意してください。sources.list の記述は次のようになります。

```
deb-src file:/root debs/
```

## 2.3 sources.list ファイルに記述すべき最適なミラーサイトの決定: netselect, netselect-apt

主に始めたばかりのユーザが間違いなく抱く疑問があります -「sources.list に記述すべき Debian のミラーサイトはどれですか?」。どのミラーにするのかを決めるには、多くの方法があります。おそらく熟練者なら、いくつかのミラーに ping を打って時間を測るスクリプトを持っているでしょう。ですが我々のためにも、同じことをしてくれるプログラムがあります - **netselect** です。

netselect のインストールは、いつものようにするだけです:

```
apt-get install netselect
```

パラメータ無しで実行すると、ヘルプが表示されます。スペースで区切ったホスト (ミラー) のリストを付けて実行すると、一つのホストとそのスコアを返します。このスコアは、推定 ping 時間と hops 数 (接続要求が目的地に到達するまでに経由したホストの数)、逆比例する推定ダウンロード速度 (よって少ないほど可) などを計算して算出されます。コマンドが返すホストは、最少のスコアとなった 1 つだけです (-vv オプションを付ければ、全候補のスコアが出力されます)。以下の例を参照してください:

```
bash$ netselect ftp.debian.org http.us.debian.org ftp.at.debian.org download.
  365 ftp.debian.org.br
bash$
```

上記の意味は、netselect のパラメータとして渡したミラーの中で、ftp.debian.org.br が最適であり、スコアは 365 であるということです (注意してください!! これは私のコンピュータで実行したものであり、ネットワーク接続図は接続ポイントによって大きく異なります。このスコアは、必ずしも他のコンピュータでの結果と一致するわけではありません)。

さあ、netselect が見つけた最速のミラーを /etc/apt/sources.list ファイルに記述して('/etc/apt/sources.list ファイル' on page 3 参照)、'パッケージの管理' on page 9 にある tipsに従ってください。

**注意:** ミラーの一覧は、 http://www.debian.org/mirror/mirrors_full というファイルを 見れば、いつでもわかります。

バージョン 0.3 から、netselect パッケージには **netselect-apt** というスクリプトが含まれるようになりました。これは上記の プロセスを自動的に行なってくれるものです。ディストリビューションの 種類 (デフォルトは stable) をパラメータとして実行するだけで、最適な main と non-US のミラーが記述された sources.list が生成され、 今いるディレクトリ内に配置されます。以下は、stable ディストリ ビューション用の sources.list を生成する例です。

```
bash$ ls sources.list
ls: sources.list: File or directory not found
bash$ netselect-apt stable
(...)
bash$ ls -l sources.list
sources.list
bash$
```

**注記:** sources.list ファイルは、コマンドを 実行したディレクトリ内に生成されます。その後、/etc/apt/ ディレクトリに移動しなければなりません。

以降は'パッケージの管理' on page 9 にある tipsに従ってください。

## 2.4 sources.list ファイルに CD-ROM を追加する

APT を使ってパッケージをインストールしたり、システムを自動的にアップ グレードするのに CD-ROM を使いたいのなら、それを sources.list に記述することができます。そのためには、以下の例のように apt-cdrom プログラムを使用してください。

```
apt-cdrom add
```

ドライブに Debian の CD-ROM を入れておいてください。プログラムは CD-ROM をマウントし、ちゃんとした Debian の CD であればパッケージに 関する情報を探します。CD-ROM の設定がやや変わっている場合でも、以下の オプションを使うことができます。

```
-h            - ヘルプを表示
-d directory  - CD-ROM のマウント位置
-r            - 認識された CD-ROM の名前変更
-m            - マウントを行なわない
-f            - 高速モード、package ファイルのチェックを行なわない
-a            - 厳密スキャンモード
```

例えば次のように:

```
apt-cdrom -d /home/kov/mycdrom add
```

sources.list に追加することなしに、CD-ROM を識別させることもできます。

```
apt-cdrom ident
```

このプログラムは CD-ROM が `/etc/fstab` 内で適切に設定 されている場合のみ動作することに注意してください。

# 章 3

# パッケージの管理

## 3.1 利用可能なパッケージの一覧を更新する

どのパッケージがインストール済で、どれが未インストールか、どのパッケージがインストール可能かについての記録用に、パッケージシステムはプライベートなデータベースを使用します。apt-get プログラムは、ユーザがインストールを要求したパッケージを見つけたり、そのパッケージが適切に動作するために必要な追加パッケージを見つけるのに、このデータベースを使います。

このデータベースを更新するには、apt-get update というコマンドを使います。このコマンドは /etc/apt/sources.list に記述されているアーカイブ内のパッケージ一覧を探します。このファイルに関する詳しい情報は '/etc/apt/sources.list ファイル' on page 3 を参照してください。

可能なパッケージ更新についての情報を最新に保つために、このコマンドを定期的に実行するのは良いアイデアです。セキュリティ関連のアップデートはなおさらです。

## 3.2 パッケージのインストール

ようやく、皆さんが待ち望んでいたプロセスです! sources.list を用意し、利用可能なパッケージの一覧を更新したら、欲しいパッケージをインストールするには apt-get を実行するだけです。例えば次のように:

```
apt-get install xchat
```

APT は指定されたパッケージの最新バージョンをデータベース中に探し、sources.list に記載された対応するアーカイブから抽出します。指定されたパッケージが他のパッケージに依存している場合 – 以下の例のように –、APT は依存関係をチェックして必要なパッケージをインストールします。以下の例を参照してください:

```
[root]@[/] # apt-get install nautilus
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  bonobo libmedusa0 libnautilus0
The following NEW packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
0 packages upgraded, 4 newly installed, 0 to remove and 1  not upgraded.
Need to get 8329kB of archives. After unpacking 17.2MB will be used.
Do you want to continue? [Y/n]
```

nautilus パッケージは、上記に引用された共有ライブラリに 依存しています。そのため、APT はそれらをアーカイブからダウンロードします。 apt-get のコマンドラインにそれらのライブラリ名が指定されていれば、 APT は作業を継続していいか尋ねてきません - 関連全パッケージをインストール したいものと受け取るのです。

このことは、APT が確認を求めてくるのは依存関係を満たすために指定された 以外のパッケージをインストールする必要がある時だけ、ということを意味 しています。

ap-get には以下のオプションが使えます。

```
-h  ヘルプを表示
-d  ダウンロードのみ - インストールやアーカイブの解凍は行ないません
-f  完全度チェックに失敗しても、作業を継続しようとします
-s  何もしません。シミュレーションを行なうだけです
-y  全ての問いに対する回答が Yes とみなし、尋ねてきません
-u  アップグレードされるパッケージの一覧を表示します
```

インストールすべき複数のパッケージを 1 行中に指定できます。 ネットワークからダウンロードされたファイルは /var/cache/apt/archives/ ディレクトリに置かれ、 それからインストールされます。

同じコマンドライン内に、削除すべきパッケージも指定することができます。以下の 例のように、削除したいパッケージ名の直後に '-' を付けるだけです:

```
[root]@[/] # apt-get install nautilus gnome-panel-
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  bonobo libmedusa0 libnautilus0
The following packages will be REMOVED:
  gnome-applets gnome-panel gnome-panel-data gnome-session
The following NEW packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
0 packages upgraded, 4 newly installed, 4 to remove and 1  not upgraded.
Need to get 8329kB of archives. After unpacking 2594kB will be used.
Do you want to continue? [Y/n]
```

パッケージの削除に関する詳細は 'パッケージの削除' on page 11 を参照してください。

インストール済のパッケージが一部壊れてしまったり、同じバージョンの パッケージ中にあるファイルを再インストールしたいだけなら、以下の例の ように `--resinstall` オプションを使えます。

```
[root]@[/] # apt-get --reinstall install gdm
Reading Package Lists... Done
Building Dependency Tree... Done
0 packages upgraded, 0 newly installed, 1 reinstalled, 0 to remove and 1  not
Need to get 0B/182kB of archives. After unpacking 0B will be used.
Do you want to continue? [Y/n]
```

本マニュアル執筆時点で使った APT のバージョンは 0.5.3 であり、これは Debian の 'unstable' (`sid`) の最新版でした。このバージョンが インストールされていれば、追加機能を使うこともできます。特定の ディストリビューション内のパッケージをインストールするのに `apt-get install package/distribution` としたり、 `apt-get install package=version` のようなコマンドラインを 使えます。例えば:

```
apt-get install nautilus/unstable
```

この例では、あなたが 'stable' を使っていても 'unstable' ディストリ ビューション内の nautilus をインストールします。 'ディストリビューション' として指定できるのは `stable`, `testing`, `unstable` の 3 つです。

重要: Debian の 'unstable' 版は、Debian パッケージが最初に アップロードされる最新版です。このディストリビューションは、パッケージに 加えられたあらゆる変更を、些細なものから多数のパッケージやシステム全体に 影響を及ぼす大掛かりなものまで受け付けます。このため、経験の浅いユーザや 確固たる安定性が必要な人は、使用するべきではありません。

'testing' ディストリビューションは、安定性という点では 'unstable' よりも わずかに優れていますが、業務上のシステムには 'stable' ディストリビューション を使うべきです。

## 3.3 パッケージの削除

あるパッケージをもはや使いたくないのなら、APT を使ってシステムから 取り除くことができます。そのためには `apt-get remove package` と タイプするだけです。例えば:

```
[root]@[/] # apt-get remove gnome-panel
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be REMOVED:
  gnome-applets gnome-panel gnome-panel-data gnome-session
0 packages upgraded, 0 newly installed, 4 to remove and 1  not upgraded.
Need to get 0B of archives. After unpacking 14.6MB will be freed.
Do you want to continue? [Y/n]
```

上記の例でわかるように、APT は削除を要求したパッケージに依存している他の パッケージも削除するようにしてくれます。依存している他のパッケージを削除 せずに、あるパッケージを削除することは、APT には不可能です。

上記のように apt-get を使うとパッケージは削除されますが、 そのパッケージの設定ファイルがあれば完全な形でシステム内に残ります。 パッケージを完全に削除するには、次のようにしてください:

```
[root]@[/] # apt-get --purge remove gnome-panel
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be REMOVED:
  gnome-applets* gnome-panel* gnome-panel-data* gnome-session*
0 packages upgraded, 0 newly installed, 4 to remove and 1  not upgraded.
Need to get 0B of archives. After unpacking 14.6MB will be freed.
Do you want to continue? [Y/n]
```

パッケージ名の後の '*' に注意してください。これは各パッケージの設定 ファイルも併せて削除されることを示しています。

install 指定を行なった時のように、特定のパッケージに対する指示を remove と逆転させるための記号を使えます。削除の際に、'+' を パッケージ名の直後に付ければ、そのパッケージは削除の代わりにインストール されます。

```
[root]@[/] # apt-get --purge remove gnome-panel nautilus+
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
The following packages will be REMOVED:
  gnome-applets* gnome-panel* gnome-panel-data* gnome-session*
The following NEW packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
0 packages upgraded, 4 newly installed, 4 to remove and 1  not upgraded.
Need to get 8329kB of archives. After unpacking 2594kB will be used.
Do you want to continue? [Y/n]
```

apt-get の表示が、追加でインストールされるパッケージ (つまり、 インストールを要求したパッケージを適切に機能させるためにインストール しなければならないパッケージ)、削除されるパッケージ、インストールされる パッケージ (再び追加パッケージ) であることに注意してください。

## 3.4 パッケージのアップグレード

パッケージのアップグレードは、APT システムの偉大な成果です。 たった一つのコマンド - apt-get upgrade で実施可能です。この コマンドは同じディストリビューション中の

パッケージをアップグレード する際も、新たなディストリビューションにアップグレードする際にも 使えます。ですが後者の場合は apt-get dist-upgrade コマンドを 使用したほうがよいでしょう。詳細については '新リリースへのアップグレード' on page 13 を 参照してください。

このコマンドには -u オプションを付けて実行されると便利です。 このオプションにより、APT はアップグレードされる全パッケージの一覧を 表示します。もし付けなければ、何が起きているのかわからないまま アップグレードすることになります。APT は各パッケージの最新バージョンを ダウンロードし、適切な順番でインストールします。アップグレードを実行する 前に、必ず apt-get update を実行しておくことが重要です。 '利用可能なパッケージの一覧を更新する' on page 9 を参照してください。以下は例です:

```
[root]@[/] # apt-get -u upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages have been kept back
  cpp gcc lilo
The following packages will be upgraded
  adduser ae apt autoconf debhelper dpkg-dev esound esound-common ftp indent
  ipchains isapnptools libaudiofile-dev libaudiofile0 libesd0 libesd0-dev
  libgtk1.2 libgtk1.2-dev liblockfile1 libnewt0 liborbit-dev liborbit0
  libstdc++2.10-glibc2.2 libtiff3g libtiff3g-dev modconf orbit procps psmisc
29 packages upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
Need to get 5055B/5055kB of archives. After unpacking 1161kB will be used.
Do you want to continue? [Y/n]
```

プロセスは非常に単純です。最初の 2, 3 行で、apt-get がいくつかの パッケージをkept back (保持) すると言っていることに注意して ください。これは、何らかの理由でインストールされない新バージョンのパッケージ があることを意味しています。考えられる理由としては、壊れた依存関係 (依存先のパッケージの適正バージョンがダウンロードできない) や、新たな依存関係 (依存するパッケージが新たに登場した) などがあります。

最初のケースを手際よく解決する方法はありません。2 番目のケースに対しては、 問題となっているパッケージに対して apt-get install を実行するだけで 充分であり、依存対象となっているパッケージをダウンロードしてくれます。 さらに手際のよい解決方法は dist-upgrade を使用することです。 '新リリースへのアップグレード' on page 13 を参照してください。

## 3.5 新リリースへのアップグレード

APT の本機能により、インターネットあるいは新 CD (購入したものか、ISO イメージとしてダウンロードしたもの) を使って、全 Debian システムを一度にアップグレード することができます。

さらにインストール済パッケージ間の関係に変化があった際にも使います。 apt-get upgrade では、これらのパッケージは何も変更されないまま 保持 (kept back) されるのです。

例えば、Debian の安定版リビジョン 0 を使っていて、リビジョン 3 の CD を 買ってきたとします。この CD を元に APT を使って、システムをアップグレード できます。その為には、apt-cdrom ('sources.list ファイルに CD-ROM を追加する' on page 6 を参照) を使って CD を /etc/apt/sources.list に追加し、それから apt-get dist-upgrade を実行します。

APT は常に最新バージョンのパッケージを探すことに注意してください。 したがって、etc/apt/sources.list に CD よりも新しいバージョンの パッケージを含むアーカイブが記述されていれば、APT はそこからパッケージを ダウンロードします。

'パッケージのアップグレード' on page 12 の節で示した例では、いくつかのパッケージが kept back されているのがわかります。この問題は、 dist-upgrade によって解決できます。

```
[root]@[/] # apt-get -u dist-upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
Calculating Upgrade... Done
The following NEW packages will be installed:
  cpp-2.95 cron exim gcc-2.95 libident libopenldap-runtime libopenldap1
  libpcre2 logrotate mailx
The following packages have been kept back
  lilo
The following packages will be upgraded
  adduser ae apt autoconf cpp debhelper dpkg-dev esound esound-common ftp gcc
  indent ipchains isapnptools libaudiofile-dev libaudiofile0 libesd0
  libesd0-dev libgtk1.2 libgtk1.2-dev liblockfile1 libnewt0 liborbit-dev
  liborbit0 libstdc++2.10-glibc2.2 libtiff3g libtiff3g-dev modconf orbit
  procps psmisc
31 packages upgraded, 10 newly installed, 0 to remove and 1 not upgraded.
Need to get 0B/7098kB of archives. After unpacking 3118kB will be used.
Do you want to continue? [Y/n]
```

パッケージがアップグレードされると共に、新たなパッケージが (パッケージ間の 新たな依存関係に応じて) インストールされていることに注意してください。 さらに、lilo が依然として kept back されていることにも注意して ください。これはおそらく、新たな依存関係を超える深刻な問題があるからで しょう。次の例によって原因がわかります:

```
[root]@[/] # apt-get -u install lilo
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
```

```
  cron debconf exim libident libopenldap-runtime libopenldap1 libpcre2
  logrotate mailx
The following packages will be REMOVED:
  debconf-tiny
The following NEW packages will be installed:
  cron debconf exim libident libopenldap-runtime libopenldap1 libpcre2
  logrotate mailx
The following packages will be upgraded
  lilo
1 packages upgraded, 9 newly installed, 1 to remove and 31 not upgraded.
Need to get 225kB/1179kB of archives. After unpacking 2659kB will be used.
Do you want to continue? [Y/n]
```

上記に示されているように、lilo には debconf-tiny パッ ケージと新たに衝突しています。この事は、debconf-tiny を除去しなければ lilo をインストール (あるいはアップグレード) できないことを意味しています。

## 3.6 APT を dselect と一緒に使う

dselect は、インストールする Debian パッケージを選択する際の 手助けとなるプログラムです。やや複雑でうんざりさせられるプログラムだと 考えられていますが、練習すればそのコンソールベースの ncurses インター フェイスを操作するコツを掴むことができます。

dselect の機能の一つに、ある Debian パッケージをインストールするにあたって 他のパッケージを “recommending” したり “suggesting” するという能力を 活用できるということがあります。このプログラムを使うには、root になって 'dselect' と実行してください。それからアクセス方法として、'apt' を選択してください。この手順は実のところ不要なのですが、CD-ROM を 使わずにインターネットからパッケージをダウンロードしたいのであれば、 dselect を使うにあたって最善の方法です。

dselect の使用方法についてさらに理解したければ、Debian のホームページ http://www.debian.org/doc/ddp にある dselect のドキュメントを読んでください。

deselect を使って選択が終われば、次のようにしてください:

```
apt-get -u dselect-upgrade
```

すると以下の例のようになります:

```
[root]@[/] # apt-get -u dselect-upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be REMOVED:
  lbxproxy
The following NEW packages will be installed:
```

```
  bonobo console-tools-libs cpp-3.0 enscript expat fingerd gcc-3.0
  gcc-3.0-base icepref klogd libdigest-md5-perl libfnlib0 libft-perl
  libgc5-dev libgcc300 libhtml-clean-perl libltdl0-dev libsasl-modules
  libstdc++3.0 metamail nethack proftpd-doc psfontmgr python-newt talk tidy
  util-linux-locales vacation xbill xplanet-images
The following packages will be upgraded
  debian-policy
1 packages upgraded, 30 newly installed, 1 to remove and 0  not upgraded.
Need to get 7140kB of archives. After unpacking 16.3MB will be used.
Do you want to continue? [Y/n]
```

同じシステムで apt-get dist-upgrade を実行した場合と比べてみてください。

```
[root]@[/] # apt-get -u dist-upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
Calculating Upgrade... Done
The following packages will be upgraded
  debian-policy
1 packages upgraded, 0 newly installed, 0 to remove and 0  not upgraded.
Need to get 421kB of archives. After unpacking 25.6kB will be freed.
Do you want to continue? [Y/n]
```

他のパッケージが “suggested” したり “recommended” している多くのパッケージが インストールされようとしていることに注意してください。他のパッケージは、 dselect の一覧表示を通じて選択したパッケージ毎にインストールされたり 削除 (lbxproxy の場合など) されたりしようとしています。APT と組み合わせて 使用すると、dselect は強力なツールとなるのです。

## 3.7 インストール済パッケージを特定バージョンのまま保持する方法

パッケージに何らかの修正を行なった後、そのプログラムの新バージョンに 反映させる時間がなかったり、反映させるのを望まないことがあるかもしれません。 あるいは例えば、Debian ディストリビューションを 3.0 にアップグレードしたものの、特定パッケージのバージョンだけは Debian 2.2 のものを使用したいことがあるかもしれません。インストールしたパッケージがアップグレードされないよう、“ピン” で止めることができます。

この目的で使うリソースは単純です。/etc/apt/preferences ファイルを 編集するだけです。

フォーマットは簡単です:

```
Package: <package>
```

```
Pin: <pin definition>
Pin-Priority: <pin's priority>
```

例えば sylpheed パッケージのバージョン 0.4.99 に “reply-to-list” を使うように修正して、それを保持するためには、次のようにします。

```
Package: sylpheed
Pin: version 0.4.99*
```

* (アスタリスク) を使っていることに注意してください。これは “ワイルドカード” であり、0.4.99 で始まる全バージョンを “ピン” で固定したいということです。なぜこのようなことをするかというと、Debian はパッケージのバージョンに “Debian リビジョン” を付加するからであり、これらのリビジョンがインストールされるのを避けたいからです。さもないと、バージョン 0.4.99-1 や 0.4.99-10 が利用可能になるとすぐにインストールされてしまいます。修正を加えたパッケージを使い続けたいのなら、このような動作を望まないでしょう。

Pin-Priority フィールドはオプションです。特に指定がなければ、デフォルト値は 989 となります。

Pin-Priority がどのように働くかを見てみましょう。値が 0 未満の場合、そのパッケージは絶対にインストールされません。値が 0 以上 100 未満だと、パッケージはインストールされておらず利用可能なバージョンもありません。これらのパッケージはバージョン選択プロセスにも現れません。100 の値は、インストール済のパッケージに当てはめられます - 他のバージョンによって置き換え可能であり、置き換える側の値は 101 以上でなければなりません。

101 以上の値は、そのパッケージがインストールされるべきであることを示しています。よくあるのは、あるパッケージのインストール済バージョンがより進んだバージョンにのみアップグレードされる場合です。100 以上 1000 以下のあらゆる値は、この典型的な動作をすることを示しています。このような値を持つパッケージは、利用可能であっても古いバージョンにはダウングレードされません。例えば sylpheed 0.5.3 をインストーしてから、sylpheed 0.4.99 の priority 値を 999 に設定しても、0.4.99 はその設定によってインストールされません。あるパッケージをダウングレード可能にするには、値を 1001 以上に設定しなければなりません。

このダウングレードという概念により、強力な応用が可能となります。たとえば Debian の stable 版を testing 版にアップグレードしたものの、それを取りやめたくなったと想像してください - たぶん、testing はあなたに合うほどにはテストされていなかったのでしょう。そこでパッケージ名にワイルドカードを用いてピン設定をすることにより、stable に戻ることができます。例えば:

```
Package: *
Pin: release a=stable
Pin-Priority: 1001
```

上記のように設定したあとで、apt-get -u dist-upgrade とすれば、 stable 版にシステムをダウングレードすることができます。

このピンの設定は、パッケージの version, release, origin に対して行なうことができます。

version に対するピン設定は、上記の例のように、ワイルドカードを 使って複数バージョンを一度に指定することも、特定のバージョンだけを 指定することもできます。

release オプションは、CD 内にある APT レポジトリのリリースファイルに依存します。このファイルを提供していないパッケージレポジトリを 使っているなら、このオプションは全く役に立ちません。/var/lib/apt/lists/ 内にあるレポジトリファイルの中身を見ることができます。リリースの パラメータとしては: a (archive), c (components), v (version), o (origin) そして l (label) があります。

以下は例です:

```
Package: *
Pin: release v=2.2*,a=stable,c=main,o=Debian,l=Debian
Pin-Priority: 1001
```

この例では、Debian バージョン 2.2* (2.2r2 や 2.2r3 が対象となります – これは “ポイントリリース” のことで、セキュリティフィックスや他の極めて重要な アップデートを含む際によく使われます) で、stable レポジトリ、 (contrib や non-free に対して) main セクション、 origin と label は Debian をそれぞれ選択しています。origin (o=) はリリースファイルの製作者を、label (l=) はディストリビューション名を表わします。 例えば Debian は Debian 自体であり、Progeny は Progeny Debian を意味します。 リリースファイルの例を以下に示します:

```
$ cat /var/lib/apt/lists/ftp.debian.org.br_debian_dists_potato_main_binary-i3
Archive: stable
Version: 2.2r3
Component: main
Origin: Debian
Label: Debian
Architecture: i386
```

# 章 4

# とても便利な補助プログラム

## 4.1 ローカルでコンパイルされたパッケージのインストール方法: equivs

Debian パッケージ化されていない、ソースコードのみ利用可能な特定バージョンの プログラムのを使いたい場合があるでしょう。しかし、これをやるとパッケージング システムがトラブルを起こすことがあります。電子メールサーバの新バージョンを コンパイルしたいと想像してみてください。全てうまくいきましたが、Debian の 多くのパッケージは MTA に依存しています。自分でコンパイルしたものをインストールしても、パッケージシステムはその事に気づいてくれません。

ここで `equivs` が登場する番です。使うには、同名の パッケージをインストールしてください。同パッケージが行なうのは、依存関係を 満たせる空のパッケージを生成することで、これによりパッケージング システムに 依存関係が満たされていると信じ込ませます。

さっそく始める前に、Debian パッケージが存在するプログラムを、異なる オプション付でコンパイルするもっと安全な方法があり、自分がしていることを 理解できないのなら、依存関係を置き換えるのに equivs を使うべきではないということは、覚えておいた方がよいでしょう。より詳しい情報は、‘ソースパッケージでの作業’ on page 29 のセクションを参照してください。

MTA を例にして話を続けましょう。あなたは新たにコンパイルした postfix を インストールしたばかりで、今度は `mutt` をインストール しようとしています。突然、`mutt` が他の MTA をインストールしたがっているのを発見しました。ですが、あなたには自分用の MTA が すでにあります。

(たとえば `/tmp` などの) ディレクトリへ移って、次のように 実行してください:

```
# equivs-control name
```

*name* の箇所は、生成したいコントロールファイル名で置き換えて ください。以下のようにファイルが生成されるでしょう:

```
Section: misc
Priority: optional
Standards-Version: 3.0.1

Package: <enter package name; defaults to equivs-dummy>
Version: <enter version here; defaults to 1.0>
Maintainer: <your name and email address; defaults to username>
Pre-Depends: <packages>
Depends: <packages>
Recommends: <packages>
Suggests: <package>
Provides: <(virtual)package>
Architecture: all
Copyright: <copyright file; defaults to GPL2>
Changelog: <changelog file; defaults to a generic changelog>
Readme: <README.Debian file; defaults to a generic one>
Extra-Files: <additional files for the doc directory, comma separated>
Description: <short description; defaults to some wise words>
 long description and info
 .
 second paragraph
```

これを望む通りに修正する必要があるだけです。フィールドのフォーマットと、その説明を眺めてください。各々について、ここで説明する必要はないでしょう。 さあ、必要なことをやってしまいましょう。

```
Section: misc
Priority: optional
Standards-Version: 3.0.1

Package: mta-local
Provides: mail-transport-agent
```

そう、これで終わりです。mutt は mailt-transport-agent に依存しており、これはあらゆる MTA によって 提供される仮想パッケージです。単純にパッケージ名を mailt-transport-agent とすることもできますが、著者は Provides を使って 仮想パッケージの概要を示す方が好みです。

あとはパッケージを構築するだけです:

```
# equivs-build name
dh_testdir
touch build-stamp
dh_testdir
dh_testroot
```

```
dh_clean -k
# Add here commands to install the package into debian/tmp.
touch install-stamp
dh_testdir
dh_testroot
dh_installdocs
dh_installchangelogs
dh_compress
dh_fixperms
dh_installdeb
dh_gencontrol
dh_md5sums
dh_builddeb
dpkg-deb: building package `name' in `../name_1.0_all.deb'.

The package has been created.
Attention, the package has been created in the current directory,
```

そして、出来あがった .deb をインストールしてください。

お判りのように、equivs には様々な使用法があります。たとえば、あなたがいつもインストールするプログラムに依存する my-favorites パッケージを生成することさえ可能です。自由に想像力の翼を拡げてかまいませんが、用心は してください。

/usr/share/doc/equivs/examples 内にコントロールファイルの見本が ありますので忘れないように。それらをチェックしてください。

## 4.2 使用されていない locale ファイルを削除: localepurge

多くの Debian ユーザは、たった 1 種類の locale しか使っていません。たとえば ブラジルの Debian ユーザなら、普段は常に pt_BR という locale を 使うでしょう。

localepurge は、これらのユーザにとって非常に役に立つ ツールです。本当に使う locale のみを残すことにより、大量のディスク容量を 節約できます。さあ、apt-get install localepurge してください。

設定はとても簡単で、debconf の質問に沿って順序通りに設定していきます。ですが 最初の質問には慎重に答えてください。間違って答えてしまうと、あなたが使いたい ものも含めて全ての locale ファイルが削除されるかもしれません。ファイルを復活 させる手段は、それらを提供するパッケージを再インストールするしかありません。

# 章 5

# パッケージ情報の入手

APT システムには、パッケージが属しているセクションや重要度、その 内容説明などを見るためのフロンとエンドプログラムと同様に、インストール 可能かあるいはインストール済のパッケージ一覧をきわめて容易に入手する ためのフロントエンドプログラムがいくつかあります。

ですが... ここでの我々の目的は APT 本体の使い方を学ぶことです。よって、 インストールしたいパッケージ名を見つけるにはどうすれば良いのでしょうか？

この目的のためには、いくつかのリソースがあります。apt-cache から 始めましょう。このプログラムは、APT システムが自分のデータベースを 整備するために使われます。ここでは、より実用的な応用について概観する ことにしましょう。

## 5.1 パッケージ名の発見

例えば、旧き良き Atari 2600 の日々について回想に耽りたいとしましょう。 APT を使って Atari エミュレータをインストールし、いくつかのゲームを ダウンロードできます。次のようにしてください:

```
[root]@[/] # apt-cache search atari
atari-fdisk-cross - Partition editor for Atari (running on non-Atari)
circuslinux - The clowns are trying to pop balloons to score points!
madbomber - A Kaboom! clone
tcs - Character set translator.
atari800 - Atari emulator for svgalib/X/curses
stella - Atari 2600 Emulator for X windows
xmess-x - X binaries for Multi-Emulator Super System
```

探しているものに関連するいくつかのパッケージと、その簡単な説明が表示 されています。特定のパッケージに関するさらに詳しい情報を得るには、 次のようにしてください:

```
[root]@[/] # apt-cache show stella
Package: stella
Priority: extra
Section: non-free/otherosfs
Installed-Size: 830
Maintainer: Tom Lear <tom@trap.mtview.ca.us>
Architecture: i386
Version: 1.1-2
Depends: libc6 (>= 2.1), libstdc++2.10, xlib6g (>= 3.3.5-1)
Filename: dists/potato/non-free/binary-i386/otherosfs/stella_1.1-2.deb
Size: 483430
MD5sum: 11b3e86a41a60fa1c4b334dd96c1d4b5
Description: Atari 2600 Emulator for X windows
 Stella is a portable emulator of the old Atari 2600 video-game console
 written in C++.  You can play most Atari 2600 games with it.  The latest
 news, code and binaries for Stella can be found at:
 http://www4.ncsu.edu/~bwmott/2600
```

上記の出力から、インストールしたい (あるいはしたくない) パッケージの 詳細が得ることができ、そのパッケージの完全な説明文もあります。 そのパッケージがインストール済で、しかも新バージョンが利用可能なら、 両バージョンの情報が表示されます。例えば:

```
[root]@[/] # apt-cache show lilo
Package: lilo
Priority: important
Section: base
Installed-Size: 271
Maintainer: Russell Coker <russell@coker.com.au>
Architecture: i386
Version: 1:21.7-3
Depends: libc6 (>= 2.2.1-2), debconf (>=0.2.26), logrotate
Suggests: lilo-doc
Conflicts: manpages (<<1.29-3)
Filename: pool/main/l/lilo/lilo_21.7-3_i386.deb
Size: 143052
MD5sum: 63fe29b5317fe34ed8ec3ae955f8270e
Description: LInux LOader - The Classic OS loader can load Linux and others
 This Package contains lilo (the installer) and boot-record-images to
 install Linux, OS/2, DOS and generic Boot Sectors of other OSes.
 .
 You can use Lilo to manage your Master Boot Record (with a simple text scree
 or call Lilo from other Boot-Loaders to jump-start the Linux kernel.

Package: lilo
Status: install ok installed
```

```
Priority: important
Section: base
Installed-Size: 190
Maintainer: Vincent Renardias <vincent@debian.org>
Version: 1:21.4.3-2
Depends: libc6 (>= 2.1.2)
Recommends: mbr
Suggests: lilo-doc
Description: LInux LOader - The Classic OS loader can load Linux and others
 This Package contains lilo (the installer) and boot-record-images to
 install Linux, OS/2, DOS and generic Boot Sectors of other OSes.
 .
 You can use Lilo to manage your Master Boot Record (with a simple text scree
 or call Lilo from other Boot-Loaders to jump-start the Linux kernel.
```

最初のものが利用可能なパッケージのもので、二番目がインストール済の ものであることに注意してください。パッケージに関する一般的な情報を得るには、次のようにしてください:

```
[root]@[/] # apt-cache showpkg penguin-command
Package: penguin-command
Versions:
1.4.5-1(/var/lib/apt/lists/download.sourceforge.net_debian_dists_unstable_mai

Reverse Depends:
Dependencies:
1.4.5-1 - libc6 (2 2.2.1-2) libpng2 (0 (null)) libsdl-mixer1.1 (2 1.1.0) libs
Provides:
1.4.5-1 -
Reverse Provides:
```

さらにどのパッケージに依存しているのかを見るには:

```
[root]@[/] # apt-cache depends penguin-command
penguin-command
  Depends: libc6
  Depends: libpng2
  Depends: libsdl-mixer1.1
  Depends: libsdl1.1
  Depends: zlib1g
```

まとめると、我々は欲しいパッケージ名を探すのに使える多様な武器を持っていることになります。

## 5.2 dpkg を使ってパッケージ名を探す

パッケージ名を特定する方法の一つに、パッケージ中の重要なファイル名を 知っている場合があります。例えば、特定の "`.h`" ファイルがないと コンパイルできない場合、次のようにしてください。

```
[root]@[/] # dpkg -S stdio.h
libc6-dev: /usr/include/stdio.h
libc6-dev: /usr/include/bits/stdio.h
perl: /usr/lib/perl/5.6.0/CORE/nostdio.h
```

あるいは:

```
[root]@[/] # dpkg -S /usr/include/stdio.h
libc6-dev: /usr/include/stdio.h
```

システムにインストール済のパッケージ名を探す時も、便利です。例えば、 ハードディスクを掃除したいなら次のようにしてください。

```
[root]@[/] # dpkg -l | grep mozilla
ii  mozilla-browse 0.9.6-7        Mozilla Web Browser
```

このコマンドの問題点は、パッケージ名が "途切れて" しまうことです。 上記の例だと、パッケージの本当の名前は `mozilla-browser` です。 これに対処するには、次のように環境変数 `COLUMNS` を使うことができます:

```
[kov]@[couve] $ COLUMNS=132 dpkg -l | grep mozilla
ii  mozilla-browser             0.9.6-7                     Mozilla Web Brows
```

あるいは次のように、description かその一部を使うこともできます:

```
[root]@[/] # apt-cache search "Mozilla Web Browser"
mozilla-browser - Mozilla Web Browser
```

## 5.3 "必要に応じて" パッケージをインストールする方法

プログラムをコンパイルしていて、まったくの突然、ドカーン！ エラーの 原因は、ある `.h` ファイルが無かったからでした。このような 展開は、`auto-apt` プログラムを使うことで避けられます。 あるパッケージが必要になった時点でインストールするかどうかを尋ね、 関連するプロセスを停止して、インストールが完了したら再開します。

基本的には、次のように実行してください:

```
auto-apt run command
```

'command' の個所には、実行するのに必要なファイルが無いコマンドを 指定してください。例えば:

```
auto-apt run ./configure
```

必要なパッケージをインストールするかどうかを尋ねてきて、自動的に apt-get を呼び出します。X の実行中であれば、デフォルトのテキスト インターフェイスに代わってグラフィカル インターフェイスが使われ ます。

auto-apt はデータベースを持っており、効果的に使うためにはそれを更新する 必要があります。そのためには、`auto-apt update`, `auto-apt updatedb`, `auto-apt update-local` などのコマンドを使います。

## 5.4 パッケージの変更点を常に知らせる方法

どのパッケージも、個別のドキュメント用ディレクトリ (`/usr/share/doc/packagename`) に、`changelog.Debian.gz` というファイルをインストールします。このファイルは、前回のバージョンからの変更点一覧を 含んでいます。こういったファイルは、'zless' などの助けを借りて読むこと ができますが、全システムをアップグレードした後などは、アップグレードされた 各パッケージの changelog を探すのに骨が折れます。

`apt-listchanges` というツールを使って、この作業を自動化できます。 まずは `apt-listchanges` パッケージをインストールする必要が あります。インストール中に、Debconf が設定を行ないます。質問に対して、 あなたの希望するように答えてください。

"Should apt-listchanges be automatically run by apt?" というオプションは、アップグレード途中で apt によってインストールされようとしている各パッケージの 変更点一覧を表示し、作業を続ける前にその内容を確認できるので、役に立ちます。 "Should apt-listchanges prompt for confirmation after displaying changes?" というオプションは、変更点一覧を読んだ後で、インストールを続けるかどうかを問い 問い合わせてくれるので役に立ちます。継続を望まないと答えれば、apt-listchanges は エラーを返し、apt はインストールを取り止めます。

apt-listchanges がインストールされればすぐに、その後ダウンロードされた (あるい は CD やマウントされたディスクから入手した) パッケージはインストールされる前に 変更点一覧を表示するようになります。

# 章 6

# ソースパッケージでの作業

## 6.1 ソースパッケージのダウンロード

フリーソフトウェアの世界では、ソースコードで勉強したり、バグの多い ソースを修正するのはよくあることです。このためには、プログラムのソース をダウンロードしなければならないでしょう。APT システムはディストリビューション 中の多くのプログラムのソースコードと、そのプログラムを .deb 化するために 必要な全ファイルを入手するための、簡単な方法を提供します。

Debian ソースのよくある使い方としては、プログラムの新しいバージョンを 適合させる場合などです。例えば、stable ディストリビューションに unstable 中のプログラムを使う場合など。あるパッケージを stable 用に コンパイルするには、そのディストリビューションで利用可能なように 依存関係を調整して .deb 化します。

このためには、`/etc/apt/sources.list` 中の `deb-src` の エントリが unstable 向けとなっていなければなりません。さらにエントリが 有効 (アンコメント) になっている必要があります。‘/etc/apt/sources.list ファイル’ on page 3 を 参照してください。

ソースパッケージをダウンロードするには、以下のコマンドを使ってください:

```
$ apt-get source packagename
```

これにより 3 つのファイルがダウンロードされます: `.orig.tar.gz`, `.dsc`, `.diff.gz` です。Debian 専用に作られたパッケージの 場合、3 番目のファイルはダウンロードされず、最初のものもファイル名中に “`org`” と付きません。

`.dsc` ファイルは、dpkg-source がソースパッケージを *packagename-version* のディレクトリに展開するために 使われます。ダウンロードされた各ソースパッケージには、.deb パッケージを 作るために必要なファイルが含まれている `debian/` ディレクトリがあります。

ダウンロードしたパッケージを自動的にビルドするには、次の例のよう に `-b` をコマンドラインに付加します:

```
$ apt-get -b source packagename
```

ダウンロード時に .deb を作らないのなら、次のようにすることで後から 作ることもできます。

```
$ dpkg-buildpackage -rfakeroot -uc -b
```

上記のコマンドは、ダウンロードされたパッケージがあるディレクトリ内で 実行してください。

apt-get の source 指定と他の指定とでは違いがあります。 source 指示は、一般ユーザにも実行可能で、特別な root 権限は 必要ありません。ファイルは、apt-get source package が実行された ディレクトリにダウンロードされます。

## 6.2 ソースパッケージのコンパイルに必要なパッケージ

通常、ソースパッケージをコンパイルするには、特定のヘッダや共有 ライブラリが存在することが必要です。全 .deb パッケージには、その コントロールファイル中に 'Build-Depends' というフィールドがあります。 これは、そのパッケージをソースからビルドする際に、必要となる追加 パッケージが指定しています。

APT はこれらのパッケージ ダウンロードするのも容易にします。 apt-get build-dep package を実行するだけです。'package' の箇所 は、ビルドしたいパッケージの名前です。例えば:

```
[root]@[/] # apt-get build-dep gmc
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  comerr-dev e2fslibs-dev gdk-imlib-dev imlib-progs libgnome-dev libgnorba-de
  libgpmg1-dev
0 packages upgraded, 7 newly installed, 0 to remove and 1  not upgraded.
Need to get 1069kB of archives. After unpacking 3514kB will be used.
Do you want to continue? [Y/n]
```

インストールされるのは、gmc を正確にビルドするために 必要なパッケージです。このコマンドは、ビルドされるプログラムのソース パッケージを探さないことに注意してください。したがって、それを入手する のに別途 apt-get source を実行する必要があります。

# 章 7

# エラーへの対処方法

## 7.1 よくあるエラー

エラーは常に起き、多くはユーザの不注意が原因です。以下は、とても頻繁に 報告されるエラーと、その対処方法です。

apt-get install package を実行した際に、以下のような メッセージを受け取ったら...

```
Reading Package Lists... Done
Building Dependency Tree... Done
W: Couldn't stat source package list 'http://people.debian.org unstable/ Pack
W: You may want to run apt-get update to correct these missing files
E: Couldn't find package penguineyes
```

最後に /etc/apt/sources.list ファイルを修正した後で、 apt-get update を実行するのを忘れているのでしょう。

以下のエラーの場合は:

```
E: Could not open lock file /var/lib/dpkg/lock - open (13 Permission denied)
E: Unable to lock the administration directory (/var/lib/dpkg/), are you root
```

source 以外の apt-get 指示を行なう場合は、 root 許可がなければなりません。つまり、一般ユーザとして実行して しまったのです。

上記に似たエラーが出るのは、apt-get を同時に 2 つ実行 しようとした場合や、dpkg のプロセスが生きている間に、 apt-get を実行しようとした場合があります。 同時に並行して実行できるのは、source 指示だけです。

インストール時、処理途中で止まってしまい、そのパッケージを インストール することも削除もできなくなった場合、次の 2 つのコマンドを実行してみて ください。

```
# apt-get -f install
# dpkg --configure -a
```

その後で、もう一度試してください。上記の例中の 2 番目のコマンドは、繰返し実行する必要があるかもしれません。これは 'unstable' を使う 冒険家にとって、重要な教えです。

## 7.2 ヘルプはどこにありますか?

疑いを抱いたなら、Debian パッケージング システム用に利用可能な 膨大なドキュメントを調べてみてください。/usr/share/doc/apt の ような /usr/share/doc 内にあるドキュメントと同様、 --help オプションや man ページも大きな助けとなるでしょう。

以上のドキュメントでも恐怖を拭い去ることができなければ、Debian の メーリング リスト中で回答を探してみてください。特定のリストに関する 情報は Debian のウェブサイトで見つかります: http://www.debian.org.

これらのリストやリソースは、Debian ユーザだけが使用すべきであることに 注意してください。他のシステムのユーザには、各ディストリビューションの コミュニティ リソースが有するより良いサポートがあるでしょう。

# 章 8

# APT に対応しているディストリビューションは?

以下に、APT を使用しているいくつかのディストリビューション名を示します。

Debian GNU/Linux (http://www.debian.org) - APT が開発されたのは、このディストリビューションのためにです。

Conectiva (http://www.conectiva.com.br) - APT で rpm を使えるように移植した、最初のディストリビューションです。

Mandrake (http://www.mandrake.com)

PLD (http://pld.org.pl (http://www.pld.org.pl))

Vine (http://www.vinelinux.org)

APT4RPM (http://apt4rpm.sf.net)

# 章 9

# クレジット

Debian-BR プロジェクトと、Debian 本体の偉大なる友人たちに深く感謝します。 絶え間のない手助けと、世界を救うという私の目的を手伝ってくれるばかりで なく、人類の利益のために作業を続ける力を私に与えてくれました :)。

CIPSGA に対して我々のプロジェクトへの多大なる助力に、また偉大なアイデア の弾みとなったあらゆるフリープロジェクトにも、感謝したいと思います。

以下の方には特に謝意を表します。

Yooseong Yang <yooseong@debian.org> - 本マニュアルを韓国語に翻訳して くれました。

Michael Bramer <grisu@debian.org> - 特定バージョンの保持の節を含める ように提案してくれました。

Bryan <Stillwell bryan@bokeoa.com> - 各種パッチや修正などを私に 送ってくれました。

Pawel Tecza <pawel.tecza@poland.com> - 各種修正を送ってくれ、また ポーランド語に翻訳してくれました。

Pablo Lorenzzoni <spectra@debian.org> - netselect に関する節を 執筆してくれました。

Steve Langasek <vorlon@netexpress.net> - 本マニュアルを英語に 翻訳してくれました。

Arnaldo Carvalho de Melo <acme@conectiva.com.br> - 現在 APT を 含んでいるディストリビューション - Mandrake, PLD, Vine - をリストに 追加してくれました。

Erik Rossen <rossen@freesurf.ch> - dpkg -l での COLUMNS の使用という tips を提供してくれました。

Ross Boylan <RossBoylan@stanfordalumni.org> - -o Debug::pkgProblemResolver=yes の使用という tips を提供してくれました。

# 章 10

# このチュートリアルの新バージョン

本マニュアルは、Debian を使うあらゆる人の助けとなることを願って、 Debian-BR (http://debian-br.cipsga.org.br) プロジェクトに よって作成されました。

最新バージョンは、プロジェクトのホームページから入手可能です: http://debian-br.cipsga.org.br/suporte/documentacao.html.

コメントや批判は、私 <kov@debian.org> 宛に直接メール してください。