

APT HOWTO

Gustavo Noronha Silva <kov@debian.org>
Traduction de Julien Louis <leonptitlouis@ifrance.com>

1.8.10.4 - mars 2005

Résumé

Ce document essaie de fournir à l'utilisateur une bonne compréhension du fonctionnement de l'utilitaire Debian de gestion de paquets, `apt`. Son but est de rendre la vie plus facile aux nouveaux utilisateurs Debian et aider ceux qui veulent approfondir leur compréhension de l'administration de ce système. Il a été créé pour le projet Debian afin d'aider à améliorer les ressources disponibles pour l'utilisateur de cette distribution.

Copyright

Copyright © 2001, 2002, 2003, 2004 Gustavo Noronha Silva

Ce manuel est libre, vous pouvez le redistribuer et/ou le modifier selon les termes de la Licence Publique Générale GNU publiée par la Free Software Foundation (version 2 ou bien toute autre version ultérieure choisie par vous).

Ce manuel est distribué car potentiellement utile, mais SANS AUCUNE GARANTIE, ni explicite ni implicite, y compris les garanties de commercialisation ou d'adaptation dans un but spécifique. Reportez-vous à la Licence Publique Générale GNU pour plus de détails.

Une copie de la Licence Publique Générale est disponible dans la distribution Debian GNU/Linux dans `/usr/share/common-licenses/GPL` ou sur Internet. Vous pouvez aussi en obtenir une copie en écrivant à la Free Software Foundation, INC., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

Table des matières

1	Introduction	1
2	Configuration de base	3
2.1	Le fichier <code>/etc/apt/sources.list</code>	3
2.2	Comment utiliser <code>apt</code> localement ?	4
2.3	Comment décider quel est le meilleur miroir pour le fichier <code>sources.list</code> : <code>netselect</code> , <code>netselect-apt</code> ?	5
2.4	Ajouter un cd-rom dans le fichier <code>sources.list</code>	6
3	Gestion des paquets	9
3.1	Mise à jour de la liste des paquets disponibles.	9
3.2	Installation de paquets	9
3.3	Suppression de paquets	11
3.4	Mise à niveau des paquets	12
3.5	Mettre à niveau vers une nouvelle distribution	13
3.6	Supprimer des paquets non utilisés : <code>apt-get clean</code> et <code>autoclean</code>	15
3.7	Utiliser <code>apt</code> avec <code>dselect</code>	16
3.8	Comment garder un système mixte ?	17
3.9	Comment mettre à niveau un paquet d'une distribution Debian spécifique ?	18
3.10	Comment garder des versions spécifiques de paquets installés (complexe)	18
4	Aides très utiles	21
4.1	Comment installer un paquet localement : <code>equivs</code>	21
4.2	Suppression des fichiers de locale inutilisés : <code>localepurge</code>	23
4.3	Comment savoir quels paquets peuvent être mis à niveau	23

5	Avoir des informations sur les paquets	25
5.1	Recherche de noms de paquets	25
5.2	Utilisation de dpkg pour trouver le nom des paquets	28
5.3	Comment installer des paquets « à la demande » ?	28
5.4	Comment savoir à quel paquet appartient un fichier ?	29
5.5	Comment rester informé sur les changements dans les paquets ?	30
6	Travailler avec des paquets sources	31
6.1	Télécharger un paquet source	31
6.2	Paquets nécessaires pour la compilation d'un paquet source	32
7	Comment traiter les erreurs ?	35
7.1	Erreurs courantes	35
7.2	Où puis-je trouver de l'aide ?	36
8	Quelles distributions supportent apt ?	37
9	Crédits	39
10	Nouvelles versions de ce manuel	41

Chapitre 1

Introduction

Au commencement, il y avait le `.tar.gz`. Les utilisateurs devaient compiler chaque programme qu'ils voulaient utiliser sur leur système GNU/Linux. Quand Debian fut créée, il a été estimé nécessaire que le système s'occupe aussi de la gestion des paquets installés sur la machine. Le nom `dpkg` fut donné à ce système. Ainsi, ce fut la première fois qu'un « système de paquets » était inclus dans GNU/Linux, longtemps avant que Red Hat ne décide de créer son système de « rpm ».

Un nouveau dilemme est rapidement survenu dans l'esprit des développeurs GNU/Linux. Ils avaient besoin d'une solution rapide, pratique et efficace pour installer les paquets qui générerait automatiquement les dépendances et qui prendrait en compte les fichiers de configuration des paquets lors de mises à niveau. Ici encore, Debian a ouvert la voie et a donné naissance à `apt`, *Advanced Packaging Tool*, qui depuis a été porté par Conectiva pour l'utiliser avec les rpm et a été adopté par quelques autres distributions.

Ce manuel ne cherche pas à documenter `apt-rpm`, le portage d'`apt` par Conectiva, mais des « rustines » en ce sens seraient les bienvenues.

Ce manuel est basé sur la prochaine version de Debian Sarge

Chapitre 2

Configuration de base

2.1 Le fichier `/etc/apt/sources.list`

Pour réaliser ses opérations, `apt` utilise un fichier qui liste les « sources » d'où peuvent provenir les paquets. Ce fichier est `/etc/apt/sources.list`.

Les entrées dans ce fichier suivent ce format :

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

Bien sûr, les entrées ci-dessus sont fictives et ne doivent pas être utilisées. Le premier mot de chaque ligne, `deb` ou `deb-src`, indique le type de l'archive : `deb` pour les paquets binaires, ce sont les paquets pré-compilés que l'on utilise habituellement, ou `deb-src` pour les paquets sources, qui sont les sources originales du programme, plus les fichiers de contrôle Debian (`.dsc`) et le `diff.gz` contenant les changements nécessaires pour « debianiser » le programme.

Nous trouvons généralement les lignes suivantes dans le `sources.list` Debian par défaut :

```
# See sources.list(5) for more information, especially
# 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-free
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
```

Ce sont les lignes nécessaires à une installation Debian de base. La première ligne `deb` pointe vers l'archive officielle, la seconde vers l'archive non-US et la troisième vers l'archive des mises à jour de sécurité Debian.

Les deux dernières lignes sont commentées (avec un « # » devant), `apt-get` les ignorera. Ce sont les lignes `deb-src`, elles pointent vers les paquets source Debian. Si vous téléchargez souvent les sources de programmes pour tester ou recompiler, décommentez-les.

Le fichier `/etc/apt/sources.list` peut contenir plusieurs types de lignes. `apt` sait traiter les archives de type `http`, `ftp`, `file` (les fichiers locaux, c'est-à-dire, un répertoire contenant une image ISO9660 montée) et `ssh`.

2.2 Comment utiliser apt localement ?

Parfois vous avez beaucoup de paquets `.deb` et vous aimeriez utiliser `apt` pour les installer afin que les dépendances soient automatiquement résolues.

Pour ce faire, créez un répertoire et ajoutez-y vos fichiers `.deb`. Par exemple :

```
# mkdir /root/debs
```

Vous pouvez modifier directement l'ensemble des définitions dans le fichier de contrôle du paquet en utilisant un fichier `override` dans votre référentiel. À l'intérieur de ce fichier, vous pouvez vouloir définir certaines options pour modifier celles incluses dans le paquet original. Un tel fichier ressemble à ce qui suit :

```
paquet priorité section
```

`paquet` est le nom du paquet, `priorité` est faible, moyenne ou haute, et `section` est la section à laquelle il appartient. Le nom du fichier importe peu, vous devrez le passer après en argument à `dpkg-scanpackages`. Si vous ne souhaitez pas écrire de fichier `override`, utilisez simplement `/dev/null` lorsque vous appelez `dpkg-scanpackages`.

Toujours dans le répertoire `/root`, faites :

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

Dans la ligne ci-dessus *fichier* est le fichier « `override` », la commande produit un fichier `Packages.gz` qui contient diverses informations sur les paquets utilisées par `apt`. Pour utiliser les paquets, ajoutez enfin :

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

Ensuite, utilisez `apt` normalement. Vous pourriez aussi fabriquer un dépôt de sources. Pour ce faire utilisez la même procédure, mais rappelez-vous que vous avez besoin des fichiers `.orig.tar.gz`, `.dsc` et `.diff.gz` dans le répertoire. De plus, vous devez utiliser `Sources.gz` au lieu de `Packages.gz`. Le programme utilisé est aussi différent. C'est `dpkg-scansources`. La ligne de commande ressemblera à :

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

Remarquez que `dpkg-scansources` n'a pas besoin d'un fichier `override`. La ligne correspondante du `sources.list` est :

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

2.3 Comment décider quel est le meilleur miroir pour le fichier `sources.list` : `netselect`, `netselect-apt` ?

Un doute très fréquent, principalement parmi les nouveaux utilisateurs est : « Quel miroir Debian à mettre dans `sources.list` ? ». Il y a plusieurs façons de décider. Les experts ont probablement un script qui mesure le temps de réponse des principaux miroirs. Mais il y a un programme qui fait ça pour nous : **netselect**.

Pour installer `netselect`, faites comme d'habitude :

```
# apt-get install netselect
```

En lançant `netselect` sans paramètre, l'aide est affichée. En le lançant avec une liste de serveurs (miroirs) séparés par un espace, il retournera un résultat et un des serveurs. Ce résultat prend en considération l'estimation du temps de réponse et le nombre de sauts (les serveurs par lesquels une requête réseau passera pour atteindre la destination) et il est inversement proportionnel à la vitesse de téléchargement (donc le plus bas est le meilleur). Le serveur retourné est celui qui a eu le plus petit score (la liste complète des résultats peut être vue en ajoutant l'option `-vv`). Regardez cet exemple :

```
# netselect ftp.debian.org http.us.debian.org ftp.at.debian.org download.unes  
365 ftp.debian.org.br  
#
```

Cela signifie que, d'après les miroirs donnés en paramètre à `netselect`, `ftp.debian.org.br` était le meilleur, avec un résultat de 365. (Attention! Comme cela a été fait de mon ordinateur et que la topographie du réseau est très différente selon le point de contact, cette valeur n'est pas forcément la bonne vitesse sur d'autres ordinateurs).

Maintenant, mettez seulement le miroir le plus rapide trouvé par netselect dans le fichier `/etc/apt/sources.list` (voir 'Le fichier `/etc/apt/sources.list`' page 3) et suivez les conseils dans 'Gestion des paquets' page 9.

Note : on peut toujours trouver la liste des miroirs sur http://www.debian.org/mirror/mirrors_full.

À partir de la version 0.3.ds1, le paquet source netselect inclut le paquet binaire **netselect-apt**, qui effectue la procédure ci-dessus automatiquement. Entrez seulement la distribution en paramètre (celle par défaut est `stable`) et le fichier `sources.list` sera créé avec les meilleurs miroirs pour main et non-US et sera sauvegardé dans le répertoire courant. L'exemple suivant crée un `sources.list` de la distribution `stable` :

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

Rappelez-vous : Le fichier `sources.list` est créé dans le répertoire courant et doit être déplacé dans le répertoire `/etc/apt`.

Ensuite, suivez les conseils dans 'Gestion des paquets' page 9.

2.4 Ajouter un cédérom dans le fichier `sources.list`

Si vous voulez plutôt utiliser vos cédéroms pour installer vos paquets ou mettre à jour votre système automatiquement avec `apt`, vous pouvez les mettre dans votre `sources.list`. Pour le faire, vous pouvez utiliser le programme `apt-cdrom` comme ceci :

```
# apt-cdrom add
```

avec le cédérom Debian dans le lecteur. Il montera le cédérom, et si c'est un cédérom Debian valide, il regardera les informations des paquets sur le disque. Si la configuration de votre cédérom est inhabituelle, vous pouvez aussi utiliser les options suivantes :

```
-h           - Aide du programme
-d directory - Point de montage du cédérom
-r           - Renommer un cédérom reconnu
-m           - Pas de montage
-f           - Mode rapide, ne vérifie pas les fichiers paquet
-a           - Mode de vérification minutieux
```

Par exemple :

```
# apt-cdrom -d /home/kov/moncdrom add
```

Vous pouvez aussi identifier un cédérom sans l'ajouter à votre sources.list :

```
# apt-cdrom ident
```

Remarquez que ce programme fonctionne seulement si votre cédérom est correctement configuré dans votre `/etc/fstab`.

Chapitre 3

Gestion des paquets

3.1 Mise à jour de la liste des paquets disponibles.

Le système de paquets utilise sa propre base de données pour garder une trace des paquets qui sont installés, de ceux qui ne sont pas installés et de ceux qui peuvent être installés. Le programme `apt-get` utilise cette base de données pour retrouver comment installer les paquets demandés par l'utilisateur ainsi que pour retrouver les paquets supplémentaires nécessaires afin qu'un paquet sélectionné fonctionne correctement.

Pour mettre à jour cette liste, vous pouvez utiliser la commande `apt-get update`. Cette commande vérifie la liste des paquets trouvés dans les archives dans `/etc/apt/sources.list` : voir 'Le fichier `/etc/apt/sources.list`' page 3 pour plus d'informations sur ce fichier.

C'est une bonne idée d'exécuter cette commande de temps en temps pour vous garder, vous et votre système, informés des possibilités de mise à jour des paquets, particulièrement les mises à jour de sécurité.

3.2 Installation de paquets

Enfin, la procédure que vous attendiez tous ! Avec votre `sources.list` prêt et votre liste de paquets disponibles à jour, tout ce que vous avez à faire est de lancer `apt-get` pour obtenir le paquet que vous désirez installer. Par exemple, vous pouvez lancer :

```
apt-get install xchat
```

`apt` cherchera dans sa base de données la version la plus récente de ce paquet et le récupérera à partir de l'archive correspondante indiquée dans `sources.list`. Dans le cas où ce paquet dépend d'autres — comme c'est le cas ici — `apt` vérifiera les dépendances et installera les paquets nécessaires. Par exemple :

```
# 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]
```

Le paquet `nautilus` dépend des bibliothèques partagées citées, et `apt` les mettra dans l'archive. Si vous aviez donné les noms de ces bibliothèques sur la ligne de commande `apt-get`, `apt` ne vous aurait pas demandé si vous vouliez continuer : il aurait supposé que vous vouliez installer tous ces paquets.

Cela signifie que `apt` demande confirmation seulement quand il est nécessaire d'installer des paquets qui n'étaient pas spécifiés sur la ligne de commande.

Les options suivantes d'`apt-get` peuvent être utiles :

```
-h Ce texte d'aide
-d télécharge seulement - N'installe PAS, ni ne décompresse les paquets
-f Essaie de continuer si la vérification de l'intégrité échoue
-s Pas d'action. Effectue seulement une simulation
-y Suppose une réponse affirmative à toutes les requêtes et n'interroge pas
-u Affiche une liste des paquets à mettre à jour
```

Plusieurs paquets peuvent être sélectionnés pour installation sur une seule ligne. Les fichiers téléchargés sur le réseau sont placés dans le répertoire `/var/cache/apt/archives` pour une installation ultérieure.

Vous pouvez indiquer les paquets à retirer sur la même ligne de commande. Mettez seulement un « - » juste après le nom du paquet à enlever, comme ceci :

```
# 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]
```

Voyez la section ‘Suppression de paquets’ de la présente page pour plus de détails sur la suppression de paquets.

Si vous endommagez d’une manière ou d’une autre un paquet installé, ou si vous voulez simplement que les fichiers d’un paquet soient réinstallés avec la nouvelle version disponible, vous pouvez utiliser l’option `--reinstall` comme ça :

```
# 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
  upgraded.
Need to get 0B/182kB of archives. After unpacking 0B will be used.
Do you want to continue? [Y/n]
```

3.3 Suppression de paquets

Si vous ne voulez plus utiliser un paquet, vous pouvez le supprimer de votre système en utilisant `apt`. Pour ce faire, tapez seulement `apt-get remove paquet`. Par exemple :

```
# 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]
```

Comme vous pouvez le voir dans l’exemple ci-dessus, `apt` prend soin de supprimer les paquets qui dépendent du paquet dont vous avez demandé la suppression. Avec `apt`, il n’est pas possible de supprimer un paquet sans supprimer aussi les paquets dépendant de celui-ci.

Exécuter `apt-get` comme ci-dessus entraînera la suppression des paquets, mais leurs fichiers de configuration, s’il y en a, resteront intacts sur le système. Pour une suppression complète du paquet, lancez :

```
# 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]
```

Remarquez après les noms le signe « * ». Cela indique que les fichiers de configuration vont aussi être supprimés.

Comme avec la méthode `install`, vous pouvez utiliser un symbole avec `remove` pour inverser le sens d'un paquet en particulier. Dans le cas de la suppression, si vous ajoutez un « + » juste après le nom du paquet, le paquet sera installé au lieu d'être supprimé.

```
# 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]
```

Notez que `apt-get` liste les paquets supplémentaires qui seront installés (c'est-à-dire, les paquets dont l'installation est nécessaire au bon fonctionnement du paquet dont l'installation a été demandée), ceux qui seront supprimés, et ceux qui seront installés (toujours dans les paquets supplémentaires).

3.4 Mise à niveau des paquets

La mise à niveau de paquets est une vraie réussite du système `apt`. Une simple commande suffit : `apt-get upgrade`. Vous pouvez utiliser cette commande pour mettre à niveau les paquets d'une même distribution, et aussi pour mettre à niveau vers une nouvelle distribution, bien que la commande `apt-get dist-upgrade` soit préférée à cette dernière ; voir 'Mettre à niveau vers une nouvelle distribution' page suivante pour plus de détails.

Il est utile d'exécuter cette commande avec l'option `-u`. Cette option oblige `apt` à afficher la liste complète des paquets qui seront mis à niveau. Sans elle, vous ferez vos mises à niveau en aveugle. `apt` téléchargera les dernières versions de chaque paquet et les installera dans le bon ordre. C'est important de toujours lancer `apt-get update` avant de l'essayer. Voir la section 'Mise à jour de la liste des paquets disponibles.' page 9. Observez cet exemple :

```
# 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 procs 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]
```

Le processus est très simple. Notez que dans les premières lignes, `apt-get` dit que certains paquets ont été laissés de côté (NdT : « kept back »). Cela signifie qu'il y a de nouvelles versions de ces paquets qui ne seront pas installées pour plusieurs raisons. Les raisons possibles sont des dépendances cassées (un paquet dont il dépend n'a pas de version disponible en téléchargement) ou de nouvelles dépendances (le paquet dépend de nouveaux paquets depuis la dernière version).

Pour le premier cas, il n'y a pas de solution propre. Pour le second, il suffit d'exécuter `apt-get install` sur le paquet en question il téléchargera ainsi les dépendances. Une meilleure solution est d'utiliser `dist-upgrade`. Voir la section 'Mettre à niveau vers une nouvelle distribution' de la présente page.

3.5 Mettre à niveau vers une nouvelle distribution

Cette caractéristique d'`apt` permet de mettre à niveau tout un système Debian en une seule fois, soit par Internet ou soit par un nouveau cédérom (commandé ou téléchargé en image ISO).

Elle est aussi utilisée quand des changements dans les dépendances des paquets installés sont faits. Avec `apt-get upgrade`, ces paquets seraient laissés de côté (NdT :kept back).

Par exemple, supposons que vous utilisiez la révision 0 de la version stable de Debian et que vous achetiez un cédérom de la révision 3. Vous pouvez utiliser `apt` pour mettre à niveau votre système avec ce nouveau cédérom. Pour le faire, utilisez `apt-cdrom` (voir la section 'Ajouter un cédérom dans le fichier sources.list' page 6) pour ajouter le cédérom à votre `/etc/apt/sources.list` et lancez `apt-get dist-upgrade`.

Il est important de noter que `apt` cherche toujours la version du paquet la plus récente. Ainsi, si votre `/etc/apt/sources.list` listait une archive qui a une version plus récente que la version sur le cédérom, `apt` téléchargerait le paquet à partir de là.

Dans l'exemple de la section 'Mise à niveau des paquets' page précédente, nous avons vu que quelques paquets avaient été laissés de côté (NdT « kept back »). Nous allons maintenant résoudre ce problème par la méthode `dist-upgrade` :

```
# 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
  procs 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]

```

Notez maintenant que le paquet va être mis à niveau et des nouveaux paquets vont aussi être installés (les dépendances des paquets). Notez également que lilo est toujours laissé de côté. Il a probablement un problème plus important qu'une nouvelle dépendance. Nous pouvons le découvrir en lançant :

```

# 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]

```

Comme affiché ci-dessus, lilo a un nouveau conflit avec le paquet debconf-tiny, ce qui signifie qu'il ne peut pas être installé (ou mis à niveau) sans suppression de debconf-tiny.

Pour savoir ce qu'un paquet ajoute ou supprime, vous pouvez utiliser ceci :

```
# apt-get -o Debug::pkgProblemResolver=yes dist-upgrade
```

```
Reading Package Lists... Done
Building Dependency Tree... Done
Calculating Upgrade... Starting
Starting 2
Investigating python1.5
Package python1.5 has broken dep on python1.5-base
  Considering python1.5-base 0 as a solution to python1.5 0
  Holding Back python1.5 rather than change python1.5-base
Investigating python1.5-dev
Package python1.5-dev has broken dep on python1.5
  Considering python1.5 0 as a solution to python1.5-dev 0
  Holding Back python1.5-dev rather than change python1.5
  Try to Re-Instate python1.5-dev
Done
Done
The following packages have been kept back
  gs python1.5-dev
0 packages upgraded, 0 newly installed, 0 to remove and 2 not upgraded.
```

Cette fois, il est facile de remarquer que le paquet `python1.5-dev` ne peut pas être installé à cause de dépendances non satisfaites : `python1.5`.

3.6 Supprimer des paquets non utilisés : `apt-get clean` et `autoclean`

Lorsque vous installez un paquet, APT récupère les fichiers nécessaires depuis les hôtes listés dans `/etc/apt/sources.list`, les stocke dans un référentiel local (`/var/cache/apt/archives/`), et ensuite procède à l'installation, voir 'Installation de paquets' page 9.

Le référentiel local peut grandir dans le temps et utiliser beaucoup d'espace disque. Heureusement, APT fournit des outils pour gérer son référentiel local : les méthodes `apt-get clean` et `autoclean`.

`apt-get clean` supprime tout à part les fichiers verrou dans `/var/cache/apt/archives/` et `/var/cache/apt/archives/partial/`. Ainsi, si vous avez besoin de réinstaller un paquet, APT devra le retélécharger.

`apt-get autoclean` supprime seulement les paquets qui ne peuvent plus être téléchargés.

L'exemple suivant montre comment `apt-get autoclean` fonctionne :

```
# ls /var/cache/apt/archives/logrotate* /var/cache/apt/archives/gpm*
logrotate_3.5.9-7_i386.deb
logrotate_3.5.9-8_i386.deb
gpm_1.19.6-11_i386.deb
```

Il y a deux fichiers pour le paquet `logrotate` et un pour le paquet `gpm` dans `/var/cache/apt/archives`.

```
# apt-show-versions -p logrotate
logrotate/stable uptodate 3.5.9-8
# apt-show-versions -p gpm
gpm/stable upgradeable from 1.19.6-11 to 1.19.6-12
```

`apt-show-versions` montre que `logrotate_3.5.9-8_i386.deb` fournit la version à jour de `logrotate`, donc `logrotate_3.5.9-7_i386.deb` est inutile car une version plus récente du paquet peut être rapatriée.

```
# apt-get autoclean
Reading Package Lists... Done
Building Dependency Tree... Done
Del gpm 1.19.6-11 [145kB]
Del logrotate 3.5.9-7 [26.5kB]
```

Finalement, `apt-get autoclean` ne supprime que les anciens fichiers. Pour plus d'informations sur `apt-show-versions`, voyez 'Comment mettre à niveau un paquet d'une distribution Debian spécifique?' page 18.

3.7 Utiliser apt avec dselect

`dselect` est un programme qui aide les utilisateurs à sélectionner des paquets Debian pour les installer. Il est considéré comme un peu difficile et plutôt rébarbatif mais avec de la pratique vous pouvez maîtriser son interface console basée sur `ncurses`.

Un avantage de `dselect` est qu'il sait comment utiliser le fait que les paquets Debian « recommandent » et « suggèrent » d'autres paquets à installer. Pour utiliser ce programme, exécutez « `dselect` » en tant que `root`. Choisissez « `apt` » comme méthode d'accès. Ce n'est pas vraiment nécessaire, mais si vous n'utilisez pas de cédérom et que vous voulez télécharger des paquets à partir d'Internet, c'est la meilleure façon d'utiliser `dselect`.

Pour acquérir une meilleure compréhension de l'usage de `dselect`, lisez la documentation de `dselect` sur la page Debian <http://www.debian.org/doc/ddp>.

Après avoir fait votre sélection avec `dselect`, utilisez :

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

comme dans l'exemple ci-dessous :

```
# 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]
```

Comparez avec ce que nous avons vu lorsque nous avons lancé `apt-get dist-upgrade` sur le même système :

```
# 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]
```

Notez que beaucoup de paquets ci-dessus vont être installés parce que les autres paquets les « suggèrent » ou les « recommandent ». D'autres seront installés ou supprimés (dans le cas de `lbxproxy`, par exemple) grâce aux choix que nous avons faits lorsque nous naviguions dans la liste de paquets de `dselect`. `dselect` peut être un outil très puissant lorsqu'il est utilisé conjointement avec `apt`.

3.8 Comment garder un système mixte ?

L'utilisation d'une des versions de Debian en tant que distribution principale et un ou plusieurs paquets d'une autre branche peut être parfois intéressante pour certains utilisateurs.

Pour définir votre version principale de Debian, vous devriez modifier le fichier `/etc/apt/sources.list` afin qu'il contienne la ligne suivante :

```
APT::Default-Release "version";
```

Où, *version* est la version de Debian que vous souhaitez utiliser comme distribution principale. Les versions que vous pouvez utiliser sont `stable`, `testing` et `unstable`. Pour installer des paquets d'une autre version, vous devez utiliser APT de la manière suivante :

```
# apt-get -t distribution install paquet
```

Pour que cela fonctionne, il est nécessaire que vous ayez au moins une source APT de la distribution que vous voulez utiliser dans votre `/etc/apt/sources.list`, les paquets doivent exister dans cette source.

Vous pouvez aussi demander une version spécifique d'un paquet en utilisant la syntaxe suivante :

```
# apt-get install paquet= version
```

Par exemple, la ligne ci-dessous installera la version 2.2.4-1 du paquet `nautilus` :

```
# apt-get install nautilus=2.2.4-1
```

IMPORTANT : la distribution « unstable » de Debian est la version où les nouvelles versions des paquets Debian sont d'abord déposées. Cette distribution voit tous les changements des paquets, petits et plus importants qui affectent beaucoup de paquets ou tout le système. Pour cette raison, cette version *ne* devrait *pas* être utilisée par des utilisateurs non expérimentés ni par ceux qui ont besoin de stabilité.

La distribution « testing » n'est pas nécessairement meilleure que « unstable » car elle ne reçoit pas rapidement les mises à jour de sécurité. Pour un serveur ou un système en production, préférez toujours la distribution stable.

3.9 Comment mettre à niveau un paquet d'une distribution Debian spécifique ?

Pour les utilisateurs de distribution mixte, `apt-show-versions` est un moyen sûr pour mettre à niveau leur système, en contrôlant la part de la distribution la moins stable qu'ils veulent. Pour l'instant, il est possible de mettre à niveau seulement vos paquets unstable en exécutant après avoir installé le paquet `apt-show-versions` :

```
# apt-get install `apt-show-versions -u -b | grep unstable | cut -d ' ' -f 1`
```

3.10 Comment garder des versions spécifiques de paquets installés (complexe)

Vous avez sûrement eu l'occasion de modifier quelque chose dans un paquet et vous n'avez pas eu le temps ou l'envie de les transférer dans une nouvelle version du programme. Ou, par exemple, vous venez juste de passer à la distribution 3.0 de Debian, mais vous voulez continuer

avec la version 2.2 pour certains paquets. Vous pouvez « étiqueter » la version que vous avez installée de manière à ce qu'elle ne soit pas mise à niveau.

Utiliser cette fonctionnalité est simple. Vous avez seulement besoin de modifier le fichier `/etc/apt/preferences`.

Le format est simple :

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

Toutes les entrées doivent être séparées par une ligne vide. Par exemple, pour garder le paquet `sylpheed` que j'ai modifié pour utiliser « `reply-to-list` » dans la version 0.4.99, j'ai ajouté :

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

Notez que j'utilise un * (astérisque). C'est un « joker » ; cela signifie que je veux que cette « étiquette » soit valable pour toutes les versions commençant par 0.4.99. C'est parce que Debian donne un numéro de révision à ses paquets et je ne veux pas éviter l'installation de ces révisions. Et, par exemple, les versions 0.4.99-1 et 0.4.99-10 seront installées dès qu'elles seront disponibles. Notez que si vous avez modifié le paquet, vous ne voudrez pas procéder ainsi.

La priorité de l'étiquetage aide à déterminer si un paquet correspondant aux lignes « `Package :` » et « `Pin :` » sera installé, plus la priorité d'un paquet est grande, plus il est probable que le paquet correspondant sera installé. Si vous souhaitez plus de détails, vous pouvez lire `apt_preferences(7)`, mais quelques exemples devraient vous donner quelques idées de base. Les exemples suivant décrivent les effets du positionnement du champ de priorité à différentes valeurs dans l'exemple sur `sylpheed` ci-dessus.

- 1001** La version 0.4.99 de `sylpheed` ne sera jamais remplacée par `apt`. Si elle est disponible, `apt` installera la version 0.4.99 même s'il doit remplacer un paquet avec une version supérieure. Seuls les paquets avec une priorité supérieure à 1000 peuvent remplacer un paquet existant par une version inférieure.
- 1000** L'effet est le même qu'avec une priorité de 1001, à l'exception qu'`apt` refusera d'installer la version 0.4.99 si une autre version est installée.
- 990** La version 0.4.99 sera remplacée seulement si une version supérieure est disponible dans la version de la distribution « préférée » en utilisant la variable `APT::Default-Release` (voir « Comment garder un système mixte ? » page 17 ci-dessus).
- 500** Toute version supérieure à 0.4.99 disponible dans n'importe quelle version de la distribution sera préférée à la version 0.4.99, mais la version 0.4.99 sera toujours préférée à une version moins élevée.
- 100** Toute version de `sylpheed` disponible dans n'importe quelle version de la distribution sera préférée à la version 0.4.99, comme toute version supérieure de `sylpheed` installée ; la version 0.4.99 sera donc installée seulement si aucune version n'est déjà installée. C'est la priorité des paquets installés.

-1 Les priorités négatives sont aussi permises et empêchent la version 0.4.99 d'être installée.

Une étiquette peut être spécifiée sur la version, la distribution ou l'origine d'un paquet.

Pour étiqueter une version, nous avons vu qu'on pouvait utiliser les numéros de version de manière littérale aussi bien que les jokers pour spécifier plusieurs versions en une fois.

L'option `release` dépend du fichier `Release` d'un référentiel d'`apt` ou d'un cédérom. Cette option peut être sans intérêt si vous utilisez des référentiels de paquets qui ne contiennent pas ce fichier. Vous pouvez voir le contenu de ce fichier `Release` dans `/var/lib/apt/lists`. Les paramètres de la distribution sont : `a` (archive), `c` (composants), `v` (version), `o` (origine) et `l` (label).

Un exemple :

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

Dans cet exemple, nous avons choisi les versions Debian 2.2* (qui peuvent être 2.2r2, 2.2r3 — cela satisfait les distributions qui incluent les correctifs de sécurité et d'autres mises à jour très importantes), le référentiel `stable`, la section `main` (opposée à `contrib` ou `non-free`) et l'origine et le label `Debian`. L'origine (`o=`) définit qui a produit ce fichier `Release`, le label (`l=`) définit le nom de la distribution : `Debian` pour Debian elle-même et `Progeny` pour Progeny, par exemple. Un simple fichier `Release` :

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

Chapitre 4

Aides très utiles

4.1 Comment installer un paquet localement : *equivs*

Parfois, des personnes veulent utiliser une version spécifique d'un programme dont est disponible seulement le code source, sans paquet Debian. Mais le système de paquets peut être perturbé quand vous le faites. Supposons que vous voulez compiler une nouvelle version de votre serveur de courriels. Tout est bien, mais beaucoup de paquets dépendent d'un MTA (« Mail Transport Agent ») dans Debian. Parce que vous avez installé quelque chose que vous avez compilé vous-même, le système de paquet ne connaît rien sur lui.

C'est là que *equivs* entre en scène. Pour l'utiliser, installez le paquet du même nom. *Equivs* crée un paquet vide capable de réaliser toutes les dépendances, en faisant croire au système de paquets que les dépendances sont satisfaites.

Avant que nous ne commençons, il est bon de vous rappeler qu'il y a d'autres méthodes plus sûres pour compiler un programme déjà empaqueté pour Debian avec différentes options, et l'on ne devrait pas utiliser *equivs* pour remplacer les dépendances quand on ne sait pas ce que l'on fait. Voir la section 'Travailler avec des paquets sources' page 31 pour plus d'informations.

Continuons avec l'exemple du MTA, vous venez juste d'installer votre *postfix* nouvellement compilé et vous continuez avec l'installation de *mutt*. Tout à coup vous découvrez que *mutt* veut installer un autre MTA. Mais vous avez déjà le vôtre.

Allez dans un répertoire (*/tmp*, par exemple) et lancez :

```
# equivs-control nom
```

Remplacez *nom* par le nom du fichier de contrôle que vous voulez créer. Le fichier sera créé comme suit :

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

```
Package: <entrer le nom du paquet~ par défaut equivs-dummy>
Version: <entrer ici la version~ par défaut 1.0>
Maintainer: <votre nom et votre adresse mail~par défaut nom d'utilisateur>
Pre-Depends: <paquets>
Depends: <paquets>
Recommends: <paquets>
Suggests: <paquet>
Provides: <paquet (virtuel)>
Architecture: all
Copyright: <fichier copyright~ par défaut GPL2>
Changelog: <fichier changelog~ par défaut un changelog générique>
Readme: <fichier README.Debian~ par défaut un générique>
Extra-Files: <fichiers supplémentaires pour le répertoire doc, séparés par de
Description: <description courte~ par défaut quelques mots judicieux>
description longue et informations
.
second paragraphe
```

Nous avons juste besoin de modifier cela pour faire ce que nous voulons. Examinons le format des champs et leurs descriptions. Il n'y a pas besoin ici de tout expliquer, faisons ce qui est nécessaire :

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

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

Oui, c'est tout. `mutt` dépend de `mail-transport-agent`, c'est un paquet virtuel fourni par tous les MTA, je peux simplement nommer le paquet `mail-transport-agent`, mais je préfère utiliser le schéma du paquet virtuel, en utilisant `Provides`.

Maintenant vous avez seulement besoin de construire le paquet :

```
# equivs-build nom
dh_testdir
touch build-stamp
dh_testdir
dh_testroot
dh_clean -k
# Ajoutez ici les commandes pour installer le paquet dans 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 `nom' in
  `../nom_1.0_all.deb'.
```

The package has been created.

Attention, the package has been created in the current directory,

Et installer le `.deb` résultant.

Comme vous pouvez le voir, il y a plusieurs utilisations d'`equivs`. L'une d'entre elles peut éventuellement créer un paquet `mes-favoris`, qui dépend des programmes que vous installez habituellement, par exemple. Laisser aller simplement votre imagination, mais soyez prudent.

Il est important de noter qu'il y a des exemples de fichiers contrôle dans `/usr/share/doc/equivs/examples`. Allez-y jeter un oeil.

4.2 Suppression des fichiers de locale inutiles : `localepurge`

Beaucoup d'utilisateurs Debian utilisent une seule locale. Par exemple, un utilisateur brésilien utilise, généralement, la locale `pt_BR` tout le temps et ne se soucie pas de l'es.

`localepurge` est un outil très utile pour ces utilisateurs. Vous pouvez libérer beaucoup d'espace en ayant seulement la locale que vous utilisez. Taper simplement `apt-get install localepurge`.

C'est très simple à configurer, les questions de `debconf` guident l'utilisateur dans une configuration pas-à-pas. Soyez très attentif à la réponse de la première question, une mauvaise réponse peut supprimer tous les fichiers de locale, y compris ceux que vous utilisez. Le seul moyen de récupérer ces fichiers est de réinstaller tous les paquets qui les fournissent.

4.3 Comment savoir quels paquets peuvent être mis à niveau

`apt-show-versions` est un programme qui affiche quels paquets du système peuvent être mis à jour et diverses informations utiles. l'option `-u` affiche une liste des paquets pouvant être mis à niveau :

```
$ apt-show-versions -u  
libeel0/unstable upgradeable from 1.0.2-5 to 1.0.2-7  
libeel-data/unstable upgradeable from 1.0.2-5 to 1.0.2-7
```

Chapitre 5

Avoir des informations sur les paquets

Il y a quelques interfaces pour le système `apt` qui facilitent indubitablement l'obtention d'une liste de paquets qui sont disponibles pour l'installation ou qui sont déjà installés, de même que pour rechercher la section dans laquelle se trouve un paquet, sa priorité, sa description, etc.

Mais... notre but ici est d'apprendre comment utiliser le vrai `apt`. Donc comment pouvez-vous retrouver le nom d'un paquet que vous voulez installer ?

Nous avons un bon nombre de recours pour cette tâche. Nous commencerons avec `apt-cache`. Ce programme est utilisé par le système `apt` pour le maintien de sa base de données. Nous jetterons un bref coup d'oeil à ses applications les plus pratiques.

5.1 Recherche de noms de paquets

Supposons, par exemple, que vous voulez vous rappeler les bons vieux jours de l'Atari 2600. Vous voulez utiliser `apt` pour installer un émulateur Atari, puis télécharger quelques jeux. Vous pouvez faire :

```
# 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
```

Nous trouvons différents paquets relatifs à ce que nous cherchons, accompagnés d'une brève description. Pour avoir des informations sur un paquet spécifique, je peux alors utiliser :

```
# 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: 11b3e86a41a60falc4b334dd96c1d4b5
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
```

Depuis cette sortie vous avez plein de détails sur le paquet que vous voulez (ou ne voulez pas) installer, accompagnés de la description complète du paquet. Si le paquet est déjà installé sur votre système et qu'il y a une nouvelle version, vous verrez les informations des deux versions. Par exemple :

```
[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
 screen)
 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
 screen)
 or call Lilo from other Boot-Loaders to jump-start the Linux kernel.
```

Notez que le premier de la liste est le paquet disponible et le second est celui déjà installé. Pour plus d'informations générales sur un paquet, vous pouvez utiliser :

```
# apt-cache showpkg penguin-command
Package: penguin-command
Versions:
1.4.5-1 (/var/lib/apt/lists/download.sourceforge.net_debian_dists_unstable_main
inary-i386_Packages)(/var/lib/dpkg/status)

Reverse Depends:
Dependencies:
1.4.5-1 - libc6 (2 2.2.1-2) libpng2 (0 (null)) libsdl-mixer1.1 (2 1.1.0)
  libsdl1.1 (0 (null)) zlib1g (2 1:1.1.3)
Provides:
1.4.5-1 -
Reverse Provides:
```

Et pour rechercher de quels paquets il dépend :

```
# apt-cache depends penguin-command
penguin-command
  Depends: libc6
  Depends: libpng2
  Depends: libsdl-mixer1.1
  Depends: libsdl1.1
  Depends: zlib1g
```

En résumé, nous avons une série d'armes que nous pouvons utiliser pour rechercher le nom d'un paquet que nous voulons.

5.2 Utilisation de dpkg pour trouver le nom des paquets

Une des solutions pour localiser le nom d'un paquet est de connaître le nom d'un fichier important trouvé dans le paquet. Par exemple, pour trouver le paquet qui fournit un fichier « .h » particulier dont vous avez besoin pour compiler, vous pouvez lancer :

```
# 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
```

ou :

```
# dpkg -S /usr/include/stdio.h
libc6-dev: /usr/include/stdio.h
```

Pour trouver le nom de paquets installés sur votre système, ce qui est utile, par exemple, si vous prévoyez de nettoyer votre disque dur, vous pouvez lancer :

```
# dpkg -l | grep mozilla
ii  mozilla-browser 0.9.6-7          Mozilla Web Browser
```

Le problème avec cette commande est qu'elle peut « casser » le nom des paquets. Dans l'exemple ci-dessus, le nom complet du paquet est mozilla-browser. Pour arranger cela, vous pouvez utiliser la variable d'environnement COLUMNS comme ceci :

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

ou la description ou une partie de celle-ci comme cela :

```
# apt-cache search "Mozilla Web Browser"
mozilla-browser - Mozilla Web Browser
```

5.3 Comment installer des paquets « à la demande » ?

Vous compilez un programme et, tout à coup : patatras ! Il y a une erreur parce qu'il faut un fichier .h que vous n'avez pas. Le programme auto-apt peut vous sauver de ces scénarios. Il vous demande d'installer des paquets nécessaires, en stoppant le processus, et en continuant une fois que le paquet est installé.

Ce que vous avez simplement à faire est de lancer :

```
# auto-apt run commande
```

Où « commande » est la commande à exécuter qui devrait nécessiter des fichiers qui ne sont pas disponibles. Par exemple :

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

Il vous demandera d'installer les paquets nécessaires et appellera `apt-get` automatiquement. Si vous êtes sous *X*, une interface graphique remplacera l'interface texte par défaut.

Pour être efficace, `auto-apt` conserve des bases de données qui doivent être à jour. C'est fait en appelant les commandes `auto-apt update`, `auto-apt updatedb` et `auto-apt update-locale`.

5.4 Comment savoir à quel paquet appartient un fichier ?

Si vous voulez installer un paquet et que vous ne parvenez pas à découvrir quel est son nom en cherchant avec `apt-cache`, mais que vous connaissez le nom du programme lui-même, ou quelque autre fichier appartenant à ce paquet, vous pouvez alors utiliser `apt-file` pour retrouver le nom de ce paquet. Ceci est fait comme suit :

```
$ apt-file search nomdufichier
```

Il fonctionne comme `dpkg -S`, mais il vous affichera aussi les paquets non installés qui contiennent ce fichier. Il peut aussi être utilisé pour trouver quel paquet contient les fichiers d'en-tête manquant à la compilation d'un programme, bien que `auto-apt` soit une meilleure solution pour résoudre ce problème, voir 'Comment installer des paquets « à la demande » ?' page ci-contre.

Vous pouvez lister le contenu d'un paquet en exécutant :

```
$ apt-file list nomdupaquet
```

`apt-file` récupère une base de données des fichiers contenus dans tous les paquets, comme le fait `auto-apt` et elle doit être à jour. C'est fait en exécutant :

```
# apt-file update
```

Par défaut, `apt-file` et `auto-apt` utilisent la même base de données, voir 'Comment installer des paquets « à la demande » ?' page précédente.

5.5 Comment rester informé sur les changements dans les paquets ?

Tous les paquets installent dans leur répertoire de documentation un fichier appelé `changelog.Debian.gz` qui contient la liste des changements faits sur le paquet depuis la dernière version. Vous pouvez lire ces fichiers à l'aide de `zless`, par exemple ; mais ce n'est pas quelque chose de très simple, après une mise à niveau du système complète, de consulter les changelog de tous les paquets mis à niveau.

Il y a une solution pour automatiser cette tâche au moyen d'un outil appelé `apt-listchanges`. Pour commencer, vous avez besoin d'installer le paquet `apt-listchanges`. Pendant l'installation du paquet, `Debconf` le configurera. Certaines questions peuvent ne pas vous être affichées car cela dépend de la priorité fixée dans `Debconf`. Répondez aux questions comme vous le voulez.

La première question vous demande si vous souhaitez qu'`apt-listchanges` affiche les changements. Vous pouvez vous les faire envoyer par courriel, ce qui est une bonne idée pour les mises à niveau automatiques ou vous pouvez simplement demander de les afficher à l'aide d'un outil de défilement tel que `less`, vous pourrez donc inspecter les changements avant de permettre la poursuite de la mise à niveau. Si vous ne voulez pas qu'`apt-listchanges` soit lancé automatiquement lors des mises à niveau, vous pouvez répondre `none`.

Après que `apt-listchanges` a été installé, aussitôt que des paquets sont téléchargés (ou récupérés sur un cédérom ou un disque monté) par `apt`, il affichera la liste des changements faits sur ces paquets avant de les installer.

Chapitre 6

Travailler avec des paquets sources

6.1 Télécharger un paquet source

Il est courant dans le monde du logiciel libre d'étudier le code source ou éventuellement de faire des corrections sur du code bogué. Pour le faire, vous aurez besoin de télécharger les sources du programme. Le système `apt` fournit une solution facile pour obtenir le code source des nombreux programmes contenus dans la distribution, en incluant tous les fichiers pour créer un `.deb` pour le programme.

Un autre usage courant des sources Debian est d'adapter une version d'un programme plus récente, de la distribution *unstable*, par exemple, pour l'utiliser dans la distribution *stable*. Compiler un paquet avec *stable* générera des `.deb` avec des dépendances ajustées pour que ce paquet puisse être disponible dans cette distribution.

Pour accomplir ceci, l'entrée `deb-src` dans votre `/etc/apt/sources.list` doit pointer vers *unstable*. Elle doit être aussi permise (décommentée). Voir la section 'Le fichier `/etc/apt/sources.list`' page 3.

Pour télécharger un paquet source, vous devez utiliser la commande suivante :

```
$ apt-get source nomdupaquet
```

Cela téléchargera trois fichiers : un `.orig.tar.gz`, un `.dsc` et un `.diff.gz`. Dans le cas où les paquets sont faits spécialement pour Debian, le dernier de ceux-ci n'est pas téléchargé et le premier n'a généralement pas « `orig` » dans le nom.

Le fichier `.dsc` est utilisé par `dpkg-source` pour dépaqueter le paquet source dans le répertoire `nomdupaquet-version`. Avec chaque paquet source téléchargé, il y a un répertoire `debian/` qui contient les fichiers nécessaires pour la création d'un paquet `.deb`.

Pour construire automatiquement le paquet lorsqu'il est téléchargé, ajoutez seulement `-b` à la ligne de commande, comme ceci :

```
$ apt-get -b source nomdupaquet
```

Si vous décidez de ne pas créer de `.deb` lorsque vous le téléchargez, vous pouvez le créer plus tard en exécutant :

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

à l'intérieur du répertoire créé pour le paquet après le téléchargement. Pour installer le paquet construit par la commande ci-dessus, vous devez directement utiliser le gestionnaire de paquets, comme ceci :

```
# dpkg -i fichier.deb
```

Il y a une différence entre les méthodes `apt-get source` et ses autres méthodes. La méthode `source` peut être utilisée par un utilisateur normal, sans avoir besoin des droits root. Les fichiers sont téléchargés dans le répertoire à partir duquel la commande `apt-get source paquet` a été appelée.

6.2 Paquets nécessaires pour la compilation d'un paquet source

Normalement, des en-têtes et des bibliothèques dynamiques spécifiques doivent être présentes pour qu'un paquet source puisse être compilé. Tous les paquets source ont un champ dans leur fichier contrôle appelé « `Builds-Depends` : » qui indique quels paquets supplémentaires sont nécessaires pour la construction à partir des sources.

`apt` possède une solution simple pour télécharger ces paquets. Exécutez juste `apt-get build-dep paquet`, où « `paquet` » est le nom du paquet que vous allez construire. Par exemple :

```
# 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-implib-dev implib-progs libgnome-dev libgnorba-de
  libgpmgl-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]
```

Les paquets qui seront installés sont les paquets nécessaires pour que `gmc` soit compilé correctement. C'est important de noter que cette commande ne cherche pas le paquet source du programme à compiler. Vous devrez par conséquent exécuter `apt-get source` séparément pour le récupérer.

Si tout ce que vous voulez est de vérifier quels paquets sont nécessaires pour construire un paquet donné, il existe une variante de la commande `apt-cache show` (voir 'Avoir des informations sur les paquets' page 25) qui affichera, parmi toutes les informations disponibles, la ligne `Build-Depends` qui liste ces paquets.

```
# apt-cache showsrc paquet
```


Chapitre 7

Comment traiter les erreurs ?

7.1 Erreurs courantes

Des erreurs arriveront souvent, la plupart d'entre elles causées par les utilisateurs qui ne font pas attention. Ce qui suit est une liste des erreurs les plus fréquemment rapportées et comment les traiter.

Si vous recevez un message qui ressemble à celui ci-dessous quand vous essayez d'exécuter `apt-get install paquet...`

```
Reading Package Lists... Done
Building Dependency Tree... Done
W: Couldn't stat source package list 'http://people.debian.org unstable/
  Packages'
  (/var/state/apt/lists/people.debian.org_%7ekov_debian_unstable_Packages) - s
  (2 No such file or directory)
W: You may want to run apt-get update to correct these missing files
E: Couldn't find package penguineyes
```

vous avez oublié d'exécuter `apt-get update` après vos derniers changements dans le fichier `/etc/apt/sources.list`.

Si l'erreur ressemble à :

```
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
```

quand vous essayez n'importe quelle méthode `apt-get` autre que `source`, vous n'avez pas les permissions `root`, parce que vous l'exécutez en utilisateur normal.

Il y a une erreur similaire à celle ci-dessus qui arrive lorsque vous exécutez deux copies d'`apt-get` en même temps, ou éventuellement si vous essayez d'exécuter `apt-get` pendant

qu'un processus `dpkg` est actif. La seule méthode qui peut être utilisée simultanément avec les autres est la méthode `source`.

Si une installation s'interrompt au milieu du processus et que vous ne pouvez plus installer ou supprimer de paquets, essayez d'exécuter ces deux commandes :

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

et ensuite essayez à nouveau. Il sera peut être nécessaire d'exécuter la seconde des commandes ci-dessus plus d'une fois. C'est une leçon importante pour ces aventuriers qui utilisent « `unstable` ».

Si vous rencontrez l'erreur « `E : Dynamic MMap ran out of room` » en lançant `apt-get update`, ajoutez la ligne suivante dans `/etc/apt/apt.conf` :

```
APT::Cache-Limit 10000000;
```

7.2 Où puis-je trouver de l'aide ?

Si vous êtes tourmenté par le doute, consultez l'importante documentation disponible pour le système de gestion des paquets Debian. Les `--help` et les pages de manuel peuvent être une énorme aide pour vous, comme le peut la documentation contenue dans les répertoires `/usr/share/doc` comme `/usr/share/doc/apt`.

Si la documentation ne suffit pas à faire disparaître votre peur, essayez de rechercher la réponse sur les listes de discussions Debian. Vous pouvez obtenir plus d'informations des listes d'utilisateurs spécifiques sur le site Web Debian : <http://www.debian.org>.

Souvenez-vous que ces listes et ressources doivent être utilisées seulement par les utilisateurs Debian ; les utilisateurs d'autres systèmes trouveront un meilleur support dans les communautés de leur propre distribution.

Chapitre 8

Quelles distributions supportent apt ?

Il a ici les noms de quelques distributions qui utilisent apt :

Debian GNU/Linux (<http://www.debian.org>) - C'est pour cette distribution que apt a été développé

Conectiva (<http://www.conectiva.com.br>) - Ce fut la première distribution à porter apt pour l'utiliser avec les rpm

Libranet (<http://libranet.com>)

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>)

Alt Linux (<http://www.altlinux.ru/>)

Red Hat (<http://www.redhat.com/>)

Sun Solaris (<http://www.sun.com/>)

SuSE (<http://www.suse.de/>)

Yellow Dog Linux (<http://www.yellowdoglinux.com/>)

Chapitre 9

Crédits

Un grand merci à mes amis du projet Debian-BR, et à Debian elle-même, qui m'aide constamment et me donne toujours la force de continuer à travailler pour le bénéfice de l'humanité, aussi bien qu'en m'aidant dans mon but de sauver le monde. :)

Je veux aussi remercier le CIPSGA pour l'énorme aide qu'elle a donnée à notre projet et pour tous les projets libres qu'elle met en marche par de belles idées.

Et des remerciements particuliers :

Yooseong Yang <yooseong@debian.org> —pour la traduction du manuel en coréen.

Michael Bramer <grisu@debian.org> —pour la suggestion de l'inclusion de la section sur la conservation d'une version spécifique.

Bryan Stillwell <bryan@bokeoa.com> —pour les diverses rustines et corrections qu'il m'a envoyées.

Pawel Tecza <pawel.tecza@poczta.fm> —pour les diverses corrections qu'il m'a envoyées ainsi que pour la traduction polonaise.

Hugo Mora <h.mora@melix.com.mx> —pour la traduction espagnole.

Luca Monducci <luca.mo@tiscali.it> —pour la traduction italienne.

Tomohiro KUBOTA <kubota@debian.org> —pour la traduction japonaise.

Pablo Lorenzoni <spectra@debian.org> pour l'écriture de la section sur netselect.

Steve Langasek <vorlon@netexpress.net> —pour la traduction du manuel en anglais.

Arnaldo Carvalho de Melo <acme@conectiva.com.br> —pour sa contribution à la liste des distributions supplémentaires qui supportent maintenant apt : Mandrake, PLD et Vine.

Erik Rossen <rossen@freesurf.ch> —pour l'astuce sur la variable COLUMNS pour dpkg -l.

Ross Boylan <RossBoylan@stanfordalumni.org> —pour l'astuce sur l'utilisation de -o Debug : :pkgProblemResolver=yes.

Matt Kraai <kraai@debian.org> —pour les diverses rustines et corrections qu'il m'a envoyées.

Aaron M. Ucko <ucko@debian.org> —pour ses relectures et corrections.

Jon Åslund <d98-jas@nada.kth.se> —pour l'écriture de la section sur apt-file.

Chapitre 10

Nouvelles versions de ce manuel

Ce manuel fut créé par le projet Debian-BR (<http://debian-br.cipsga.org.br>), dans le but d'aider à l'utilisation journalière de Debian.

Des nouvelles versions de ce document seront disponibles dans la page du projet, à <http://www.debian.org/doc/ddp>

Les commentaires et critiques peuvent m'être envoyés directement par courriel à <kov@debian.org>.