Raphaël **Hertzog**

Cahiers de l'Admin Debian

Collection dirigée par Nat **Makarévitch**

Avec la contribution de Christophe Le Bars,
Sébastien Blondeel et Florence Henry

© Groupe Eyrolles, 2004,

ISBN 2-212-11398-6

EYROLLES

6

Maintenance et mise à jour : les outils APT

Ce qui rend Debian si populaire auprès des administrateurs, c'est la facilité avec laquelle il est possible d'y installer des logiciels et de mettre à jour le système complet. Cet avantage unique est dû en grande partie au programme *APT*, outil dont les administrateurs de Falcot SA se sont donc empressés d'étudier les possibilités.

SOMMAIRE

- Renseigner le fichier « sources.list »
- Commande **apt-get**
  - Initialisation
  - Installation et suppression
  - Mise à jour
  - Options de configuration
  - Gérer les priorités associées aux paquets
- Commande **apt-cache**
- Frontaux : **aptitude**, **synaptic**, **gnome-apt**
- Vérification d'authenticité des paquets
- Mise à jour automatique
  - Configuration de **dpkg**
  - Configuration d'APT
  - Configuration de **debconf**
  - Gestion des interactions en ligne de commande
  - La combinaison miracle

MOTS-CLEFS

- apt-get
- apt-cache
- aptitude
- synaptic
- sources.list
- apt-cdrom

B.A.-BA **Compression `gzip` et `bzip2`**

Une extension .gz dénote un fichier compressé avec l'utilitaire **`gzip`**. De la même manière, .bz2 indique une compression par **`bzip2`**. **`gzip`** est l'utilitaire Unix traditionnel pour compresser les fichiers, rapide et efficace. **`bzip2`**, plus récent, obtient de meilleurs taux de compression mais nécessite plus de temps de calcul pour comprimer un fichier.

APT est l'abréviation de *Advanced Package Tool* (outil avancé pour les paquets). Ce que ce programme a d'« avancé », c'est la manière d'aborder la problématique des paquets. Il ne se contente pas de les évaluer un par un, mais les considère dans leur ensemble et réalise la meilleure combinaison possible de paquets en fonction de tout ce qui est disponible et compatible (au sens des dépendances).

APT a besoin qu'on lui fournisse une « liste de sources » : c'est le fichier /etc/apt/sources.list qui décrira les différents emplacements possibles (ou « sources ») des paquets Debian. APT devra ensuite rapatrier la liste des paquets disponibles pour chacune de ces sources, ainsi que leurs en-têtes. Il réalise cette opération en téléchargeant les fichiers Packages.gz ou Packages.bz2 (cas d'une source de paquets binaires) et Sources.gz ou Sources.bz2 (cas d'une source de paquets sources) et en analysant leur contenu.

Renseigner le fichier « sources.list »

Le fichier /etc/apt/sources.list contient sur chaque ligne active une description de source, qui se décompose en 3 parties séparées par des blancs.

Le premier champ indique le type de la source :

- « `deb` » pour des paquets binaires,
- « `deb-src` » pour des paquets sources.

Le deuxième champ indique l'URL de base de la source (combinée aux noms de fichier présents dans les fichiers Packages.gz, elle doit donner une URL complète valide) : il peut s'agir d'un miroir Debian ou de toute autre archive de paquets mise en place par des tierces personnes. L'URL peut débuter par `file://` pour indiquer une source locale située dans l'arborescence de fichiers du système, par `http://` pour indiquer une source accessible depuis un serveur web, ou encore par `ftp://` pour une source disponible sur un serveur FTP.

Le dernier champ a une syntaxe variable selon que la source correspond à un miroir Debian ou non. Dans le cas d'un miroir Debian, on nomme la distribution choisie (`stable`, `testing`, `unstable` ou leurs noms de code actuels respectifs) puis les différentes sections souhaitées (choisies parmi `main` , `contrib`, et `non-free`). Dans les autres cas, on indique simplement le sous-répertoire de la source désirée (on y trouve souvent le simple « ./ » dénotant l'absence de sous-répertoire — les paquets sont alors directement à l'URL indiquée).

D'une manière générale, le contenu d'un fichier sources.list standard pourrait être le suivant :

VOCABULAIRE **Les archives main, contrib et non-free**

Debian prévoit trois sections pour différencier les paquets selon les licences prévues par les auteurs des programmes respectifs. *Main* (archive principale) rassemble tous les paquets répondant pleinement aux principes du logiciel libre selon Debian.
L'archive *non-free* (non libre), spéciale, contient des logiciels ne répondant pas (totalement) à ces principes mais néanmoins distribuables librement. Cette archive, qui ne fait pas officiellement partie de Debian, est un service rendu aux utilisateurs qui pourraient avoir besoin de ses logiciels — mais Debian recommande toujours l'emploi d'alternatives libres.
Contrib (contributions) est un stock de logiciels libres ne fonctionnant pas sans certains éléments non libres. Il peut s'agir de programmes dépendant de logiciels de la section *non-free* ou de fichiers non libres tels que des ROM de jeux, des BIOS de consoles, etc. On y trouve encore des logiciels libres dont la compilation nécessite des éléments propriétaires. C'était au début le cas de la suite bureautique OpenOffice.org, qui avait besoin d'un environnement Java propriétaire.

EXEMPLE **Fichier « /etc/apt/sources.list »**

```
# Mises à jour de sécurité
deb http://security.debian.org/ stable/updates main contrib non-free

# Miroir Debian
deb http://ftp.fr.debian.org/debian stable main contrib non-free
deb http://ftp.fr.debian.org/debian-non-US stable/non-US main contrib non
   -free
deb-src http://ftp.fr.debian.org/debian stable main contrib non-free
deb-src http://ftp.fr.debian.org/debian-non-US stable/non-US main contrib
    non-free
```

Ce fichier référence toutes les sources de paquets associées à la version stable de Debian. Si vous souhaitez utiliser *testing* ou *unstable*, il faudra évidemment y ajouter (ou les remplacer par) les lignes adéquates.

Le fichier sources.list comporte encore d'autres types d'entrées : celles décrivant des cédéroms Debian à votre disposition. Contrairement aux autres entrées, un cédérom n'est pas disponible en permanence puisqu'il faut l'insérer dans le lecteur et qu'un seul disque peut être lu à la fois — les cédéroms sont donc gérés un peu différemment. On ajoutera ces entrées à l'aide du petit programme **apt-cdrom**, habituellement invoqué avec le paramètre **add**. Il demande alors d'insérer le CD dans le lecteur, parcourt son contenu à la recherche de fichiers Packages, qu'il utilisera pour mettre à jour sa base de données de paquets disponibles (opération habituellement réalisée par la commande **apt-get update**). Dès lors, **apt-get** pourra vous demander d'insérer le CD en question s'il a besoin de l'un de ses paquets.

POUR ALLER PLUS LOIN **Les paquets experimental**

L'archive de paquets *experimental*, présente sur tous les miroirs Debian, contient des paquets qui n'ont pas encore leur place dans la version *unstable* pour cause de qualité insuffisante — ce sont fréquemment des versions de développement ou pré-versions (alpha, bêta, *release candidate*...) des logiciels. Il arrive également qu'un paquet y soit envoyé après avoir subi des changements importants, potentiellement sources de problèmes. Le mainteneur cherche alors à débusquer ceux-ci avec l'aide des utilisateurs avancés capables de gérer les soucis importants. Après cette première phase, le paquet passe dans *unstable*, au public beaucoup plus vaste, et où il subira donc des tests de bien plus grande envergure.

On réservera donc *experimental* aux utilisateurs qui n'ont pas peur de casser leur système puis de le réparer. Cette distribution peut quand même permettre de rapatrier ponctuellement un paquet que l'on tient à essayer ou utiliser. C'est d'ailleurs la logique standard que Debian lui associe, puisque son ajout dans le fichier sources.list d'APT n'entraîne pas l'emploi systématique des paquets qui s'y trouvent. La ligne qu'il convient d'ajouter est la suivante :

```
deb http://ftp.fr.debian.org/debian ../project/
   experimental main contrib non-free
```

Signalons encore qu'*experimental* ne dispose pas de la même infrastructure de maintenance et de portabilité qu'*unstable* : les paquets n'y sont notamment pas automatiquement compilés pour toutes les architectures.

Commande `apt-get`

APT est un projet relativement vaste, incluant à l'origine une interface graphique. Pour le moment, il ne dispose pourtant que d'une interface en ligne de commande : **`apt-get`**.

En effet, ses interfaces graphiques n'émanent pas directement de ses développeurs : **`synaptic`**, **`gnome-apt`**, **`aptitude`** (mode texte), **`wajig`**, etc. Le frontal le plus recommandé, **`aptitude`**, est celui employé lors de l'installation initiale. Sa syntaxe en ligne de commande, très similaire à celle d'**`apt-get`**, en fait un sérieux candidat de remplacement.

Initialisation

Un préalable à tout travail avec APT est la mise à jour de la liste des paquets disponibles, qui s'effectue avec un simple **`apt-get update`**. Selon le débit de votre connexion, cette opération peut durer un peu puisqu'elle télécharge un certain nombre de fichiers Packages.(gz|bz2) (voire Sources.(gz|bz2)), devenus assez volumineux au fil de la croissance de Debian (3 Mo pour le plus gros Packages.gz, correspondant à la section `main`). Évidemment, une installation à partir d'un jeu de cédéroms ne nécessite aucun téléchargement — cette opération est alors très rapide.

Installation et suppression

APT permet d'ajouter ou de supprimer des paquets sur le système, respectivement avec **`apt-get install`** ***`paquet`*** et **`apt-get remove`** ***`paquet`***. Dans chaque cas, APT installera automatiquement les dépendances nécessaires ou supprimera les paquets dépendant du paquet en cours de destruction. L'option

COMMUNAUTÉ **Ressources non officielles : `apt-get.org` et `mentors.debian.net`**

Il existe de nombreuses sources non officielles de paquets Debian, mises en place par des utilisateurs avancés ayant recompilé certains logiciels, par des programmeurs mettant leur création à disposition, et même par des développeurs Debian proposant des préversions de leur paquet en ligne. Un site web fut mis en place pour trouver plus facilement ces sources alternatives. On y trouve une quantité impressionnante de sources de paquets Debian prêtes à être intégrées dans les fichiers sources.list. Attention toutefois à ne pas rajouter n'importe quoi. Chaque source est en effet prévue pour une version particulière de Debian (celle employée pour compiler les paquets concernés) ; on veillera à maintenir une certaine cohérence dans ce que l'on choisit d'installer.

▸ http://www.apt-get.org

Signalons également l'existence du site `mentors.debian.net`, qui regroupe des paquets réalisés par des prétendants au statut de développeur Debian officiel ou par des volontaires souhaitant créer des paquets Debian sans passer par ce processus d'intégration. Ces paquets sont donc fournis sans aucune garantie de qualité ; prenez garde à bien les tester avant d'envisager de les déployer sur de nombreux ordinateurs. Pensez en particulier au fait qu'installer un paquet revient à donner les droits *root* à son concepteur (les scripts d'initialisation sont en effet exécutés avec ces droits, et le concepteur peut y placer n'importe quelle commande). Évaluez le degré de confiance que vous accordez à l'intégrité du paquet par rapport à l'importance de la machine sur laquelle vous pensez l'installer.

▸ http://mentors.debian.net

--purge demande une désinstallation complète — les fichiers de configuration sont alors également supprimés.

Si le fichier sources.list mentionne plusieurs distributions, il est possible de préciser la version du paquet à installer. On peut demander un numéro de version précis avec **apt-get install *paquet=version***, mais on préférera en général simplement indiquer la distribution d'origine du paquet (*stable*, *testing* ou *unstable*) avec la syntaxe **apt-get install *paquet/distribution***. Avec cette commande, on pourra donc revenir à une ancienne version d'un paquet (si par exemple on sait qu'elle fonctionne bien).

EXEMPLE **Installation de la version unstable de SpamAssassin**

```
# apt-get install spamassassin/unstable
```

ASTUCE **Installer la même sélection de paquets plusieurs fois**

Il est parfois souhaitable de pouvoir installer systématiquement la même liste de paquets sur plusieurs ordinateurs. C'est possible assez facilement.
Récupérons d'abord la liste des paquets installés sur l'ordinateur qui servira de « modèle » à dupliquer.

```
$ dpkg --get-selections >liste-pkg
```

Le fichier liste-pkg contient alors la liste des paquets installés.
Il faut alors transférer le fichier liste-pkg sur les ordinateurs à mettre à jour et y employer les commandes suivantes :

```
# dpkg --set-selections <liste-pkg
# apt-get dselect-upgrade
```

La première commande enregistre les vœux de paquets à installer, que l'invocation d'**apt-get** exauce ensuite !

ASTUCE **Supprimer et installer en même temps**

Il est possible, en ajoutant un suffixe, de demander à **apt-get** d'installer certains paquets et d'en supprimer d'autres sur la même ligne de commande. Lors d'une commande **apt-get install**, ajoutez un « - » aux noms des paquets que vous souhaitez supprimer. Lors d'une commande **apt-get remove**, ajoutez un « + » aux noms des paquets que vous souhaitez installer.
L'exemple suivant montre deux manières d'installer *paquet1* et de supprimer *paquet2*.

```
# apt-get install paquet1 paquet2-
[...]
# apt-get remove paquet1+ paquet2
[...]
```

Mise à jour

Des mises à jour régulières sont recommandées, car elles mettront en place les derniers correctifs de sécurité. Pour cela, on tapera **apt-get upgrade** (évidemment précédé par **apt-get update**). Cette commande cherche les mises à jour des paquets installés, réalisables sans ajouter ou supprimer de paquets. Autrement dit, l'objectif est d'assurer une mise à jour la moins intrusive possible.

Remarquons cependant qu'**apt-get** retiendra en général le numéro de version le plus récent (à l'exception des paquets *experimental*, ignorés par défaut quel que soit leur numéro de version). Si vous avez mentionné *testing* ou *unstable* dans votre sources.list, **apt-get upgrade** migrera tout votre système *stable* en *testing* ou *unstable*, ce qui n'est peut-être pas l'effet recherché.

Pour indiquer à **apt-get** d'utiliser telle ou telle distribution pour ses recherches de paquets mis à jour, il faut utiliser l'option **-t** ou **--target-release** (version

POUR ALLER PLUS LOIN **Cache des fichiers « .deb »**

apt-get conserve dans le répertoire /var/cache/apt/archives/ une copie de chaque fichier .deb téléchargé. Dans le cas de mises à jour fréquentes, ce répertoire peut rapidement occuper beaucoup d'espace disque avec plusieurs versions de chaque paquet ; il convient donc d'y faire régulièrement le tri. Deux commandes existent pour cela : **apt-get clean** vide le répertoire ; **apt-get autoclean** ne supprime que les paquets qui, n'étant plus téléchargeables (car ayant disparu du miroir Debian), sont clairement inutiles (le paramètre de configuration `APT::Clean-Installed` permet d'empêcher la suppression de fichiers .deb encore actuellement installés).

cible) ou **--default-release** (version par défaut), suivie du nom de la distribution en question (exemple : **apt-get -t stable upgrade**). Pour éviter de spécifier cette option à chaque invocation d'**apt-get**, vous pouvez ajouter `APT::Default-Release "stable";` dans le fichier /etc/apt/apt.conf.d/local.

Pour les mises à jour plus importantes, comme lors du basculement d'une version majeure de Debian à la suivante, il faut utiliser **apt-get dist-upgrade** (mise à jour de la distribution). Cela effectue la mise à jour même s'il y a des paquets obsolètes à supprimer et de nouvelles dépendances à installer. C'est également la commande employée par ceux qui exploitent quotidiennement la version *unstable* de Debian et suivent ses évolutions au jour le jour. Elle est si simple qu'elle parle d'elle-même : c'est bien cette fonctionnalité qui a fait la renommée d'APT.

Options de configuration

Outre les éléments de configuration déjà mentionnés, il est possible de configurer quelques aspects d'APT en ajoutant des directives dans un fichier du répertoire /etc/apt/apt.conf.d/. Rappelons par exemple qu'il est possible pour APT d'indiquer à **dpkg** d'ignorer les erreurs de collision de fichiers en précisant `DPkg::Options { "--force-overwrite"; }`.

Si l'accès au Web n'est possible qu'à travers un mandataire (*proxy*), il faut ajouter une ligne semblable à `Acquire::http::proxy "http://`*monproxy*`:3128"`. Pour un *proxy* FTP, on écrira `Acquire::ftp::proxy "ftp://`*monproxy*`"`. Découvrez par vous-même les autres options de configuration en consultant la page de manuel `apt.conf(5)`, avec la commande **man apt.conf**.

Gérer les priorités associées aux paquets

Une des problématiques les plus importantes dans la configuration d'APT est la gestion des priorités des différentes sources de paquets. Il arrive en effet assez fréquemment qu'on souhaite compléter une distribution d'un ou deux paquets plus récents issus de *testing*, *unstable*, ou *experimental*. Il est possible d'affecter une priorité à chaque paquet disponible (un même paquet pouvant recevoir plusieurs priorités, selon sa version ou sa distribution d'appartenance). Ces priorités dicteront à APT son comportement : pour chaque paquet, il sélectionnera systématiquement la version de plus haute priorité (sauf si cette version est plus ancienne que celle installée et que la priorité associée est inférieure à 1000).

APT définit un certain nombre de priorités par défaut. Chaque version de paquetage déjà installée a une priorité de 100, une version non installée reçoit une priorité de 500 sauf si elle fait partie de la distribution cible (*Target Release*), qu'on spécifie avec l'option **-t** ou la directive `APT::Target-Release`, auquel cas sa priorité passe à 990.

On modifiera ces priorités en intervenant sur le fichier /etc/apt/preferences pour y ajouter des entrées de quelques lignes décrivant le nom du ou des paquets concernés, leur version, leur origine, et leur nouvelle priorité.

APT refusera toujours d'installer une version antérieure d'un paquet (portant un numéro de version inférieur à celui de la version actuelle), sauf si la priorité du paquet concerné est supérieure à 1000. APT installera toujours la version de priorité la plus élevée. Si deux versions ont la même priorité, APT installe la plus récente (de numéro de version le plus grand). Si deux paquets de même version ont la même priorité mais diffèrent dans leur contenu, APT installe la version qui n'est pas installée (cette règle doit couvrir le cas d'une mise à jour de paquet sans incrément — normalement indispensable — du numéro de révision).

Concrètement, un paquet de priorité inférieure à 0 ne sera jamais installé. Un paquet de priorité comprise entre 0 et 100 ne sera installé que si aucune autre version du même paquet n'est installée. Avec une priorité comprise entre 100 et 500, le paquet ne sera installé que s'il n'en existe aucune version plus récente, installée ou disponible dans une autre distribution). Un paquet de priorité entre 500 et 990 ne sera installé qu'à défaut de version plus récente, installée ou disponible dans la distribution cible. Une priorité entre 990 et 1000 fera installer le paquet, sauf si la version installée est plus récente. Une priorité supérieure à 1000 provoquera l'installation du paquet, même si cela force APT à installer une version plus ancienne que la version actuelle.

Quand APT consulte le fichier /etc/apt/preferences, il prend d'abord en compte les entrées les plus précises (souvent, celles spécifiant le paquet concerné) puis les

VOCABULAIRE **Répertoire en « .d »**

Les répertoires de suffixe .d sont de plus en plus souvent employés. Chacun abrite des fichiers ventilant un fichier de configuration. Ainsi, tous les fichiers contenus dans /etc/apt/apt.conf.d/ constituent les instructions de configuration d'APT. APT les lira dans l'ordre alphabétique, de sorte que les derniers pourront modifier un élément de configuration défini dans l'un des premiers.
Cette structure apporte une certaine souplesse à l'administrateur de la machine et aux mainteneurs de paquets. En effet, l'administrateur peut facilement modifier la configuration du logiciel en déposant un fichier tout prêt dans le répertoire en question sans devoir modifier de fichier existant. Les mainteneurs de paquets ont la même problématique lorsqu'ils doivent adapter la configuration d'un autre logiciel pour assurer une parfaite cohabitation avec le leur. Mais la charte Debian interdit explicitement toute modification de fichiers de configuration relevant d'autres paquets, interdiction justifiée par le fait que seuls les utilisateurs sont habilités à intervenir ainsi. Rappelons en effet que **dpkg** invite l'utilisateur, lors d'une installation, à choisir la version du fichier de configuration qu'il souhaite conserver lorsqu'une modification y est détectée. Toute modification externe du fichier déclencherait une telle requête, qui ne manquerait pas de perturber l'administrateur certain de n'avoir rien touché.
En l'absence de répertoire .d, il est impossible à un paquet externe d'adapter les réglages d'un logiciel sans en modifier le fichier de configuration. Il doit alors inviter l'utilisateur à intervenir lui-même, en documentant les opérations à effectuer dans le fichier /usr/share/doc/<*paquet*>/README.Debian.
Selon les applications, le répertoire .d est directement exploité, ou géré par un script externe qui en concaténera tous les fichiers pour créer le fichier de configuration à proprement parler. Il est alors important d'exécuter ce script après toute intervention dans ce répertoire pour que les dernières modifications soient prises en compte. De même, on prendra garde à ne pas travailler directement sur le fichier de configuration construit automatiquement, sous peine de tout perdre à la prochaine exécution du script. Le choix de cette méthode fut dicté par des gains en terme de souplesse de configuration compensant largement les petites complications induites. Elle concerne les options de configuration des modules du noyau (le fichier /etc/modules.conf est généré par le script **update-modules** à partir du répertoire /etc/modutils).

plus génériques (incluant par exemple tous les paquets d'une distribution). Si plusieurs entrées génériques existent, la première correspondant au paquet dont on cherche la priorité est utilisée. Les critères de sélection disponibles comprennent notamment le nom du paquet et sa source de provenance. Chaque source est identifiée par un ensemble d'informations contenues dans un fichier Release, qu'APT télécharge en même temps que les fichiers Packages.gz. Ce dernier spécifie l'origine (habituellement « Debian » pour les paquets des miroirs officiels, mais il peut s'agir du nom d'une personne ou d'un organisme proposant une archive de paquets Debian) ; il précise également le nom de l'archive (habituellement *stable*, *testing*, *unstable* ou *experimental* pour les archives standards fournies par Debian) ainsi que sa version (par exemple 3.1 pour *Debian sarge*). Étudions-en la syntaxe précise au travers de quelques cas vraisemblables d'emploi de ce mécanisme.

Supposons qu'on souhaite utiliser exclusivement des paquets provenant de la version stable de Debian, sans jamais installer ceux des autres versions sauf demande explicite. Il est possible d'écrire ce qui suit dans le fichier /etc/apt/preferences :

```
Package: *
Pin: release a=stable
Pin-Priority: 900

Package: *
Pin: release o=Debian
Pin-Priority: -10
```

`a=stable` précise le nom de l'archive concernée. `o=Debian` restreint l'entrée aux paquets dont l'origine est « Debian ».

Supposons maintenant que nous disposions d'un serveur ayant installé de nombreux programmes spécifiques à la version 5.8 de Perl et que l'on veuille s'assurer qu'aucune mise à jour n'en installera une autre version. On peut pour cela utiliser cette entrée :

```
Package: perl
Pin: version 5.8*
Pin-Priority: 1001
```

La documentation de référence sur ce fichier de configuration est disponible dans la page de manuel `apt_preferences(5)`.

CAS PARTICULIER **Priorité d'experimental**

Si vous avez inscrit *experimental* dans votre fichier sources.list, les paquets correspondants ne seront quasiment jamais installés, leur priorité APT étant de 50. C'est un cas particulier qui évite que les gens installent des paquets *experimental* par erreur, et les oblige à opérer en tapant **`apt-get install paquet/experimental`** — ils ont donc pleinement conscience des risques encourus. Il est possible, mais ce n'est *pas* recommandé, de considérer les paquets *experimental* comme ceux des autres distributions en leur affectant une priorité de 500 grâce à une entrée dans le fichier /etc/apt/preferences :

```
Package: *
Pin: release a=experimental
Pin-Priority: 500
```

ASTUCE **Commentaires dans « /etc/apt/preferences »**

Il n'existe pas de syntaxe standard pour introduire des commentaires dans le fichier /etc/apt/preferences, mais il est possible d'y expliquer le rôle de chaque entrée à l'aide d'un ou plusieurs champs « *Explanation :* » (explication) placés en début de bloc :

```
Explanation: Le paquet xfree86-xserver contenu dans
Explanation: experimental peut être utilisé
Package: xfree86-xserver
Pin: release a=experimental
Pin-Priority: 500
```

Commande `apt-cache`

La commande **`apt-cache`** permet de consulter un certain nombre d'informations stockées dans la mémoire *cache* d'APT. Ses fonctionnalités comprennent notamment la possibilité de rechercher des paquets à l'aide de mots-clés, en tapant **`apt-cache search`** ***`mot-clé`***. On peut aussi consulter les en-têtes des différentes versions disponibles d'un paquet avec **`apt-cache show`** ***`paquet`***. Cette commande produira la description du paquet ainsi que ses dépendances, le nom de son mainteneur, etc.

Certaines fonctions ne servent que bien plus rarement. Ainsi, **`apt-cache policy`** permet de consulter les priorités des différentes sources de paquets ainsi que celles des paquets qui bénéficient d'un traitement particulier. On peut encore citer **`apt-cache dumpavail`** qui affiche les en-têtes de toutes les versions disponibles de tous les paquets. **`apt-cache pkgnames`** affiche une liste de tous les paquets existants dans la mémoire *cache*.

Frontaux : `aptitude`, `synaptic`, `gnome-apt`

APT est un programme C++ dont la majorité du code est déportée dans la bibliothèque partagée **`libapt-pkg`**. C'est pourquoi il est relativement facile de réaliser un frontal qui s'intègre dans sa famille d'outils.

`aptitude` est un programme interactif pour la console qui permet de naviguer dans la liste des paquets installés et disponibles, de consulter l'ensemble des informations, et de les marquer en vue d'une installation ou d'une suppression. Il exploite intelligemment le concept de « tâche ». On peut choisir une tâche complète à installer ou supprimer et consulter la liste des paquets inclus dans une tâche afin d'en sélectionner un sous-ensemble plus limité. Par ailleurs, **`aptitude`** se rappelle quels paquets ont été installés explicitement ou par le jeu des dépendances, ce qui lui permet de supprimer automatiquement les paquets rendus superflus par une mise à jour.

`synaptic` et **`gnome-apt`** sont deux gestionnaires équivalents de paquets Debian, en mode graphique (ils utilisent GTK+/GNOME).

Vérification d'authenticité des paquets

Étant donné l'importance qu'accordent les administrateurs de Falcot SA à la sécurité, ils veulent s'assurer de ne jamais installer que des paquets garantis provenir de Debian et pas altérés en cours de route. Pour cela, il existe un mécanisme de signature des paquets.

Cette signature n'est pas directe : le fichier signé est un fichier Release placé sur les miroirs Debian et qui donne la liste des différents fichiers Packages.gz, accompagnés de leur somme de contrôle MD5 (pour vérifier que leur contenu n'a pas été altéré). Ces fichiers Packages renferment à leur tour une liste de paquets Debian et leurs sommes de contrôle MD5, afin de garantir que leur contenu n'a pas lui non plus été altéré.

La version d'APT disponible dans Debian Sarge n'est pas encore capable de vérifier systématiquement ces signatures, mais il en existe une version *experimental* implémentant cette fonctionnalité, et qui sera probablement intégrée sous peu à la version *unstable*.

En attendant, Anthony Towns a développé une solution intermédiaire qui consiste en un script séparé, à exécuter juste après **`apt-get update`**. Ce script rapatrie les fichiers Release et Release.gpg associés à chaque source de paquets et vérifie qu'APT a téléchargé les bons fichiers Packages. Il ne fonctionne parfaitement qu'avec les entrées du fichier /etc/apt/sources.list respectant la syntaxe longue `deb http://`*`miroir`*`/debian `*`distribution composant(s)`*.

Pour l'exploiter, il faut également télécharger les parties publiques des clés avec lesquelles Debian signe les fichiers Release et les ajouter au trousseau de clés (*keyring*) trustedkeys.gpg, utilisé par **`gpgv`** pour vérifier les signatures :

```
# wget http://people.debian.org/~ajt/apt-check-sigs
[...]
# wget http://ftp-master.debian.org/ziyi_key_2004.asc
[...]
# wget http://ftp-master.debian.org/ziyi_key_2003.asc
[...]
# touch ~/.gnupg/trustedkeys.gpg
# chmod 0600 ~/.gnupg/trustedkeys.gpg
# gpg --no-options --no-default-keyring --keyring trustedkeys.gpg --
  import ziyi_key_2004.asc
gpg: key 1DB114E0: public key "Debian Archive Automatic Signing Key
   (2004) <ftpmaster@debian.org>" imported
gpg:        Quantité totale traitée: 1
gpg:                       importée: 1  (RSA: 1)

# gpg --no-options --no-default-keyring --keyring trustedkeys.gpg --
  import ziyi_key_2003.asc
[ ... ]
# gpg --keyring trustedkeys.gpg --list-keys --fingerprint
  ftpmaster@debian.org

pub  1024R/1DB114E0 2004-01-15 Debian Archive Automatic Signing Key
   (2004) <ftpmaster@debian.org>
 Empreinte de la clé = D051 FE3A 848D CABD 4625  787A 6FFA 8EF9 1DB1 14
   E0

pub  1024D/38C6029A 2002-12-20 Debian Archive Automatic Signing Key
   (2003) <ftpmaster@debian.org>
 Empreinte de la clé = EB2F A2AF 170D 2359 26A7  7BF3 B629 A24C 38C6 029
   A
```

`apt-check-sigs` désactive les sources non signées ou dépourvues de signature valide en changeant le nom des copies locales des fichiers Packages exploités par APT. Cela désactive immédiatement les sources correspondantes (jusqu'au pro-

SÉCURITÉ **Vérifier le fingerprint des clés**

Le téléchargement des clés publiques employées pour vérifier la signature des archives n'étant pas une opération sûre, il faut en contrôler le *fingerprint* (l'empreinte) en le comparant à celui des clés créées originellement, que leurs possesseurs publient (on s'assure ainsi qu'il s'agit bien des mêmes clés).

▸ http://lists.debian.org/debian-devel-announce/2004/01/msg00007.html

chain **`apt-get update`**). L'emploi régulier d'**`apt-check-sigs`** vous garantit ainsi qu'aucun paquet illégitime ne sera installé.

```
# apt-get update
[ ... ]
# ./apt-check-sigs

Checking sources in /etc/apt/sources.list:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


You should take care to ensure that the distributions you're
downloading are the ones you think you are downloading, and that they
are as up to date as you would expect (testing and unstable should be
no more than two or three days out of date, stable-updates no more
than a few weeks or a month).

Source: deb http://security.debian.org/ stable/updates main contrib non-
  free
  o Origin: Debian/Debian-Security
  o Suite: stable/woody
  o Date: Mon, 16 Feb 2004 10:54:57 UTC
  o Description: Debian 3.0 Security Updates
  o Signed by: Debian Archive Automatic Signing Key (2004) <
    ftpmaster@debian.org>
  o Okay: main contrib non-free

Source: deb http://localhost/debian stable main contrib non-free
  o Origin: Debian/Debian
  o Suite: stable/woody
  o Date: Thu, 20 Nov 2003 18:57:17 UTC
  o Description: Debian 3.0r2 Released 20th November 2003
  o Signed by: Debian Archive Automatic Signing Key (2004) <
    ftpmaster@debian.org>
  o Okay: main contrib non-free

Source: deb http://localhost/debian-non-US stable/non-US main contrib
  non-free
  o Origin: Debian/Debian
  o Suite: stable/woody
  o Date: Thu, 20 Nov 2003 00:40:58 UTC
  o Description: Debian 3.0r2 Released 20th November 2003
  * NO VALID SIGNATURE
  * PROBLEMS WITH main (NOCHECK, NOCHECK)
  * PROBLEMS WITH contrib (NOCHECK, NOCHECK)
  * PROBLEMS WITH non-free (NOCHECK, NOCHECK)

Source: deb http://marillat.free.fr/ stable main
  o Origin: Christian Marillat/Unofficial Packages Free
  o Suite: stable/woody
  o Date: Mon, 16 Feb 2004 11:53:50 UTC
  o Description: Unofficial Packages marillat.free.fr
  * COULDN'T CHECK SIGNATURE BY KEYID: 07DC563D1F41B907
  * NO VALID SIGNATURE
  * PROBLEMS WITH main (NOCHECK, NOCHECK)

Results
~~~~~~~
```

```
The contents of the following files in /var/lib/apt/lists could not be
validated due to the lack of a signed Release file, or the lack of an
appropriate entry in a signed Release file. This probably means that
the maintainers of these sources are slack, but may mean these sources
are being actively used to distribute trojans.  The files have been
renamed to have the extension .FAILED and will be ignored by apt.

    localhost_debian-non-US_dists_stable_non-US_main_binary-i386_Release
    localhost_debian-non-US_dists_stable_non-US_main_binary-
      i386_Packages
    localhost_debian-non-US_dists_stable_non-US_contrib_binary-
      i386_Release
    localhost_debian-non-US_dists_stable_non-US_contrib_binary-
      i386_Packages
    localhost_debian-non-US_dists_stable_non-US_non-free_binary-
      i386_Release
    localhost_debian-non-US_dists_stable_non-US_non-free_binary-
      i386_Packages
    marillat.free.fr_dists_stable_main_binary-i386_Release
    marillat.free.fr_dists_stable_main_binary-i386_Packages
```

Dans l'exemple ci-dessus, deux sources ont été désactivées : celle qui correspond au serveur « non-US » de Debian, situé hors du territoire des États-Unis d'Amérique — archive signée avec la clé de 2003, qui a désormais expiré... il s'agit là d'un problème au niveau de Debian — et celle qui correspond à une archive tierce fournie par Christian Marillat. Nous aurions pu valider cette dernière en ajoutant la clé de Christian (dont l'empreinte est `1D7F C53F 80F8 52C1 88F4 ED0B 07DC 563D 1F41 B907`) dans le trousseau de clés trustedkeys.gpg.

Mise à jour automatique

Afin de limiter les interventions sur chaque PC, les administrateurs de Falcot SA veulent pouvoir automatiser au maximum leurs mises à jour. Pour cela, il faut supprimer les interactions avec les utilisateurs.

Configuration de `dpkg`

Nous avons déjà vu comment empêcher **`dpkg`** de demander confirmation du remplacement d'un fichier de configuration (avec les options **`--force-confdef --force-confold`**). Il reste trois éléments à prendre en compte : les interactions générées par APT lui-même, celles provenant de **`debconf`**, et les interactions en ligne de commande intégrées dans les scripts de configuration des paquets.

Configuration d'APT

En ce qui concerne APT, la réponse est simple. Il suffit de lui préciser l'option **`-y`** ou **`--assume-yes`**, qui répondra « oui » automatiquement à toutes les questions qu'il aurait pu poser.