## I. Quelques distributions

### a. Debian

Le projet Debian a été fondé en 1993 par Ian Murdock à une époque où l'idée même de distribution Linux en était encore à ses balbutiements. Le nom Debian provient de Debra (la femme de Murdock) et Ian. Debian a longtemps été la seule distribution entièrement et uniquement composée de logiciels libres et Open Source ce qui lui vaut toujours le nom officiel de Debian GNU/Linux. Debian a aussi été supporté quelques temps officiellement par la FSF comme distribution Linux de référence. Les avantages de Debian sont nombreux :

- Un nombre gigantesque de packages qui se chiffre en milliers,
- Un logiciel d'installation appelé APT très pratique et performant,
- Une distribution 100% open source,
- Une stabilité à toute épreuve pour un environnement de production.

Ces avantages entraînent aussi quelques inconvénients :

- Des packages souvent anciens,
- Des mises à jour de la distribution irrégulières et trop espacées,
- Des risques liés à la multiplication des paquets et des dépendances,
- Une installation et une configuration compliquées.

Tous ces inconvénients ne sont pas forcément des défauts. Faut-il préférer une version ancienne mais totalement exempte de bugs ou la toute dernière version d'un produit dont la fiabilité n'a pas été pleinement éprouvée ?
Tous ces éléments font de Debian une distribution idéale pour les informaticiens, les ingénieurs et administrateurs système et réseau, les environnements de production en entreprise…

### b. Ubuntu

Le milliardaire sud-africain Mark Shuttleworth, principalement connu du monde entier pour avoir été l'un des premiers touristes de l'espace, mais aussi des informaticiens pour avoir fait fortune en revendant sa société Thawte spécialisée dans la sécurité à Verisign, est un vrai informaticien qui a contribué au projet Debian. Devant les quelques inconvénients de la distribution il crée la distribution Ubuntu Linux en 2005 avec un budget initial de 10 millions de dollars pour rémunérer les développeurs. Le mot Ubuntu est un mot du langage africain bantou signifiant « humanité aux autres » ou encore « je suis ce que je suis grâce à ce que nous sommes tous ». Cette définition reflète ce qu'est la distribution : un dérivé de Debian dont le but est de fournir des logiciels plus récents et très fortement axés sur la convivialité et l'ergonomie à l'aide du support du plus grand nombre :

- Une distribution issue de Debian,
- Une compatibilité avec les packages de Debian,
- Un système d'installation très simple,
- Une sortie tous les 6 à 8 mois,
- Un environnement graphique agréable.

Cette distribution est idéale pour les étudiants, cependant la tentation est très forte de revenir au fonctionnement d'une distribution Debian, les deux étant compatibles.

### c. Red Hat et Fedora

S'il y a bien une société commerciale dans le monde Linux qui a marqué et qui continue à marquer son époque, c'est bien la société Red Hat. Fondée en 1995 par Robert Young et Marc Ewing, elle édite la célèbre distribution éponyme dont la première version officielle date de 1994 (la société a été fondée après la sortie de la distribution). Le système de package RPM est apparu avec la version 2.0. Les distributions Red Hat ont très fortement marqué les esprits car elles sont restées la référence pendant presque dix ans. Chaque version était innovante tant dans l'intégration des logiciels que dans son installateur (appelé anaconda) et ses outils de configuration.
Cependant en 2003 la version 9.0 est la dernière destinée officiellement au grand public. Les versions suivantes ont été confiées au projet communautaire **Fedora** qui continue tous les six mois à sortir une nouvelle version. Red Hat se concentre maintenant sur le monde de l'entreprise avec des distributions commerciales appelées **RHEL** (*Red Hat Enterprise Linux*) :

- Des versions professionnelles destinées aux entreprises,
- Des solutions du poste de travail au plus gros serveur,
- Des architectures matérielles nombreuses,
- Un support commercial,
- Des mises à jour assurées pendant sept ans,
- 100% libre.

## II. Obtenir de l'aire sur Linux

### 1. L'aide propre aux commandes

Il n'est pas possible de connaître par coeur tous les paramètres et arguments d'une commande. Linux propose heureusement au moins deux mécanismes pour connaître ceux qui sont supportés par une commande. La plupart du temps, le paramètre --help affiche l'aide incluse directement au sein du programme appelé. Parfois l'aide apportée est suffisante pour trouver ce que vous cherchez.

```
$ date --help
```

Il peut cependant arriver que l'aide soit trop concise ou manque d'explications, ou bien soit totalement absente. Dans ce cas --help est considéré comme un paramètre invalide et vous risquez d'obtenir un message d'erreur et/ou une ligne d'informations :

```
$ cal --help
```

### 2. L'aide interne au shell

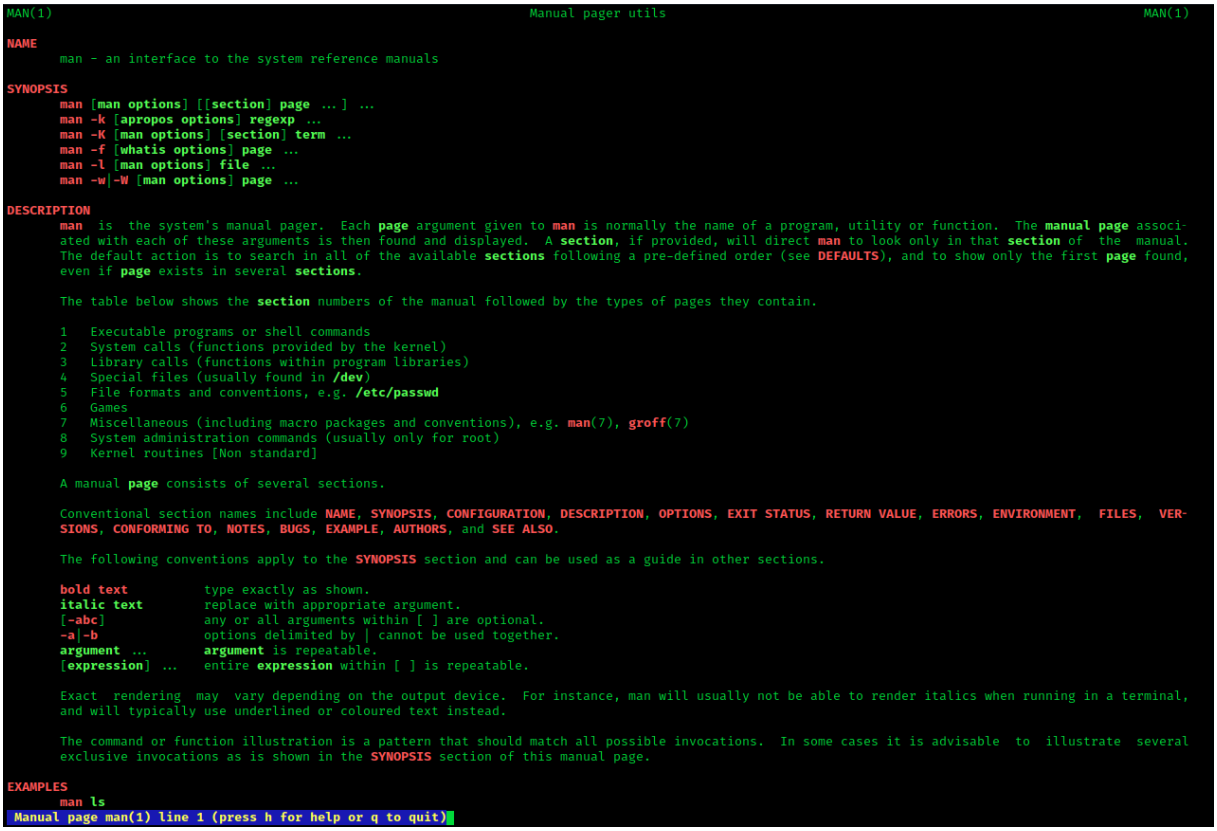
Les commandes internes n'acceptent pas de paramètre --help, mais pour ces commandes l'interpréteur de commandes propose une commande **help**. Utilisée seule elle vous fournit la liste des commandes internes. Si vous lui passez comme paramètre le nom d'une commande interne, l'aide de celle-ci est affichée. C'est ainsi que vous pouvez apprendre que pwd admet deux paramètres optionnels.

```
$ help pwd
```

### 3. Obtenir plus d'aide avec la commande man

### a. Accès

Quand les deux mécanismes d'aide précédents se révèlent être insuffisants, il est très probable que l'aide recherchée se situe au sein du manuel Unix. Ce manuel est standard sur tous les Unix dont Linux, et quel que soit le shell puisqu'il s'agit d'une commande externe.
Le manuel est accessible depuis la commande **man**.




Une page du manuel est composée de plusieurs sections dont celles-ci, sachant qu'elles ne sont pas forcément toutes présentes :

- **Nom** : nom et rôle de la commande.
- **Synopsis** : syntaxe générale, paramètres et arguments acceptés.
- **Description** : mode d'emploi détaillé du fonctionnement de la commande et des arguments principaux.
- **Options** : description détaillée de chaque paramètre possible, généralement sous forme de liste.
- **Exemples** : le manuel peut fournir des exemples concrets d'utilisation de la commande.
- **Environnement** : le fonctionnement de la commande peut réagir différemment si des variables du shell sont positionnées à certaines valeurs.
- **Conformité** : la commande est conforme à des recommandations ou normes (par exemple POSIX ).
- **Bogues** : la commande peut parfois rencontrer des dysfonctionnements dans des cas ponctuels qui peuvent être énumérés à cet endroit.

- **Diagnostics/retour** : la commande, selon son résultat, peut retourner des codes d'erreurs significatifs dont la valeur permet de déterminer le type de problème (fichier en argument absent, etc.)
- **Voir aussi** : • liste des commandes liées au programme qui peuvent intéresser l'utilisateur.

Le manuel Linux ne fait pas que référencer les commandes classiques. C'est un manuel bien plus complet que ça.
Les commandes simples, celles d'administration, les fichiers de configuration, les périphériques, les appels systèmes, les fonctions de programmation de divers langages, et bien d'autres choses encore, peuvent y être référencés. C'est pourquoi le manuel est composé de plusieurs sections distinctes.

## III. Les gestionnaires de paquets

### A. Red Hat Package Manager

#### a. Notion de package

Contrairement à d'autres systèmes d'exploitation, il n'est pas courant sur Linux et Unix en général de disposer de logiciels fournis avec un programme l'installation interactif (pas de install.exe). Certains éditeurs proposent des scripts d'installation et bien souvent ceux-ci se contentent de décompresser et de désarchiver quelques fichiers.

Avec Linux, il est très classique de disposer des divers produits, outils, mises à jour, etc. sous forme de paquetages (packages). Un package est un fichier (parfois gros) qui contient le produit à installer et des règles. Ces règles peuvent être multiples :

- Gestion des dépendances : le produit ne pourra être installé que si les produits qu'il utilise lui-même sont déjà présents.
- Pré-installation : des actions sont à prévoir avant de pouvoir installer le produit (changer des droits, créer des répertoires, etc.).
- Post-installation : des actions sont à prévoir après l'installation du produit (paramétrage d'un fichier de configuration, compilation annexe, etc.).

Sur Red Hat, Fedora, SuSE, Mandriva et quelques autres distributions le format de package par défaut est le **RPM** (*Red Hat Package Manager*). Sous Debian, Knoppix, Kaella, Ubuntu, c'est le format **DPKG** (*Debian Package*). Outre le format, ce sont surtout les outils qui les différencient.
Le fait de disposer des informations de dépendances permet d'obtenir des outils performants qui peuvent seuls les résoudre en cascade. En installant un package, l'outil pourra installer toutes les dépendances nécessaires. On peut parfois spécifier plusieurs emplacements (repositories) pour ces packages, soit locaux (disque dur, CD-Rom, DVD, etc.) soit distants (http, ftp, etc.).
Il faut toujours utiliser un package prévu pour sa distribution quand il existe. Si ce n'est pas le cas, il est parfois possible d'utiliser un package d'un produit concurrent ou de recompiler le produit soi-même.

#### b. Le gestionnaire RPM

**RPM** est un gestionnaire de packages inventé par Red Hat puis utilisé massivement par de nombreuses autres distributions. Il simplifie fortement la distribution, l'installation, la mise à

jour et la suppression des logiciels. Il se base sur des commandes (ex : **rpm**), une base de données locale et des packages au format rpm (extension rpm).
La base de données est située dans **/var/lib/rpm**. Toutes les informations concernant les logiciels installés, leurs versions, leurs fichiers et droits, et leurs dépendances y sont précisées. Sauf gros problème, il ne faut JAMAIS modifier cette base à la main. Il faut utiliser les outils RPM. Chaque logiciel est fourni sous forme de package au format RPM. Le rpm répond à une nomenclature précise. **nom-version-edition.architecture.rpm**
par exemple : **php-4.1.2-2.1.8.i586.rpm**

L'édition est un identifiant de version du package RPM propre à l'éditeur. Ici c'est la version **2.1.8** du package **PHP** version **4.1.2.** L'architecture est **i586** (Intel Pentium). On peut aussi trouver i386, i686, x86_64 (64 bits), ppc64, s390x ou noarch. Un package noarch ne contient pas de programmes ou bibliothèques binaires mais du code indépendant comme des scripts, de la documentation, des images, du son, de la vidéo, etc.

**c. Installation, mise à jour et suppression**
Vous installez un package rpm avec le paramètre -i.
**rpm -i php-4.1.2 - 2.1.8.i586.rpm**

Comme il est possible d'utiliser des caractères de substitution (**rpm -i *.rpm**), vous pouvez afficher le nom du package en cours d'installation avec le paramètre -v. Le paramètre -h affiche des caractères # pour indiquer la progression de l'installation. L'installation ne fonctionnera pas si les dépendances ne sont pas résolues.
La mise à jour d'un produit vers une version supérieure depuis un package se fait avec le paramètre -U. Dans ce cas tous les fichiers sont mis à jour par ceux de la nouvelle version : les anciens sont supprimés et remplacés par les nouveaux. Les anciens fichiers de configuration sont sauvés avec l'extension **.rpmsave**. Si le package n'était pas installé, la mise à jour joue le rôle d'installation. Attention avec ce paramètre : il installe le package même si une version précédente n'était pas installée.
**rpm -Uvh php-4.1.3-1.i586.rpm**

La mise à jour est aussi possible avec -F. Mais si le package n'était pas installé, il ne le sera pas non plus lors de la mise à jour contrairement à -U. Ainsi, si vous disposez de tous les packages de mise à jour du système et que vous ne souhaitez mettre à jour que ceux qui sont réellement installés, alors vous pouvez taper : **rpm -Fvh *.rpm**

La suppression s'effectue avec le paramètre -e. Attention cependant, c'est le nom du package installé qui doit être passé en paramètre et pas le nom du fichier de package.
rpm -e php

Plusieurs options supplémentaires sont possibles :

- **--force** : en cas de conflit avec un autre package (le cas le plus courant est celui où deux packages proposent le même fichier au même endroit), cette option force tout de même l'installation.
- **--nodeps** : si le package refuse de s'installer à cause d'un problème de dépendances, cette option forcera l'installation. Il arrive parfois que cette erreur se produise quand la dépendance en question a été installée autrement que depuis un package rpm (ex : compilation, binaire copié à la main).

**d. Requêtes RPM**

La base de données RPM peut être interrogée facilement avec le paramètre -q suivi de plusieurs options.

- **-a** : liste de tous les packages installés.
- **-i** : informations générales (le résumé) du package.
- **-l** : liste des fichiers installés.
- **-f nom** : trouve le package qui contient le fichier donné.
- **-p nom** : la recherche s'effectue dans le fichier de package donné.
- **--requires** : dépendances du package.
- **--provides** : ce que fournit le package.
- **--scripts** : scripts exécutés à l'installation et la suppression.
- **--changelog** : l'historique du package.

Exemple**: $ rpm -qilp libjpeg-6.2.0-738.i586.rpm**

**e. Vérification des packages**

Il est possible qu'après l'installation d'un package, un ou plusieurs des fichiers installés aient été altérés (changement de droit, de propriétaire, édition, suppression, etc.). Comme la base RPM contient toutes les informations nécessaires, on peut demander une vérification avec le paramètre -V.

**$ rpm -V php**

S.5....T c /etc/php.ini

Un point signifie qu'une étape de vérification est OK. Sinon :

- **S** : la taille du fichier a été modifiée.
- **5** : la somme MD5 ne correspond plus.
- **T** : la date de modification n'est plus la même.
- **U** : le propriétaire a été modifié.
- **G** : le groupe a été modifié.
- **L** : le lien symbolique a été modifié.
- **M** : les permissions ou le type du fichier ont été modifiés.
- **D** : le périphérique a été modifié (major/minor).

Le **c** indique qu'il s'agit d'un fichier de configuration.

Les fichiers de packages RPM sont très souvent signés par l'éditeur de la distribution de manière à en garantir l'intégrité. On peut vérifier l'intégrité d'un package avec une clé publique GPG mais il faut par avance avoir déjà chargé cette clé publique sur le système.

**gpg --import RPM-GPG-KEY**
**rpm --import RPM-GPG-KEY**
**rpm --checksig libjpeg-6.2.0-738.i586.rpm**

**B. YUM : Nouvellement dnf**

YUM est aux fichiers rpm ce que APT est aux fichiers dpkg : un logiciel de gestion de packages. Il récupère les packages au sein de dépôts et gère les dépendances à votre place. YUM signifie *Yellow dog Updater Modified*. Il est principalement utilisé sur les distributions Redhat (les version Entreprise) et Fedora, mais peut être utilisé sur n'importe quelle distribution de type RPM, si les dépôts associés le supportent.

Les commandes et exemples suivants se basent sur un serveur Redhat Enterprise Linux 5. Le fichier de configuration est **/etc/yum.conf (/etc/dnf/dnf. conf)**.

## 1. Configuration des dépôts

Les dépôts sont placés soit dans le fichier de configuration principal, soit dans le répertoire **/etc/yum.repos.d**.

## 2. Utilisation des dépôts

### a. Lister les packages

Le paramètre list permet de lister les packages. Tous sont listés par défaut. Vous pouvez préciser une liste de packages, ou fournir des caractères jokers. Plusieurs options sont disponibles :

- **all** : c'est le cas par défaut : les packages installés sont listés en premier, puis les packages disponibles pour installation.
- **available** : les packages disponibles pour installation.
- **updates** : les packages pouvant être mis à jour.
- **installed** : les packages mis à jour.
- **obsoletes** : les packages du systèmes rendus obsolètes par des versions supérieures disponibles.
- **recent** : les derniers packages ajoutés dans les dépôts.

La commande suivante liste les noyaux disponibles :
**# yum list available kernel***

### b. Installer des packages

Passez le paramètre install, suivi des noms des packages à installer.
Voici un exemple d'installation du package mc :
**#yum install ssh**

### c. Mises à jour

Vérifiez la présence de mises à jour avec le paramètre **check-update** :
**# yum check-update**

Vous avez deux possibilités pour installer les mises à jour :

- **update** : mise à jour d'un package ou de tous si aucun package n'est précisé.
- **upgrade** : mise à niveau complète de la distribution : les packages vus comme obsolètes sont remplacés par ceux de la dernière version disponible.

### d. Rechercher un package

Utilisez le paramètre search, suivi du ou des packages à rechercher dans les dépôts. Les caractères jockers sont autorisés.
**# yum search tomcat**

### e. Supprimer un package

Pour supprimer un package, utilisez le paramètre remove :
**# yum remove mc**

## C. Debian Package

### 1. dpkg : le gestionnaire de paquets Debian

La commande **dpkg** est le pendant de rpm pour les distributions Debian et dérivées, dont Ubuntu. Elle fait la même chose, ou presque, que rpm. Les packages Debian portent une extension .deb pour les reconnaître et disposent des mêmes informations et moyens qu'un package rpm. La commande **dpkg** est chargée de l'installation, la création, la suppression et la gestion des paquets Debian.
La base de données dpkg est généralement placée dans **/var/lib/dpkg**. Les fichiers qui y sont présents sont au format texte. Cependant n'éditez pas les fichiers à la main. Le fichier **/var/lib/dpkg/status** contient l'intégralité des packages connus par dpkg avec leur état.

Dpkg dispose d'une interface graphique, GDebi, qui permet d'éviter l'utilisation de la ligne de commande.

### 2. Installation, mise à jour et suppression

L'option **-i**, ou **-install**, installe le ou les packages passés comme argument.
**# dpkg -i monpaquet.deb**

Notez que comme rpm, dpkg ne gère pas seul les dépendances. S'il manque des dépendances, la commande vous en informera. Dans ce cas, vous devez installer les dépendances de la même manière avant d'installer votre package.
Vous pouvez demander l'installation de tous les packages présents au sein d'une arborescence avec le paramètre -R, pour récursif. Dans ce cas, indiquez comme argument un nom de répertoires : tous les packages présents dans le répertoire et ses sous-répertoires seront installés.
**# dpkg -R folder**

La mise à jour s'effectue de la même manière que l'installation, avec le **-i**. Si vous installez un package déjà présent, dpkg en effectue une mise à jour. Ainsi, une installation ou une mise à jour respectent la méthodologie suivante :

- Extraction des fichiers de contrôle du nouveau paquet.
- Quand une ancienne version du même paquet est déjà installée, exécution du script pré supression de l'ancien paquet.
- Lancement du script de préinstallation s'il est fourni par le paquet.
- Dépaquetage des nouveaux fichiers et sauvegarde des anciens pour pouvoir les restaurer en cas de problème.
- Si une ancienne version du paquet est déjà installée, exécution du script de post suppression de l'ancien paquet.
- Configuration du paquet.

Il n'existe pas d'équivalence au mode freshen (-F) de rpm. Si vous souhaitez mettre à jour un package uniquement s'il est déjà installé, vous devez tout d'abord vérifier s'il est installé. Si un package est installé il commence par ii dans la liste :
**# (dpkg -l zip | grep ^ii >/dev/null) && echo PRESENT || echo ABSENT**
PRESENT

**# (dpkg -l slapd | grep ^ii >/dev/null) && echo PRESENT || echo ABSENT**
ABSENT

Pour mettre à jour un paquet seulement s'il est présent utilisez ce genre de ligne de commande, ou un équivalent de votre cru :
**# (dpkg -l zip | grep ^ii >/dev/null) && dpkg -l zip.deb**

Notez que le plus simple pour mettre à jour vos paquets déjà présents est de créer un dépôt APT contenant vos mises à jour et d'exécuter un apt-get upgrade depuis le client.

La suppression d'un package s'effectue avec le paramètre -r (en minuscule). Là encore, c'est à vous de gérer les dépendances.
La suppression d'un paquet effectue les étapes suivantes :

- Exécution du script de pré-suppression.
- Suppression des fichiers installés.
- Exécution du script de post-suppression.

**# dpkg -r zip**

Tout est supprimé sauf les fichiers de configuration et ce, afin d'éviter une reconfiguration de l'outil si vous le réinstallez. Pour tout supprimer, y compris ces fichiers, précisez le paramètre -P (purge).
**# dpkg -P apache**

Si vous remplacez le nom du package par les paramètres -a ou --pending les packages non installés (non dépaquetés) mais présents dans les informations de la base pour être purgés ou supprimés, sont effacés.
L'utilisation des options --force-all et --purge permet de forcer la désinstallation du paquet et de supprimer les fichiers de configuration associés.
**# dpkg --force-all --purge nom_du_paquet**

**3. Requêtes dpkg**
**a. Lister les paquets**
Vous pouvez lister tous les packages Debian connus du système avec le paramètre -l :
**# dpkg -l**

Vous pouvez indiquer un motif particulier :
**# dpkg -l "apt*" |grep ^ii**

**b. Trouver un paquet contenant un fichier**
Le paramètre -S suivi du nom d'un fichier (son chemin) permet de retrouver le paquet d'origine.
**# dpkg -S /usr/bin/basename**

**c. Lister le contenu d'un paquet**
Le paramètre -L liste le contenu du ou des paquets indiqués :
**# dpkg -L coreutils**

**4. Convertir des packages**
L'outil **alien** permet de convertir des packages RPM en DPKG et vice versa. Certains packages ne sont fournis que pour l'un ou l'autre des systèmes. C'est embêtant lorsqu'un produit n'est fourni que sous une forme et qu'il faut tout de même l'installer sur une autre plate-forme Linux. Voici l'exemple d'un package, le client Networker, uniquement fourni pour Red Hat. Avec Alien, il est possible de le convertir au format dpkg.
Le paramètre par défaut -d convertit du rpm au dpkg :

**# alien -d lgtoclnt-7.4-1.i686.rpm**

Comme indiqué, la conversion par défaut va vérifier les dépendances, mais ne va pas inclure les scripts de préinstallation et de post-installation. Vous devez alors préciser le paramètre --scripts.
**# alien --scripts -d lgtoclnt-7.4-1.i686.rpm**

### 5. L'outil dselect

L'outil **dselect** est un frontend (comme APT) pour dpkg, qui gère les dépendances et les conflits. Historiquement, il est le premier. Cependant son remplaçant APT dispose d'une bien meilleure qualité.
Le manuel de dselect indique clairement que l'outil est aujourd'hui tombé en désuétude.

- L'interface est confuse.
- Il y a peu de maintenance.
- Le manuel est incomplet.
- Il n'y a pas de filtre.
- Les accès ne sont plus standards.

Pour toutes ces raisons, il est préférable d'utiliser APT.

## D. Gestionnaire APT

### 1. Principe

Que ce soit avec rpm ou dpkg le problème est le même : ces deux outils contrôlent les dépendances des packages pour autoriser ou non leur installation, mais ne les gèrent pas. Autrement dit, si une dépendance sur un package est absente, il ne sera pas installé, sauf si la dépendance est résolue :

- soit en installant auparavant les packages manquants,
- soit en indiquant sur la même ligne le chemin de ces mêmes packages.

De même lors d'une mise à niveau il se pose un problème avec les fichiers de configuration. Que faut-il en faire ?
APT permet de résoudre ces problèmes en gérant les dépendances à votre place. **APT** signifie *Advanced Packaging Tool*.
Au lieu de spécifier un paquet (local ou distant), il prend en charge des dépôts de packages situés sur un CD, un DVD, dans un répertoire local, sur une source distante sur Internet (ftp, http), etc.
Un dépôt contient un ensemble de packages qui dépendent soit les uns des autres, soit d'autres packages en provenance d'autres dépôts. APT peut gérer plusieurs dépôts, à divers endroits. Il se débrouille seul : lorsque vous installez un package, il installe aussi ses dépendances (s'il les trouve).

### 2. Les dépôts

#### a. Configuration

Les dépôts sont indiqués dans le fichier **/etc/apt/sources.list**. Le fichier suivant provient d'une installation Debian
**$ cat /etc/apt/sources.list**

Les dépôts sont préparés côté serveur dans une arborescence de répertoires. La commande **genbasedir** permet de créer un dépôt.

**b. Mise à jour de la base**
Une fois vos dépôts configurés, vous devez mettre à jour la base de données locale de APT avec la commande **aptget** et l'option **update**.
**# apt-get update**

**3. Mise à jour de la distribution**
Une fois les dépôts à jour, vous pouvez mettre à jour en une seule commande tous les packages installés sur votre distribution : APT vérifie si des packages plus récents sont disponibles dans les dépôts. Il se base pour cela sur la base de données locale. Si elle n'est pas à jour il est possible que certains des packages trop anciens ne soient plus présents.
Exécutez la commande **apt-get** avec l'option **upgrade**. APT vous informe que huit packages peuvent être mis à jour.
Vous pouvez accepter ou refuser. Si vous acceptez, APT télécharge ces packages et leurs éventuelles dépendances, et les installe. Le processus peut être plus ou moins long selon le nombre de mises à jour et le type de support.
**# apt-get upgrade**

Une autre possibilité est de faire une mise à jour profonde (appelée mise à jour distante). APT garde une certaine cohérence dans les packages lors de la mise à jour, notamment concernant la version de la distribution. Vous pouvez spécifier plusieurs distributions Debian dans vos dépôts. Mais même si une distribution est plus récente, un simple upgrade ne va pas transformer la vôtre en la toute dernière. Vous pouvez demander à APT de forcer la mise à jour vers la nouvelle distribution avec un **dist-upgrade**.
**$ apt-get dist-upgrade**

**4. Rechercher et installer un package individuel**
La commande **apt-cache** permet de rechercher un package, par son nom ou son commentaire, au sein de la base de données locale APT.
**# apt-cache search torrent**

La commande **apt-get install** xxx installe le package xxx :
**# apt-get install vim-gtk**

**5. Client graphique**
L'outil synaptic est un front-end : une interface graphique qui fait appel aux fonctions de APT. Il permet toutes les opérations proposées par APT tout en étant très convivial.

**6. Installer depuis les sources**
**1. Obtenir les sources**
Il n'est parfois pas possible d'obtenir un logiciel ou une bibliothèque depuis un package pour sa distribution. Dans ce cas, il reste la solution de compiler et d'installer soi-même le produit depuis les sources.
Cela est possible pour une majorité de produits sous Linux, grâce aux avantages des logiciels libres et de la licence GPL telle que définie au premier chapitre. Tout logiciel libre est fourni avec ses sources. Il est donc possible de reconstruire soi-même le logiciel en le recompilant.

**F. Remarques importantes**
**1. Quelques dates à retenir**
• Linux est issu d'UNIX, le système d'exploitation leader dans les années 1970, 1980 et 1990.
• Il s'inscrit dans le mouvement lancé par la Free Software Foundation (FSF).

• GNU est la branche de la FSF qui fournit des outils libres pour UNIX.
• Dans les années 1980, de nombreux outils étaient déjà disponibles.
• En 1992, Linus Torvalds a ajouté le noyau Linux à GNU.
• En combinant le noyau Linux avec d'autres outils GNU, il est devenu possible d'installer Linux comme système d'exploitation.
• En 1995, le premier salon et série de conférences dédiés à Linux a été lancé.
• En 2000, Steve Jobs a fait une offre que Linus a refusée.

**2. Free Software Foundation**

La FSF (Free Software Foundation) est une organisation à but non lucratif fondée en 1985 par Richard Stallman, dans le but de promouvoir la liberté du logiciel et de soutenir le mouvement du logiciel libre.
La FSF défend les principes du logiciel libre, fondés sur les libertés essentielles dont doivent bénéficier les utilisateurs pour utiliser, étudier, modifier et redistribuer les logiciels. L'organisation a joué un rôle crucial dans la définition du concept de « logiciel libre » et dans la création de la Licence Publique Générale GNU (GPL), l'une des licences les plus utilisées au sein de la communauté du logiciel libre.
Les principaux objectifs de la FSF sont :
1. Promouvoir l'idée du logiciel libre : sensibiliser à l'importance des libertés offertes par le logiciel libre.
2. Protéger ces libertés : défendre les utilisateurs contre les restrictions telles que celles imposées par les licences propriétaires.
3. Soutenir les projets de logiciel libre : apporter un soutien financier, juridique et technique aux projets qui adhèrent à ces principes.
La FSF milite pour un monde où tous les logiciels sont libres, permettant aux utilisateurs de mieux contrôler leur technologie et de protéger leur vie privée.

**3. Libertés du logiciel**

Les libertés du logiciel, souvent évoquées dans le contexte des logiciels libres et open source (FOSS), sont généralement définies comme les quatre libertés essentielles suivantes :
1. Liberté d'exécuter le programme : L'utilisateur est libre d'exécuter le logiciel à toutes fins, sans aucune restriction.
2. Liberté d'étudier et de modifier le programme : L'utilisateur peut accéder au code source, étudier son fonctionnement et le modifier selon ses besoins.
3. Liberté de distribuer des copies : L'utilisateur peut partager le logiciel avec d'autres, garantissant ainsi sa libre utilisation et sa libre diffusion.
4. Liberté de distribuer des versions modifiées : L'utilisateur peut distribuer des versions modifiées du logiciel, garantissant ainsi que les améliorations ou personnalisations puissent également être partagées et appréciées par d'autres.
Ces libertés garantissent que le logiciel peut être librement utilisé, étudié, amélioré et partagé, favorisant ainsi la collaboration et l'innovation.