5.4. Manipuler des paquets avec `dpkg`

`dpkg` est la commande de base pour manipuler des paquets Debian sur le système. Si vous disposez de fichiers `.deb`, c'est `dpkg` qui permet de les installer ou d'analyser leur contenu. Toutefois ce programme n'a qu'une vision partielle de l'univers Debian : il sait ce qui est installé sur le système et ce qu'on lui indique en ligne de commande, mais, n'ayant aucune connaissance de tous les autres paquets disponibles, il échouera si une dépendance n'est pas satisfaite. Un outil comme `apt-get` établira au contraire la liste des dépendances pour tout installer aussi automatiquement que possible.

***NOTE* `dpkg` ou `apt-get` ?**

Il faut voir `dpkg` comme un outil système (de *backend*) et `apt-get` comme un outil plus proche de l'utilisateur, qui permet de dépasser les limitations du précédent. Mais ces deux outils marchent de concert, chacun a ses spécificités et convient mieux à certaines tâches.

5.4.1. Installation de paquets

`dpkg` est avant tout l'outil qui permet d'installer un paquet Debian déjà accessible (car il ne peut télécharger). On utilise pour cela son option `-i` ou `--install`.

Exemple 5.2. Installation d'un paquet avec `dpkg`

```
# dpkg -i man-db_2.6.2-1_amd64.deb
(Reading database ... 96357 files and directories currently installed.)
Preparing to replace man-db 2.6.1-3 (using man-db_2.6.2-1_amd64.deb) ...
Unpacking replacement man-db ...
Setting up man-db (2.6.2-1) ...
Building database of manual pages ...
```

On peut observer les différentes étapes suivies par `dpkg` ; on sait ainsi à quel niveau s'est produite une éventuelle erreur. L'installation peut aussi s'effectuer en deux temps, dépaquetage puis configuration. `apt-get` en tire profit pour limiter le nombre d'invocations de `dpkg` (coûteuses en raison du chargement de la base de données en mémoire — notamment la liste des fichiers déjà installés).

Exemple 5.3. Dépaquetage et configuration séparée

```
# dpkg --unpack man-db_2.6.2-1_amd64.deb
(Reading database ... 96357 files and directories currently installed.)
Preparing to replace man-db 2.6.2-1 (using man-db_2.6.2-1_amd64.deb) ...
Unpacking replacement man-db ...
#  dpkg --configure man-db
Setting up man-db (2.6.2-1) ...
Building database of manual pages ...
```

Parfois, `dpkg` échouera à installer un paquet et renverra une erreur ; si on lui ordonne de l'ignorer, il se contentera alors d'émettre un avertissement : c'est à cela que servent les différentes options `--force-*`. La commande `dpkg --force-help` ou la documentation de cette commande donneront la liste complète de ces options. L'erreur la plus fréquente, et qui ne manquera pas de vous concerner tôt ou tard, est la collision de fichiers. Lorsqu'un paquet contient un fichier déjà installé par un autre paquet, `dpkg` refuse de l'installer. Les messages suivants apparaissent alors :

```
Unpacking libgdm (from .../libgdm_3.8.3-2_amd64.deb) ...
dpkg: error processing /var/cache/apt/archives/libgdm_3.8.3-2_amd64.deb (--
unpack):
 trying to overwrite '/usr/bin/gdmflexiserver', which is also in package gdm3
3.4.1-9
```

Dans ce cas, si vous pensez que remplacer ce fichier ne constitue pas un risque important pour la stabilité de votre système (ce qui est presque toujours le cas), vous pouvez employer l'option `--force-overwrite`, qui indiquera à `dpkg` d'ignorer cette erreur et d'écraser le fichier.
Bien que de nombreuses options `--force-*` existent, seule `--force-overwrite` est susceptible d'être employée de manière régulière. Ces options existent juste pour des situations exceptionnelles et il convient de s'en passer autant que possible afin de respecter les règles imposées par le mécanisme de paquetage — règles qui garantissent la cohérence et la stabilité du système, rappelons-le.

***ATTENTION* Du bon usage de `--force-*`**

Si l'on n'y prend garde, l'usage d'une option `--force-*` peut mener à un système où les commandes de la famille APT refuseront de fonctionner. En effet, certaines de ces options permettent d'installer un paquet alors même qu'une dépendance n'est pas satisfaite, ou en dépit d'un conflit mentionné. Le résultat est un système incohérent du point de vue des dépendances et les commandes APT refuseront d'exécuter la moindre action, sauf celles qui permettent de revenir dans un état cohérent (cela consiste souvent à installer la dépendance manquante ou à supprimer le paquet problématique). Cela se traduit souvent par un message comme celui-ci, obtenu après avoir installé une nouvelle version de rdesktop en ignorant sa dépendance sur une version plus récente de la libc6 :

```
# apt-get dist-upgrade
[...]
Vous pouvez lancer « apt-get -f install » pour corriger ces problèmes.»
Les paquets suivants contiennent des dépendances non satisfaites :
  rdesktop: Dépend: libc6 (>= 2.5) mais 2.3.6.ds1-13etch7 est installé
E: Dépendances manquantes. Essayez d'utiliser l'option -f.
```

L'administrateur aventureux qui est certain de la justesse de son analyse peut choisir d'ignorer une dépendance ou un conflit, donc d'employer l'option `--force-*` correspondante. Dans ce cas, s'il veut pouvoir continuer d'employer `apt-get` ou `aptitude`, il doit éditer `/var/lib/dpkg/status` pour supprimer/modifier la dépendance ou le conflit qu'il a choisi d'outrepasser.
Cette manipulation relève d'un bricolage honteux et ne devrait — si possible — jamais être employée. Bien souvent, une solution plus propre consiste à recompiler le paquet dont la dépendance ne convient pas (voir Section 15.1, « Recompiler un paquet depuis ses sources ») voire

à récupérer une version plus récente (potentiellement corrigée) sur un site comme celui de `backports.debian.org` (voir Section 6.1.2.4, « Rétroportages vers stable »).

5.4.2. Suppression de paquet

En invoquant `dpkg` avec l'option `-r` ou `--remove` suivie d'un nom de paquet, on supprime celui-ci. Cette suppression n'est cependant pas complète : tous les fichiers de configuration, scripts de configuration, fichiers de logs (journaux système) et toutes les données d'utilisateur manipulées par le paquet subsistent. L'intérêt de les conserver est de désactiver un programme en le désinstallant tout en se ménageant la possibilité de le remettre en service rapidement et à l'identique. Pour tout supprimer pour de bon, il convient de faire appel à l'option `-P` ou `--purge` suivie du nom de paquet.

Exemple 5.4. Suppression puis purge du paquet *debian-cd*

```
# dpkg -r debian-cd
(Reading database ... 97747 files and directories currently installed.)
Removing debian-cd ...
# dpkg -P debian-cd
(Reading database ... 97401 files and directories currently installed.)
Removing debian-cd ...
Purging configuration files for debian-cd ...
```

5.4.3. Consulter la base de données de `dpkg` et inspecter des fichiers `.deb`

***B.A.-BA* Syntaxe des options**

La plupart des options sont disponibles en version « longue » (un ou plusieurs mots significatifs, précédés d'un tiret double) ou « courte » (une seule lettre, souvent l'initiale d'un mot de la version longue, et précédée d'un seul tiret). Cette convention est si fréquente qu'elle est normée POSIX. Avant de conclure cette section, nous allons décrire un certain nombre d'options de `dpkg` permettant d'interroger sa base de données interne afin d'obtenir des informations. En donnant d'abord les options longues puis les options courtes correspondantes (qui prendront évidemment les mêmes éventuels arguments), citons `--listfiles` *`paquet`* (ou `-L`), qui affiche la liste des fichiers installés par ce paquet ; `--search` *`fichier`* (ou `-S`), qui retrouve le paquet d'où provient ce fichier ; `--status` *`paquet`* (ou `-s`), qui affiche les en-têtes d'un paquet installé ; `--list` (ou `-l`), qui affiche la liste des paquets connus du système ainsi que leur état d'installation ; `--contents` *`fichier.deb`* (ou `-c`), qui affiche la liste des fichiers contenus dans le paquet Debian spécifié ; `--info` *`fichier.deb`* (ou `-I`), qui affiche les en-têtes de ce paquet Debian.

Exemple 5.5. Diverses requêtes avec `dpkg`

```
$ dpkg -L base-passwd
/.
/usr
/usr/sbin
/usr/sbin/update-passwd
/usr/share
/usr/share/man
/usr/share/man/ru
/usr/share/man/ru/man8
```

```
/usr/share/man/ru/man8/update-passwd.8.gz
/usr/share/man/pl
/usr/share/man/pl/man8
/usr/share/man/pl/man8/update-passwd.8.gz
/usr/share/man/man8
/usr/share/man/man8/update-passwd.8.gz
/usr/share/man/fr
/usr/share/man/fr/man8
/usr/share/man/fr/man8/update-passwd.8.gz
/usr/share/doc-base
/usr/share/doc-base/users-and-groups
/usr/share/base-passwd
/usr/share/base-passwd/passwd.master
/usr/share/base-passwd/group.master
/usr/share/lintian
/usr/share/lintian/overrides
/usr/share/lintian/overrides/base-passwd
/usr/share/doc
/usr/share/doc/base-passwd
/usr/share/doc/base-passwd/copyright
/usr/share/doc/base-passwd/users-and-groups.html
/usr/share/doc/base-passwd/changelog.gz
/usr/share/doc/base-passwd/users-and-groups.txt.gz
/usr/share/doc/base-passwd/README
$ dpkg -S /bin/date
coreutils: /bin/date
$ dpkg -s coreutils
Package: coreutils
Essential: yes
Status: install ok installed
Priority: required
Section: utils
Installed-Size: 13822
Maintainer: Michael Stone <mstone@debian.org>
Architecture: amd64
Multi-Arch: foreign
Version: 8.13-3.5
Replaces: mktemp, timeout
Depends: dpkg (>= 1.15.4) | install-info
Pre-Depends: libacl1 (>= 2.2.51-8), libattr1 (>= 1:2.4.46-8), libc6 (>= 2.7),
libselinux1 (>= 1.32)
Conflicts: timeout
Description: GNU core utilities
 This package contains the basic file, shell and text manipulation
 utilities which are expected to exist on every operating system.
 .
 Specifically, this package includes:
 arch base64 basename cat chcon chgrp chmod chown chroot cksum comm cp
 csplit cut date dd df dir dircolors dirname du echo env expand expr
 factor false flock fmt fold groups head hostid id install join link ln
 logname ls md5sum mkdir mkfifo mknod mktemp mv nice nl nohup nproc od
 paste pathchk pinky pr printenv printf ptx pwd readlink rm rmdir runcon
 sha*sum seq shred sleep sort split stat stty sum sync tac tail tee test
 timeout touch tr true truncate tsort tty uname unexpand uniq unlink
 users vdir wc who whoami yes
Homepage: http://gnu.org/software/coreutils
$ dpkg -l 'b*'
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name         Version        Architecture   Description
+++-============-==============-==============-================================
un  backupninja  <none>                        (no description available)
un  base         <none>                        (no description available)
```

```
un  base-config  <none>                        (no description available)
ii  base-files   7.1            amd64         Debian base system miscellaneous
ii  base-passwd  3.5.26         amd64         Debian base system master passwo
[...]
$ dpkg -c /var/cache/apt/archives/gnupg_1.4.12-7_amd64.deb
drwxr-xr-x root/root         0 2013-01-02 19:28 ./
drwxr-xr-x root/root         0 2013-01-02 19:28 ./usr/
drwxr-xr-x root/root         0 2013-01-02 19:28 ./usr/share/
drwxr-xr-x root/root         0 2013-01-02 19:28 ./usr/share/doc/
drwxr-xr-x root/root         0 2013-01-02 19:28 ./usr/share/doc/gnupg/
-rw-r--r-- root/root      3258 2012-01-20 10:51 ./usr/share/doc/gnupg/TODO
-rw-r--r-- root/root       308 2011-12-02 18:34 ./usr/share/doc/gnupg/FAQ
-rw-r--r-- root/root      3543 2012-02-20 18:41
./usr/share/doc/gnupg/Upgrading_From_PGP.txt
-rw-r--r-- root/root       690 2012-02-20 18:41
./usr/share/doc/gnupg/README.Debian
-rw-r--r-- root/root      1418 2012-02-20 18:41
./usr/share/doc/gnupg/TODO.Debian
[...]
$ dpkg -I /var/cache/apt/archives/gnupg_1.4.12-7_amd64.deb
 new debian package, version 2.0.
 size 1952176 bytes: control archive=3312 bytes.
    1449 bytes,    30 lines      control
    4521 bytes,    65 lines      md5sums
     479 bytes,    13 lines   *  postinst             #!/bin/sh
     473 bytes,    13 lines   *  preinst              #!/bin/sh
 Package: gnupg
 Version: 1.4.12-7
 Architecture: amd64
 Maintainer: Debian GnuPG-Maintainers <pkg-gnupg-maint@lists.alioth.debian.org>
 Installed-Size: 4627
 Depends: libbz2-1.0, libc6 (>= 2.4), libreadline6 (>= 6.0), libusb-0.1-4 (>=
2:0.1.12), zlib1g (>= 1:1.1.4), dpkg (>= 1.15.4) | install-info, gpgv
 Recommends: libldap-2.4-2 (>= 2.4.7), gnupg-curl
 Suggests: gnupg-doc, xloadimage | imagemagick | eog, libpcsclite1
 Section: utils
 Priority: important
 Multi-Arch: foreign
 Homepage: http://www.gnupg.org
 Description: GNU privacy guard - a free PGP replacement
  GnuPG is GNU's tool for secure communication and data storage.
  It can be used to encrypt data and to create digital signatures.
  It includes an advanced key management facility and is compliant
  with the proposed OpenPGP Internet standard as described in RFC 4880.
[...]
```

POUR ALLER PLUS LOIN Comparaison de versions

`dpkg` étant le programme de référence pour manipuler les paquets Debian, il fournit également l'implémentation de référence de la logique de comparaison des numéros de version. C'est pourquoi il dispose d'une option `--compare-versions` utilisable par des programmes externes (et notamment les scripts de configuration exécutés par `dpkg` lui-même). Cette option requiert trois paramètres : un numéro de version, un opérateur de comparaison et un deuxième numéro de version. Les différents opérateurs possibles sont `lt` (strictement plus petit que — *lower than*), `le` (plus petit ou égal à — *lower or equal*), `eq` (égal à — *equal*), `ne` (différent de — *not equal*), `ge` (plus grand ou égal à — *greater or equal*) et `gt` (strictement plus grand que — *greater than*). Si la comparaison est avérée, `dpkg` renvoie le code de retour 0 (succès) ; sinon il renvoie une valeur non nulle (indiquant un échec).

```
$ dpkg --compare-versions 1.2-3 gt 1.1-4
$ echo $?
0
$ dpkg --compare-versions 1.2-3 lt 1.1-4
$ echo $?
1
$ dpkg --compare-versions 2.6.0pre3-1 lt 2.6.0-1
$ echo $?
1
```

Notez l'échec inattendu de la dernière comparaison : pour `dpkg`, `pre` — dénotant généralement une pré-version — n'a pas de signification particulière et ce programme compare les caractères alphabétiques de la même manière que les chiffres (a < b < c ...), dans l'ordre dit « lexicographique ». C'est pourquoi il considère que « `0pre3` » est plus grand que « `0` ». Lorsque l'on souhaite intégrer dans le numéro de version d'un paquet qu'il s'agit d'une préversion, on fait usage du caractère « `~` » :

```
$ dpkg --compare-versions 2.6.0~pre3-1 lt 2.6.0-1
$ echo $?
0
```

5.4.4. Journal de dpkg

`dpkg` tient un journal de toutes ses actions, dans `/var/log/dpkg.log`. Ce journal est extrêmement verbeux, car il détaille chacune des étapes par lesquelles passent les paquets manipulés par `dpkg`. En plus d'offrir un moyen de suivre le comportement de `dpkg`, cela donne surtout un historique de l'évolution du système : on peut retrouver l'instant précis où chaque paquet a été installé ou mis à jour et ces informations peuvent être extrêmement utiles pour comprendre un changement récent de comportement. Par ailleurs, toutes les versions étant enregistrées, il est facile de croiser les informations avec le `changelog.Debian.gz` des paquets incriminés, voire avec les rapports de bogues disponibles en ligne.

5.4.5. Support multi-architecture

Tous les paquets Debian ont un champ `Architecture` dans leur information de contrôle. Ce champ peut prendre la valeur `all` (pour les paquets ne dépendant pas d'une architecture particulière), ou le nom de l'architecture visée dans le cas contraire (comme `amd64`, `armhf`, etc.). Dans ce dernier cas, par défaut, `dpkg` n'acceptera d'installer le paquet que si son architecture déclarée est la même que celle renvoyée par `dpkg --print-architecture`.
Cette restriction assure que les utilisateurs ne vont pas se retrouver avec des binaires compilés pour une architecture incorrecte. Elle présente tout de même le défaut que certains ordinateurs sont capables de faire fonctionner des binaires compilés pour différentes architectures, soit de manière native (un système amd64 peut exécuter des programmes i386) soit par le biais d'émulateurs.

5.4.5.1. Activer le support multi-architecture

Le support multi-architecture de `dpkg` permet à l'administrateur de définir des architectures supplémentaires dont les paquets pourront être installés sur le système. Cela se fait simplement par la commande `dpkg --add-architecture` comme l'illustre l'exemple ci-dessous. Il existe

aussi une commande `dpkg --remove-architecture` pour désactiver le support d'une architecture supplémentaire, mais elle n'est utilisable que si aucun paquet de cette architecture n'est installé.

```
# dpkg --print-architecture
amd64
# dpkg --print-foreign-architectures
# dpkg -i gcc-4.7-base_4.7.2-5_armhf.deb
dpkg: error processing gcc-4.7-base_4.7.2-5_armhf.deb (--install):
 package architecture (armhf) does not match system (amd64)
Errors were encountered while processing:
 gcc-4.7-base_4.7.2-5_armhf.deb
# dpkg --add-architecture armhf
# dpkg --add-architecture armel
# dpkg --print-foreign-architectures
armhf
armel
# dpkg -i gcc-4.7-base_4.7.2-5_armhf.deb
Selecting previously unselected package gcc-4.7-base:armhf.
(Reading database ... 97399 files and directories currently installed.)
Unpacking gcc-4.7-base:armhf (from gcc-4.7-base_4.7.2-5_armhf.deb) ...
Setting up gcc-4.7-base:armhf (4.7.2-5) ...
# dpkg --remove-architecture armhf
dpkg: error: cannot remove architecture 'armhf' currently in use by the database
# dpkg --remove-architecture armel
# dpkg --print-foreign-architectures
armhf
```

***NOTE* Le support multi-architecture dans APT**

APT détecte quand dpkg a été configuré pour reconnaître des architectures supplémentaires et télécharge automatiquement les fichiers `Packages` correspondants pendant son processus de mise à jour.
Les paquets des architectures supplémentaires peuvent alors être installés avec `apt-get install` *`package`*`:`*`architecture`*.

***EN PRATIQUE* Utilisation de binaires i386 propriétaires sur amd64**

Il existe de nombreux cas d'usage pour le support multi-architectures ; parmi les plus populaires, citons la possibilité d'exécuter des binaires 32 bits (i386) sur des systèmes 64 bits (amd64), en particulier pour tenir compte du fait que plusieurs applications populaires bien que propriétaires (comme Skype) ne sont disponibles qu'en version 32 bits.
Avant le support multi-architecture, pour faire fonctionner une application 32 bits sur un système 64 bits, il fallait installer ia32-libs. Ce paquet, qui rassemblait dans un paquet « amd64 » des versions 32 bits des bibliothèques les plus courantes, était un affreux bricolage.

5.4.5.2. Changements liés au support multi-architecture

Pour que le support multi-architecture soit réellement utile et utilisable, les bibliothèques ont dû être réempaquetées et déplacées vers un répertoire dépendant de l'architecture, de sorte que plusieurs copies de la même bibliothèque (mais compilées pour des architectures différentes) puissent être installées en même temps. Ces paquets ont été mis à jour pour inclure le champ d'en-tête `Multi-Arch: same`, qui signale au système de gestion des paquets que plusieurs versions de ces paquets peuvent être co-installées sans risque (et que ces paquets ne peuvent satisfaire que des dépendances

de paquets ayant la même architecture). Comme le support multi-architecture n'est présent que depuis Debian Wheezy, toutes les bibliothèques n'ont pas encore été converties (mais celles qui étaient rassemblées dans ia32-libs l'ont été !).

```
$ dpkg -s gcc-4.7-base
dpkg-query: error: --status needs a valid package name but 'gcc-4.7-base' is
not: ambiguous package name 'gcc-4.7-base' with more than one installed instance

Use --help for help about querying packages.
$ dpkg -s gcc-4.7-base:amd64 gcc-4.7-base:armhf | grep ^Multi
Multi-Arch: same
Multi-Arch: same
$ dpkg -L libgcc1:amd64 |grep .so
/lib/x86_64-linux-gnu/libgcc_s.so.1
$ dpkg -S /usr/share/doc/gcc-4.7-base/copyright
gcc-4.7-base:armhf, gcc-4.7-base:amd64: /usr/share/doc/gcc-4.7-base/copyright
```

Il est à noter que les paquets `Multi-Arch: same` ne sont identifiables sans ambiguïté que si leur nom est qualifié avec leur architecture. Ils ont également la possibilité de partager des fichiers avec d'autres instances du même paquet ; `dpkg` s'assure que ces fichiers partagés sont identiques au bit près. Pour terminer, mentionnons que toutes les instances d'un même paquet doivent avoir la même version et qu'ils doivent donc être mis à jour en même temps.
Le support multi-architecture apporte également quelques complications dans la gestion des dépendances. Une dépendance, pour être satisfaite, requiert soit un paquet marqué `Multi-Arch: foreign`, soit un paquet dont l'architecture est identique à celle du paquet déclarant la dépendance (lors de ce processus de résolution des dépendances, les paquets indépendants de l'architecture sont considérés comme ayant l'architecture principale du système). Une dépendance peut aussi être affaiblie de manière à pouvoir être satisfaite par un paquet d'architecture quelconque, avec la syntaxe *`paquet`*`:any`, mais les paquets des architectures supplémentaires ne peuvent satisfaire cette dépendance que s'ils sont marqués comme `Multi-Arch: allowed`.