Cahiers de l'Admin

Collection dirigée par Nat Makarévitch

Debian GNU/Linux

Raphaël Hertzog

Avec la contribution
de Christophe Le Bars et Roland Mas



EYROLLES

2e édition

Cahiers de l'Admin

Debian GNU/Linux

2e édition

Debian GNU/Linux, distribution Linux non commerciale extrêmement populaire, est réputée pour sa fiabilité et sa richesse. Soutenue par un impressionnant réseau de développeurs dans le monde, elle a pour mots d'ordre l'engagement vis-à-vis de ses utilisateurs et la qualité.

Cette 2e édition du cahier de l'Admin Debian détaille la plus récente version de **Debian Sarge 3.1.** Elle traite des outils et méthodes qu'un administrateur Linux doit maîtriser, depuis l'installation et la mise à jour du système jusqu'à la création de paquetages, en passant par la supervision, la sauvegarde, et les migrations. Elle aborde également de nouveaux thèmes tels que la compilation d'un noyau Linux avec les outils Debian, et fournit un glossaire des principaux termes Debian.

Raphaël Hertzog est ingénieur en informatique diplômé de l'INSA de Lyon et développeur Debian depuis 1997. Fondateur de Freexian, première SSII spécialisée dans Debian GNU/Linux, il est l'un des contributeurs français majeurs participant à ce projet Linux.

Premier développeur français de la distribution Debian GNU/Linux. **Christophe Le Bars** est expert en sécurité et migration vers les logiciels libres.

Développeur Debian depuis 5 ans, développeur et mainteneur du logiciel libre Gforge, **Roland Mas** est consultant indépendant spécialisé dans l'installation et la migration de systèmes Debian GNU/Linux.

Configuration requise :
• PC, processeur AMD ou Intel, famille x86 • 128 Mo de mémoire RAM, 64 Mo requis pour le programme d'installation • 500 Mo d'espace disponible sur le disque dur • Lecteur CD-Rom 24X ou davantage

Connexion Internet haut débit recommandée (mais non nécessaire).

Debian, système d'exploitation universel • Les principes du logiciel libre selon Debian • Développeurs Debian, utilisateurs, équipes et sous-projets • Rôle d'une distribution • **Présentation de l'étude de cas** • Pourquoi Debian GNU/Linux ? Pourquoi Debian Sarge ? **Prise en compte de l'existant et méthode de migration** • Coexistence en environnement hétérogène • Installer et configurer les services • Installation • **Système de paquetage, outils et principes de base** • Concurrents du format .deb • Paquet source • Découverte de dpkg • **Maintenance et mise à jour avec les outils APT** • apt-get et apt-cache • Frontaux : aptitude, synaptic, gnome-apt • **Se documenter** • **Résolution de problèmes** • **Configuration de base** • Configurer le clavier • Configurer le réseau • Ethernet et PPP • Nommage et résolution de noms • Utilisateurs et groupes avec ou sans LDAP • Impression • Chargeur de démarrage • LILO et GRUB • Rotation des fichiers de logs • Synchronisation horaire • Partage des droits d'administration • Points de montage • Quotas • **Supervision** • **Sauvegarde** • Hotplug • **Gestion de l'énergie** • APM, ACPI, PCMCIA • **Configuration et installation d'un noyau** • **Services Unix** • Démarrage • Connexion à distance • SSH • VNC • Webmin • Debconf • Syslog • Inetd • Cron et atd • Anacron • **Infrastructure réseau** • Passerelle • Masquerading • Filtre de paquets • VPN • QoS • DNS • IDS • **Services réseau** • Postfix • Apache • NFS • Samba • Squid • LDAP • **Station de travail** • XFree86 • L'interface graphique • GNOME et KDE • Courrier électronique, navigateurs web, développement, travail collaboratif, suites bureautiques, émulation Windows • **Techniques avancées** • Recompiler un paquet depuis ses sources • Récupérer les sources • Lancer une recompilation • Construire un paquet • Méta-paquet ou faux-paquet • Devenir mainteneur de paquet • **Distributions dérivées** • **Glossaire.**

EYROLLES

Raphaël **Hertzog**

Cahiers de l'Admin

Debian

2e édition

Collection dirigée par Nat **Makarévitch**

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

EYROLLES

ÉDITIONS EYROLLES
61, bd Saint-Germain
75240 Paris Cedex 05
www.editions-eyrolles.com

Remerciements à Thierry Stempfel pour les illustrations.



Le code de la propriété intellectuelle du 1er juillet 1992 interdit en effet expressément la photocopie à usage collectif sans autorisation des ayants droit. Or, cette pratique s'est généralisée notamment dans les établissements d'enseignement, provoquant une baisse brutale des achats de livres, au point que la possibilité même pour les auteurs de créer des œuvres nouvelles et de les faire éditer correctement est aujourd'hui menacée.
En application de la loi du 11 mars 1957, il est interdit de reproduire intégralement ou partiellement le présent ouvrage, sur quelque support que ce soit, sans autorisation de l'éditeur ou du Centre Français d'Exploitation du Droit de Copie, 20, rue des Grands-Augustins, 75006 Paris.
© Groupe Eyrolles, 2004, 2005, ISBN : 2-212-11639-X

Préface

Les professionnels découvrent enfin le projet Debian, dont le souci de réaliser un ensemble riche, souple et requérant peu d'attention correspond bien à leurs attentes. Ils apprécient le soin apporté à la robustesse-fiabilité, à l'automatisation des tâches subalternes ainsi qu'à la mise au point et au respect de spécifications garantes de la cohérence, donc de la pérennité des savoirs.

Dans le même temps, de grands acteurs de l'informatique perçoivent vraisemblablement aujourd'hui l'intérêt stratégique d'une distribution Linux mûre et non gérée par une entité commerciale. Certains de leurs clients comprennent, dans le même registre, qu'une plate-forme logicielle ne dépendant pas d'accords tissés entre des fournisseurs réduit les contraintes pesant sur eux après l'achat.

De nombreux amateurs, enfin, découvrent Debian grâce aux évolutions spécifiques de la famille Knoppix et certains, souhaitant fuir l'empirisme, « ouvrent le capot ».

Debian GNU/Linux, longtemps discrète, convainquit tout d'abord le passionné, souvent attiré par l'esprit qui l'anime. Il y trouve un projet aux objectifs clairs et aux réalisations transparentes, au sein duquel tous œuvrent afin de bien concevoir *avant* de construire — renonçant d'emblée aux échéances, donc à leurs contraintes menaçant la qualité de tant d'autres logiciels. Il y trouve un projet dirigé par ses acteurs. Il y adopte ou rejoint, en somme, un projet bénéficiant pleinement des avantages du logiciel libre... afin d'en produire.

Ce *Cahier de l'Admin* guidera et éclairera le lecteur afin de le rendre autonome. Seul pouvait le rédiger un tandem auteur-relecteur maîtrisant les aspects techniques tout autant que les caractéristiques propres du projet Debian, et connaissant parfaitement les besoins des francophones, professionnels aguerris comme amateurs

éclairés. Raphaël et Christophe disposaient des qualités requises et surent, aidés par Roland, créer cet ouvrage. Je les en remercie vivement et suis certain que sa lecture vous sera utile et agréable.

Nat Makarevitch (empreinte PGP/GPG : `2010 4A02 9C0E 7D1F 5631 ADF0 453C 4549 0230 D602`)

Avant-propos

Linux a le vent en poupe depuis quelques années, et sa popularité croissante convainc de plus en plus de faire le grand saut. Cette aventure commence par le choix d'une distribution, décision importante car les différentes distributions diffèrent sensiblement. Autant s'épargner de futurs efforts inutiles de migration vers une autre distribution !

Debian GNU/Linux est une distribution Linux « généraliste », convenant a priori à tous. Je vous propose d'en découvrir toutes les facettes ; vous pourrez donc la retenir (ou pas) en toute connaissance de cause...

Pourquoi ce livre ?

Linux commence à bénéficier d'une couverture médiatique non négligeable, profitant essentiellement aux distributions commerciales (Red Hat, SuSE, Mandrake...). Debian, souvent placée par les sondages dans le trio de tête des distributions les plus populaires, est pourtant loin d'être marginale. En 2003, les lecteurs du *Linux Journal* lui ont attribué le titre de « distribution Linux préférée ». Il est donc difficile de la négliger.

Ce livre a ainsi pour vocation de faire découvrir cette distribution. J'espère vous faire profiter de toute l'expérience acquise depuis que j'ai rejoint le projet en tant que développeur-contributeur, en 1997. Peut-être parviendrai-je à vous communiquer mon enthousiasme, et vous donner l'envie de rejoindre nos rangs d'ici quelque temps, qui sait...

Il comble aussi un manque criant : pour autant que je sache, c'est le premier livre français consacré exclusivement à Debian.

B.A.-BA **Distribution et noyau Linux**

Linux n'est en fait qu'un noyau, la brique logicielle de base assurant l'interface entre le matériel et les programmes.
Une distribution Linux est un système d'exploitation complet incluant un noyau Linux, un programme d'installation, et surtout des applications et utilitaires transformant l'ordinateur en outil réellement exploitable.

CULTURE **Distributions commerciales**

La plupart des distributions Linux sont adossées à une entreprise commerciale qui les développe et les commercialise. C'est par exemple le cas de *Mandrake Linux*, réalisée par la société française *MandrakeSoft SA*, ou encore celui de *SuSE LINUX*, œuvre de la société allemande *Suse Linux AG* (passée dans le giron de Novell en novembre 2003).
À l'instar de l'*Apache Software Foundation*, qui développe les serveurs web du même nom, Debian est avant tout un projet du monde du logiciel libre. C'est une organisation regroupant des bénévoles qui coopèrent par l'Internet.

À qui s'adresse cet ouvrage ?

Ses divers niveaux de lecture permettront à différents profils d'en tirer le meilleur parti. En premier lieu, les administrateurs système (débutants ou expérimentés) y trouveront des explications sur l'installation de Debian et son déploiement sur de nombreux postes. Ils passeront aussi en revue un ensemble relativement étoffé de services disponibles sur Debian et les instructions de configuration correspondantes, qui prennent en compte les spécificités et améliorations de la distribution. La compréhension des mécanismes régissant le développement de Debian leur permettra encore de faire face à tout imprévu, en s'appuyant au besoin sur la collaboration des membres de la communauté.

Les utilisateurs d'une autre distribution Linux ou d'un autre Unix découvriront les spécificités de Debian ; ils y seront ainsi très vite opérationnels, tout en bénéficiant des avantages propres à cette distribution.

Enfin, tous ceux qui connaissent déjà un peu Debian et souhaitent en savoir plus sur son fonctionnement communautaire seront exaucés. Après la lecture de ce livre, ils pourront rejoindre les rangs de nos contributeurs.

Approche adoptée

Toutes les documentations génériques s'appliquent à Debian GNU/Linux, qui propose les logiciels libres les plus courants. En vous y limitant, vous en négligeriez pourtant les améliorations apportées par cette distribution. C'est pourquoi j'ai pris le parti de présenter en priorité les manières de procéder recommandées par Debian.

C'est bien de suivre le chemin tracé par Debian, mais c'est encore mieux d'en comprendre les tenants et les aboutissants. Je ne me contenterai donc pas d'explications pratiques, mais détaillerai également le fonctionnement du projet, afin de vous fournir des connaissances complètes et cohérentes.

Structure du livre

Comme tous les ouvrages de cette collection, ce livre s'articulera autour d'un cas d'étude concret qui servira à la fois de support et d'illustration pour tous les sujets traités.

Le chapitre 1, réservé à une présentation non technique de Debian, en exposera les objectifs et le mode de fonctionnement. Ces aspects sont importants, car ils permettent de fixer un cadre où viendront se greffer les contenus des autres chapitres.

Les chapitres 2 et 3 présenteront les grandes lignes de l'étude de cas retenue.

Site web et courriel de l'auteur

Une section de mon site web est dédiée à ce livre, et hébergera tout ce qui peut le compléter utilement. On y trouvera par exemple une liste (cliquable) de toutes les URL citées, ou encore les éventuels errata découverts après impression. N'hésitez pas à la consulter et profitez-en pour me faire part de vos remarques ou messages de soutien en m'écrivant à *hertzog@debian.org*.

▸ http://www.ouaza.com/livre/admin-debian/

Nous débuterons ensuite logiquement par l'installation (chapitre 4), puis découvrirons aux chapitres 5 et 6 les outils de base utiles à tout administrateur Debian, notamment la famille *APT*, largement responsable de la bonne réputation de cette distribution.

Un chapitre intermédiaire, le chapitre 7, présentera des méthodes à suivre pour utiliser efficacement toute la documentation et comprendre rapidement ce qui se passe afin de résoudre les problèmes.

La suite détaillera la configuration pas à pas du système en commençant par les infrastructures et services de base (chapitres 8 à 10) pour remonter progressivement vers les applicatifs utilisateur (chapitre 12).

Le chapitre 13 sera consacré aux administrateurs qui souhaitent aller plus loin et créer des paquets Debian personnalisés.

> VOCABULAIRE **Paquet Debian**
>
> Un paquet Debian est une archive qui renferme un ensemble de fichiers permettant d'installer un logiciel. En général, il s'agit d'un fichier d'extension .deb, qu'on manipule avec le programme **dpkg**. Un paquet sera qualifié de *binaire* s'il contient des fichiers fonctionnels directement utilisables (programmes, documentation) ou de *source* s'il abrite les codes sources du logiciel et les instructions nécessaires à la fabrication du paquet binaire.

Cette deuxième édition traite de nouveaux thèmes, par exemple de la compilation d'un noyau Linux avec les outils Debian (voir page 129). Une nouvelle annexe présente les distributions les plus populaires dérivant de Debian. Enfin, toutes les informations ont été mises à jour en fonction des évolutions de la distribution.

Nous avons placé dans les marges des notes et remarques diverses. Elles ont plusieurs rôles : attirer votre attention sur un point délicat, compléter ou détailler une notion abordée dans le cas d'étude, définir un terme, ou faire des rappels. Voici une liste non exhaustive de ces encadrés :

- *B.A.-BA* : rappelle une information supposée connue du lecteur ;
- *VOCABULAIRE* : définit un terme technique spécifique au projet Debian ;
- *COMMUNAUTÉ* : présente des personnages importants ou les rôles définis au sein du projet ;
- *CHARTE DEBIAN* : évoque une règle ou recommandation de la « charte Debian ». Ce document essentiel décrit comment empaqueter les logiciels. Toutes ces connaissances s'avéreront utiles pour découvrir un nouveau logiciel. Tout paquet Debian devant se conformer à la charte, on saura ainsi où en trouver la documentation, des exemples de fichiers de configuration, etc.
- *OUTIL* : présente un outil ou service pertinent ;
- *EN PRATIQUE* : la pratique a parfois des spécificités, que présenteront ces encadrés. Ils pourront aussi donner des exemples explicites et concrets ;
- d'autres encadrés, moins fréquents, sont relativement explicites : *CULTURE*, *ASTUCE*, *EN CAS DE COUP DUR*, *ATTENTION*, *POUR ALLER PLUS LOIN*, *SPÉCIFICITÉ DEBIAN*, *SÉCURITÉ*...

Cédérom d'accompagnement

Le cédérom offert avec ce livre permet d'installer Debian GNU/Linux (pour architecture *i386*) simplement en y amorçant l'ordinateur. Ainsi, après avoir installé cette distribution, il sera directement possible de mettre en pratique les enseignements du livre. Le disque contient en effet la quasi-totalité des programmes étudiés (mis à part les bureaux graphiques GNOME et KDE, trop volumineux).

Tous les détails sur le fonctionnement du programme d'installation sont donnés dans le chapitre 4.

Remerciements

En premier lieu, je tiens à remercier Nathanaël Makarevitch, qui m'a proposé d'écrire ce livre et m'a accompagné tout au long de sa réalisation ; merci également à toute l'équipe d'Eyrolles qui a contribué à ce livre et notamment à Muriel Shan Sei Fan, très patiente avec moi. Merci à Sébastien Blondeel et à Florence Henry pour leurs contributions.

Ce livre ne serait pas ce qu'il est sans les relecteurs qui m'ont fait part de leurs judicieuses remarques : Christophe Le Bars et Roland Mas en particulier. Merci aussi à Charles-André Habib.

Je remercie également Thierry Stempfel pour les belles illustrations introduisant chaque chapitre.

Merci enfin à Sophie d'avoir été si patiente avec moi et de m'avoir soutenu jusqu'au bout.

Table des matières

1. Le projet Debian 2
Qu'est-ce que Debian ? 4
Un système d'exploitation multi-plate-formes 4
La qualité des logiciels libres 5
Le cadre : une association 6
Les textes fondateurs 6
L'engagement vis-à-vis des utilisateurs 6
Les principes du logiciel libre selon Debian 7
Fonctionnement du projet Debian 10
Les développeurs Debian 10
Le rôle actif des utilisateurs 13
Équipes et sous-projets 15
Rôle d'une distribution 19
L'installateur : **`debian-installer`** 19
La bibliothèque de logiciels 19
Cycle de vie d'une *release* 20
Le statut *experimental* 20
Le statut *unstable* 20
La migration vers *testing* 22
La promotion de *testing* en *stable* 22

2. Présentation de l'étude de cas 26
Des besoins informatiques en forte hausse 28
Plan directeur 28
Pourquoi une distribution GNU/Linux ? 29
Pourquoi la distribution Debian ? 30
Distributions communautaires et commerciales 31
Pourquoi Debian Sarge ? 32

3. Prise en compte de l'existant et migration 34
Coexistence en environnement hétérogène 36
Intégration avec des machines Windows 36
Intégration avec des machines Mac OS 36
Intégration avec d'autres machines Linux/Unix 36
Démarche de migration 36
Recenser et identifier les services 37
Conserver la configuration 38
Prendre en main un serveur Debian existant 39
Installer Debian 40
Installer et configurer les services sélectionnés 41

4. Installation 42
Méthodes d'installation 44
Installation depuis un cédérom 44
Démarrage depuis une clé USB 45
Installation par *boot* réseau 45
Étapes du programme d'installation 46
Exécution du programme d'installation 46
Choix de la langue 46
Choix du pays 47
Choix de la disposition du clavier 48
Détection du matériel 48
Chargement des composants 49
Détection du matériel réseau 49
Configuration du réseau 49
Détection des disques et autres périphériques 50
Démarrage de l'outil de partitionnement 50
Installation du système de base Debian 55
Installation du chargeur d'amorçage GRUB 55
Terminer l'installation et redémarrer 55
Le premier démarrage 56
Horloge et fuseau horaire 56
Mot de passe administrateur 56
Création du premier utilisateur 56
Configuration de l'outil Debian de gestion de paquets (**`apt`**) 56
Installation de logiciels supplémentaires 57
Mise à jour du système 58
Fin de l'installation 58

5. Système de paquetage, outils et principes fondamentaux 60
Structure d'un paquet binaire 62
Méta-informations d'un paquet 63
Description : fichier **`control`** 63
Scripts de configuration 67
Sommes de contrôle, liste des fichiers de configuration 70
Structure d'un paquet source 72
Format 72
Utilité chez Debian 73
Manipuler des paquets avec `dpkg` 74
Installation de paquets 74
Suppression de paquet 75
Autres fonctionnalités de **`dpkg`** 76
Cohabitation avec d'autres systèmes de paquetages 78

6. Maintenance et mise à jour : les outils APT 80
Renseigner le fichier `sources.list` 82
Commande `apt-get` 84
Initialisation 84
Installation et suppression 84
Mise à jour 86
Options de configuration 86
Gérer les priorités associées aux paquets 87
Travailler avec plusieurs distributions 89
Commande `apt-cache` 90
Frontaux : `aptitude`, `synaptic`, `gnome-apt` 91
Vérification d'authenticité des paquets 92
Mise à jour automatique 94
Configuration de **`dpkg`** 94
Configuration d'APT 95
Configuration de **`debconf`** 95
Gestion des interactions en ligne de commande 95
La combinaison miracle 95

7. Résolution de problèmes et sources d'information 98
Les sources de documentation 100
Les pages de manuel 100
Documentation au format *info* 102
La documentation spécifique 102
Les sites web 103
Les *HOWTO* 103
Procédures type 104
Configuration d'un logiciel 104
Surveiller l'activité des démons 105
Demander de l'aide sur une liste de diffusion 106
Signaler un bogue en cas de problème incompréhensible 106

8. Configuration de base : réseau, comptes, impression... 108
Francisation du système 110
Définir la langue par défaut 110
Configurer le clavier en mode console 111
Configurer le clavier en mode graphique 111
Configuration du réseau 112
Interface Ethernet 112
Interface PPP 113
Attribution et résolution des noms 114
Résolution de noms 115
Base de données des utilisateurs et des groupes 116
Liste des utilisateurs : **`/etc/passwd`** 116
Le fichier des mots de passe chiffrés et cachés : **`/etc/shadow`** 117
Modifier un compte ou mot de passe existant 117
Bloquer un compte 118
Liste des groupes : **`/etc/group`** 118
Création de comptes 118
Environnement des interpréteurs de commandes 119
Configuration de l'impression 121
Configuration du chargeur d'amorçage 121
Identifier ses disques 121
Configuration de LILO 122
Configuration de GRUB 123
Cas des Macintosh : configuration de Yaboot 124
Autres configurations : synchronisation, logs, partages... 125
Fuseau horaire 125
Rotation des fichiers de logs 125
Synchronisation horaire 126
Partage des droits d'administration 126
Liste des points de montage 127
`locate` et **`updatedb`** 128
Compilation d'un noyau 129
Introduction et prérequis 129
Récupérer les sources 129
Configuration du noyau 130
Compilation et génération du paquet 131
Compilation de modules externes 132
Emploi d'un patch sur le noyau 133
Installation d'un noyau 133
Caractéristiques d'un paquet Debian du noyau 133
Installation avec **`dpkg`** 134

9. Services Unix 136
Démarrage du système 138
Connexion à distance 140
Connexion à distance : **`telnet`** 140
Connexion à distance sécurisée : SSH 140
Accéder à distance à des bureaux graphiques 142
Gestion des droits 142
Interfaces d'administration 144
Administrer sur interface web : **`webmin`** 145
Configuration des paquets : **`debconf`** 145
Les événements système de `syslog` 146
Principe et fonctionnement 146
Le fichier de configuration 147
Le super-serveur `inetd` 148
Planification synchrone : `cron` et `atd` 149
Format d'un fichier **`crontab`** 150
Emploi de la commande **`at`** 151
Planification asynchrone : `anacron` 152

Les quotas 152
Supervision 153
Surveillance des logs avec `logcheck` 153
Surveillance de l'activité 154
Sauvegarde 155
Branchements « à chaud » : `hotplug` 157
Gestion de l'énergie 157
Gestion avancée de l'énergie : APM 157
Économie d'énergie moderne : ACPI 157
Cartes pour portables : PCMCIA 158

10. Infrastructure réseau 160
Passerelle 162
Pare-feu ou filtre de paquets 163
Fonctionnement de *netfilter* 163
Syntaxe d'`iptables` 165
Créer les règles 166
Installer les règles à chaque démarrage 167
Réseau privé virtuel 168
SSH et PPP 168
IPsec 168
PPTP 169
Qualité de service 172
Principe et fonctionnement 172
Configuration et mise en œuvre 172
Routage dynamique 174
IPv6 175
Serveur de noms (DNS) 176
Principe et fonctionnement 176
Configuration 177
DHCP 179
Présentation 179
Configuration 179
DHCP et DNS 180
Détection d'intrusion (IDS/NIDS) 181

11. Services réseau : Postfix, Apache, NFS, Samba, Squid, LDAP 182
Serveur de messagerie électronique 184
Installation de Postfix 184
Configuration de domaines virtuels 186
Restrictions à la réception et à l'envoi 188
Intégration d'un antivirus 192
Serveur web (HTTP) 194
Installation d'Apache 194
Configuration d'hôtes virtuels 195
Directives courantes 196
Analyseur de logs 197
Serveur de fichiers NFS 199
Sécuriser NFS (au mieux) 200
Serveur NFS 201
Client NFS 202
Partage Windows avec Samba 203
Samba en serveur 203
Samba en client 207
Mandataire HTTP/FTP 208
Installation 209
Configuration d'un cache 209
Configuration d'un filtre 209
Annuaire LDAP 210
Installation 211
Remplissage de l'annuaire 212
Utiliser LDAP pour gérer les comptes 213

12. Station de travail 220
Configuration de XFree86 222
Détection automatique 222
Script de configuration 223
Configuration du clavier 224
Configuration de la souris 224
Configuration de l'écran 224
Personnalisation de l'interface graphique 225
Choix d'un gestionnaire d'écran (*display manager*) 225
Choix d'un gestionnaire de fenêtres 225
Gestion des menus 226
Bureaux graphiques 227
GNOME 228
KDE 228
Xfce et autres 229
Outils 229
Courrier électronique 229
Navigateurs web 230
Développement 231
Travail collaboratif 231
Suites bureautiques 235
L'émulation Windows : Wine, VMWare, VNC, QEMU... 236

13. Conception d'un paquet Debian 238
Recompiler un paquet depuis ses sources 240
Récupérer les sources 240
Effectuer les modifications 240
Démarrer la recompilation 241
Construire son premier paquet 242
Méta-paquet ou faux paquet 242

Simple archive de fichiers 243
Créer une archive de paquets pour APT 246
Devenir mainteneur de paquet 248
Apprendre à faire des paquets 248
Processus d'acceptation 250

14. Conclusion : l'avenir de Debian 254
Développements à venir 256
Avenir de Debian 256
Avenir de ce livre 257

Annexe : Distributions dérivées 258
Ubuntu Linux 260
Knoppix 260
Mepis Linux 261
Xandros 261
Libranet 261
Linspire 262

Glossaire 265

Index 285



1



Le projet Debian

Avant de plonger dans la technique, découvrons ensemble ce qu'est le projet Debian : ses objectifs, ses moyens et son fonctionnement.

SOMMAIRE

- Qu'est-ce que Debian ?
- Les textes fondateurs
- Fonctionnement du projet Debian
- Rôle d'une distribution
- Cycle de vie d'une *release*

MOTS-CLEFS

- Objectif
- Moyens
- Fonctionnement
- Bénévole

Qu'est-ce que Debian ?

Lorsqu'il a créé Debian en 1993 sous l'impulsion de la FSF, Ian Murdock avait des objectifs clairs, qu'il a exprimés dans le *Manifeste Debian*. Le système d'exploitation libre qu'il recherchait devait présenter deux caractéristiques principales. En premier lieu, la qualité : Debian serait développée avec le plus grand soin, pour être digne du noyau Linux. Ce serait également une distribution non commerciale suffisamment crédible pour concurrencer les distributions commerciales majeures. Cette double ambition ne serait à son sens atteinte qu'en ouvrant le processus de développement de Debian, à l'instar de Linux et de GNU. Ainsi, la revue des pairs améliorerait constamment le produit.

CULTURE **Origine du nom de Debian**

Ne cherchez plus, Debian n'est pas un acronyme. Ce nom est en réalité une contraction de deux prénoms : celui de Ian Murdock et de sa femme Debra. Debra + Ian = Debian

CULTURE **GNU, le projet de la FSF**

Le projet GNU est un ensemble de logiciels libres développés ou parrainés par la *Free Software Foundation* (FSF), dont Richard Stallman est le créateur emblématique. C'est l'acronyme récursif de « GNU N'est pas Unix ».

Un système d'exploitation multi-plate-formes

Debian, restée fidèle à ses principes initiaux, a connu un tel succès qu'elle atteint aujourd'hui une taille pharaonique. Ses 12 architectures et plus de 8000 paquets sources disponibles couvrent désormais presque tout le spectre des matériels existants et domaines d'application imaginables.

Cet embonpoint devient parfois gênant : il est peu raisonnable de distribuer 14 cédéroms de Debian... C'est pourquoi on la considère de plus en plus comme une « méta-distribution », dont on extrait des distributions plus spécifiques et orientées vers un public particulier : Debian-Desktop pour un usage bureautique traditionnel, Debian-Edu pour un emploi éducatif et pédagogique en milieu scolaire, Debian-Med pour les applications médicales, *Debian Jr.* (*Debian Junior*) pour les jeunes enfants, etc.

Ces scissions, organisées dans un cadre bien défini et garantissant une compatibilité entre les différentes « sous-distributions », ne posent aucun problème. Toutes suivent le planning général des publications de nouvelles versions. S'adossant sur les mêmes briques de base, elles peuvent facilement être étendues, complétées et personnalisées par des applications disponibles au niveau de Debian.

Tous les outils de Debian évoluent dans cette direction : **`debian-cd`** permet depuis longtemps de créer des jeux de cédéroms ne comportant que des paquets

COMMUNAUTÉ **Le parcours de Ian Murdock**

Ian Murdock, fondateur du projet Debian, en fut le premier leader, de 1993 à 1996. Après avoir passé la main à Bruce Perens, il s'est fait plus discret. Il est ensuite revenu sur le devant de la scène du logiciel libre en créant la société Progeny, visant à commercialiser une distribution dérivée de Debian. Ce fut un échec commercial, au développement depuis abandonné. Mais les contributions de Progeny subsistent : citons *Progeny Graphical Installer (PGI)* (installateur graphique) ou *discover* (détection automatique du matériel). Plus récemment, Progeny, devenue une SSII spécialiste des logiciels libres, a adapté l'installateur automatique de Red Hat (anaconda) pour permettre son utilisation avec Debian. Son dernier projet, *Componentized Linux*, consiste en une nouvelle approche d'une distribution Linux. On n'y assemble plus des paquets, mais des collections de paquets cohérents, plus faciles à gérer. Chacune progresse indépendamment des autres, facilitant ainsi l'évolution globale du système.

B.A.-BA **À chaque ordinateur son architecture**

Le terme « architecture » désigne un type d'ordinateur (les plus connues sont Mac ou PC). Chaque architecture se différencie principalement par son modèle de processeur, généralement incompatible avec les autres. Ces différences de matériel impliquent des fonctionnements distincts et imposent une compilation spécifique de tous les logiciels pour chaque architecture.
La plupart des logiciels disponibles pour Debian sont écrits en des langages de programmation portables : le même code source est compilé sur les diverses architectures. En effet, un exécutable binaire, toujours compilé pour une architecture donnée, ne fonctionne généralement pas sur les autres.
Rappelons que chaque logiciel est créé en rédigeant un code source ; il s'agit d'un fichier textuel composé d'instructions provenant d'un langage de programmation. Avant de pouvoir utiliser le logiciel, il est nécessaire de compiler le code source, c'est-à-dire de le transformer en code binaire (une succession d'instructions machines exécutables par le processeur). Chaque langage de programmation dispose d'un compilateur pour effectuer cette opération (par exemple **gcc** pour le langage C).

préalablement sélectionnés ; **debian-installer** est également un installateur modulaire, facilement adaptable à des besoins particuliers. **APT** installera des paquets d'origines diverses tout en garantissant la cohérence globale du système.

OUTIL **Nouvel installateur**

debian-installer, le plus récent programme d'installation de Debian, fut développé pour remplacer **boot-floppies**. Sa conception modulaire permet de l'employer dans un grand nombre de scénarios d'installation différents. Le travail de développement est coordonné sur la liste de diffusion debian-boot@lists.debian.org sous la direction de Joey Hess (joeyh@debian.org).

OUTIL **Créer un cédérom Debian**

debian-cd permet de créer des images ISO de cédéroms d'installation prêts à l'emploi. J'en suis le mainteneur, et l'auteur de la dernière réécriture. Tout ce qui concerne ce logiciel se discute (en anglais) sur la liste de diffusion debian-cd@lists.debian.org.

La qualité des logiciels libres

Debian suit tous les principes du logiciel libre, et ses nouvelles versions ne sortent que lorsqu'elles sont prêtes. Aucun calendrier préétabli ne contraint les développeurs à bâcler pour respecter une échéance arbitraire. On reproche donc souvent à Debian ses délais de publication, mais cette prudence en garantit aussi la légendaire fiabilité : de longs mois de tests sont en effet nécessaires pour que la distribution complète reçoive le label « stable ».

Debian ne transige pas sur la qualité : tous les *bogues* critiques connus seront corrigés dans toute nouvelle version, même si cela doit retarder la date de sortie initialement prévue.

Debian n'exclut aucune catégorie d'utilisateurs, aussi minoritaire soit-elle. Son programme d'installation est longtemps resté fruste, car c'était le seul capable de fonctionner sur toutes les architectures gérées par le noyau Linux. Il n'était pas envisageable de le remplacer par un programme plus convivial mais limité aux PC (architecture *i386*). Heureusement, depuis l'arrivée de **debian-installer**, cette époque est révolue.

Le cadre : une association

Juridiquement parlant, Debian est un projet mené par une association sans but lucratif américaine regroupant des bénévoles, similaire à nos associations loi 1901. Le projet compte un millier de *développeurs Debian* mais fédère un nombre bien plus important de contributeurs (traducteurs, rapporteurs de bogues, développeurs occasionnels...).

Pour mener à bien sa mission, Debian dispose d'une importante infrastructure, comportant de nombreux serveurs reliés à l'Internet. De nombreux mécènes offrent à la fois le matériel et l'hébergement sur l'Internet.

COMMUNAUTÉ **Derrière Debian, l'association SPI**

Debian ne possède aucun serveur en son nom propre, puisque ce n'est qu'un projet au sein de l'association *Software in the Public Interest* (SPI), qui en gère les aspects matériels et financiers (dons, achat de matériel…). Bien qu'initialement créée sur mesure pour Debian, cette association coiffe maintenant d'autres projets du monde du logiciel libre.

▸ http://www.spi-inc.org

Les textes fondateurs

Quelques années après son lancement, Debian a formalisé les principes qu'elle devait suivre en tant que projet de logiciel libre. Cette démarche militante permet une croissance sereine en s'assurant que tous les membres progressent dans la même direction. Pour devenir développeur Debian, tout candidat doit d'ailleurs convaincre de son adhésion aux principes établis dans les textes fondateurs du projet.

Le processus de développement est constamment débattu, mais ces textes fondateurs sont très consensuels, même si non inaltérables. La constitution Debian offre toutefois des garanties supplémentaires : une majorité qualifiée de trois quarts est nécessaire pour approuver tout amendement.

L'engagement vis-à-vis des utilisateurs

On trouve aussi un « contrat social ». Quelle est la place d'un tel texte dans un projet ne visant qu'à concevoir un système d'exploitation ? C'est très simple, Debian œuvre pour ses utilisateurs, et, par extension, pour la société. Ce contrat résume donc les engagements pris. Voyons ces points plus en détail :

1. Debian demeurera un ensemble logiciel totalement libre.
   C'est la règle numéro un. Debian est et restera constituée exclusivement de logiciels libres. De plus, tous les logiciels développés en propre par Debian seront libres.
2. Nous donnerons en retour à la communauté du logiciel libre.
   Toute amélioration apportée par le projet Debian à un logiciel intégré à la distribution est envoyée à l'auteur de ce dernier (dit « amont »). D'une manière générale, Debian coopère avec la communauté au lieu de travailler isolément.
3. Nous ne cacherons pas les problèmes.
   Debian n'est pas parfaite, et l'on y découvre tous les jours des problèmes à corriger. Tous ces bogues sont répertoriés et consultables librement, par exemple sur le Web.

COMMUNAUTÉ **Auteur amont ou développeur Debian ?**

Traduction littérale de *upstream author*, le terme « auteur amont » désigne le ou les auteurs/développeurs d'un logiciel, qui l'écrivent et le font évoluer. A contrario, un « développeur Debian » se contente en général de partir d'un logiciel existant pour le transformer en paquet Debian (la désignation « mainteneur Debian » est plus explicite).
Bien souvent, la ligne de démarcation n'est pas aussi nette. Le mainteneur Debian écrit parfois un correctif, qui profite à tous les utilisateurs du logiciel. De manière générale, Debian encourage l'implication des responsables de paquets dans le développement « amont » (ils deviennent alors contributeurs sans se cantonner au rôle de simples utilisateurs d'un logiciel).

COMMUNAUTÉ **Pour ou contre la section non-free ?**

L'engagement de conserver une structure d'accueil pour des logiciels non libres (i.e. la section *non-free*, voir encadré « VOCABULAIRE » page 83) est régulièrement remis en cause au sein de la communauté Debian.
Ses détracteurs arguënt qu'il détourne certaines personnes de logiciels libres équivalents et contredit le principe de servir exclusivement la cause des logiciels libres. Les partisans (dont je suis) rappellent plus prosaïquement que la majorité des logiciels de *non-free* sont des logiciels « presque libres », entravés seulement par une ou deux restrictions gênantes (la plus fréquente étant l'interdiction de tirer un bénéfice commercial du logiciel). En distribuant ces logiciels dans la branche *non-free*, on explique indirectement à leur auteur que leur création serait mieux reconnue et plus utilisée si elle pouvait être intégrée dans la section *main* : ils sont ainsi poliment invités à changer leur licence pour servir cet objectif.

La suppression totale de la section *non-free* n'est pas encore à l'ordre du jour, mais tôt ou tard nous y viendrons. Signalons que son existence gêne considérablement la *Free Software Foundation*, qu'elle empêche de recommander officiellement Debian comme système d'exploitation.
Le scénario le plus probable est à mon sens le suivant :

- suppression du point 5 du contrat social, Debian n'est plus contrainte de fournir *non-free*, mais continue de le faire pendant une durée indéterminée ;
- mise en place d'un site tiers dans le but d'héberger les paquets Debian de logiciels non libres (le nom de domaine `nonfree.org` a été réservé dans ce but par Bruce Perens) ;
- suppression officielle de *non-free* au sein de Debian (décision avalisée par un vote de l'ensemble des développeurs).

4. Nos priorités sont nos utilisateurs et les logiciels libres.

   Cet engagement est plus difficile à définir. Debian s'impose ainsi un biais lorsqu'elle doit prendre une décision, et écartera une solution de facilité pénalisante pour ses utilisateurs au profit d'une solution plus élégante, même si plus difficile à mettre en œuvre. Il s'agit de prendre en compte en priorité les intérêts des utilisateurs et du logiciel libre.

5. Programmes non conformes à nos standards sur les logiciels libres.

   Debian accepte et comprend que ses utilisateurs souhaitent utiliser certains logiciels non libres. Elle s'engage donc à mettre à leur disposition une partie de son infrastructure, pour distribuer sous forme de paquets Debian les logiciels qui l'autorisent.

COMMUNAUTÉ **Responsable de paquet ou mainteneur ?**

L'équipe chargée de l'adaptation de Debian en français (on parle de « localisation ») a retenu le terme de « responsable de paquet » pour désigner la personne chargée d'intégrer un paquet à Debian et de l'y faire évoluer. Le terme anglais correspondant est *maintainer* ; c'est pourquoi j'emploie souvent le mot « mainteneur », plus concis et tout aussi explicite.

Les principes du logiciel libre selon Debian

Ce texte de référence définit quels logiciels sont « suffisamment libres » pour être intégrés à Debian. Si la licence d'un logiciel est conforme à ces principes, il peut être intégré à la section *main* ; dans le cas contraire, et si sa libre redistribution est permise, il peut rejoindre la section *non-free*. Celle-ci ne fait pas officiellement partie de Debian : il s'agit d'un service annexe fourni aux utilisateurs.

Plus qu'un critère de choix pour Debian, ce texte fait autorité en matière de logiciel libre puisqu'il a servi de socle à la « définition de l'Open Source ». C'est donc historiquement la première formalisation de la notion de « logiciel libre ».

La licence publique générale de GNU (*GNU General Public License*), la licence BSD et la licence artistique sont des exemples de licences libres traditionnelles respectant les 9 points mentionnés dans ce texte. Vous en trouverez ci-dessous la traduction, telle que publiée sur le site web de Debian.

▸ http://www.debian.org/social_contract.fr.html#guidelines

B.A.-BA **Les licences libres**

La GNU GPL, la licence BSD et la licence artistique respectent toutes trois les principes du logiciel libre selon Debian. Elles sont pourtant très différentes.

La GNU GPL, utilisée et promue par la FSF (*Free Software Foundation*, ou fondation du logiciel libre), est la plus courante. Elle a pour particularité de s'appliquer à toute œuvre dérivée et redistribuée : un programme intégrant ou utilisant du code GPL ne peut être diffusé que selon ses termes. Elle interdit donc toute récupération dans une application propriétaire. Ceci pose également de gros problèmes pour le réemploi de code GPL dans des logiciels libres incompatibles avec cette licence. Ainsi, il est parfois impossible de lier une bibliothèque diffusée sous GPL à un programme placé sous une autre licence libre. En revanche, cette licence est très solide en droit américain : les juristes de la FSF ont participé à sa rédaction, et elle a souvent contraint des contrevenants à trouver un accord amiable avec la FSF sans aller jusqu'au procès.

▸ http://www.gnu.org/copyleft/gpl.html

La licence BSD est la moins restrictive : tout est permis, y compris l'intégration de code BSD modifié dans une application propriétaire. Microsoft ne s'en est d'ailleurs pas privé car la couche TCP/IP de Windows NT est fondée sur celle du noyau BSD.

▸ http://www.opensource.org/licenses/bsd-license.php

Enfin, la licence artistique réalise un compromis entre les deux précédentes : l'intégration du code dans une application propriétaire est possible, mais toute modification doit être publiée.

▸ http://www.opensource.org/licenses/artistic-license.php

Retrouvez le texte complet de ces licences dans /usr/share/common-licenses/ sur tout système Debian.

1. **Redistribution libre et gratuite**

   La licence d'un composant de Debian ne doit pas empêcher un contractant de vendre ou donner le logiciel sous forme de composant d'un ensemble (distribution) constitué de programmes provenant de différentes sources. La licence ne doit requérir ni redevance ni rétribution sur une telle vente.

2. **Code source**

   Le programme doit inclure le code source, et la diffusion sous forme de code source comme sous forme de programme compilé doit être autorisée.

3. **Applications dérivées**

   La licence doit permettre les modifications et les applications dérivées, et elle doit permettre à celles-ci d'être distribuées sous les mêmes termes que la licence du logiciel original.

4. **Intégrité du code source de l'auteur**

   La licence peut défendre de distribuer le code source modifié *seulement* si elle autorise la distribution avec le code source de fichiers correctifs destinés à modifier le programme au moment de la génération. La licence doit autoriser explicitement la distribution de logiciels générés à partir de code source modifié. Elle peut requérir que les applications dérivées portent un nom ou un numéro de version différent de ceux du logiciel original (*ceci est un compromis : le groupe Debian encourage tous les auteurs à ne restreindre en aucune manière les modifications d'un quelconque fichier, source ou binaire*).

5. **Aucune discrimination de personne ou de groupe**

   La licence ne doit discriminer aucune personne ou groupe de personnes.

6. **Aucune discrimination de champ d'application**

   La licence ne doit pas défendre d'utiliser le logiciel dans un champ d'application particulier. Par exemple, elle ne doit pas défendre l'utilisation du logiciel dans une entreprise ou pour la recherche génétique.

B.A.-BA **Le copyleft**

Le *copyleft* (ou « gauche d'auteur ») est un principe qui consiste à faire appel au mécanisme des droits d'auteurs pour garantir la liberté d'une œuvre et de ses dérivées — au lieu de restreindre les droits des utilisateurs comme dans le cas des logiciels propriétaires. Il s'agit d'ailleurs d'un jeu de mots sur le terme *copyright*, équivalent américain du droit d'auteur. Richard Stallman a trouvé cette idée quand un ami friand de calembours écrivit sur une enveloppe qu'il lui adressa : « *copyleft : all rights reversed* » (*copyleft* : tous droits renversés). Le *copyleft* impose la conservation de toutes les libertés initiales lors de la distribution d'une version modifiée (ou non) du logiciel. Il est donc impossible de dériver un logiciel propriétaire d'un logiciel placé sous *copyleft*.
La licence *copyleft* la plus célèbre est sans aucun doute la GNU GPL (elle a pour petites sœurs la *GNU LGPL — GNU Lesser General Public License* et la *GNU FDL — GNU Free Documentation License*). Malheureusement, les licences *copyleft* sont généralement incompatibles entre elles ! En conséquence, il est préférable de n'en utiliser qu'une seule.

7. **Distribution de licence**

   Les droits attachés au programme doivent s'appliquer à tous ceux à qui il est distribué sans obligation pour aucune de ces parties de se conformer à une autre licence.

8. **La licence ne doit pas être spécifique à Debian**

   Les droits attachés au programme ne doivent pas dépendre du fait qu'il fasse partie du système Debian. Si le programme est extrait de Debian et est utilisé et distribué sans Debian mais au contraire sous les termes de sa propre licence, toutes les parties auxquelles il est redistribué doivent jouir des même droits que ceux accordés avec le système Debian.

9. **La licence ne doit pas contaminer d'autres logiciels**

   La licence ne doit pas placer de restrictions sur d'autres logiciels distribués avec le logiciel licencié. Par exemple, la licence ne doit pas exiger que tous les autres programmes distribués sur le même médium soient des logiciels libres.

COMMUNAUTÉ **Bruce Perens, un leader chahuté**

Bruce Perens, deuxième leader du projet Debian juste après Ian Murdock, fut très controversé pour ses méthodes dynamiques et assez dirigistes. Il n'en reste pas moins un contributeur important, à qui Debian doit notamment la rédaction des fameux « principes du logiciel libre selon Debian » (ou DFSG pour *Debian Free Software Guidelines*), idée originelle d'Ean Schuessler. Par la suite, Bruce en dérivera la célèbre « définition de l'Open Source » en y gommant toutes les références à Debian.

▸ http://www.opensource.org

Son départ du projet fut quelque peu mouvementé mais Bruce est resté assez fortement attaché à Debian puisqu'il continue de promouvoir cette distribution dans les sphères politiques et économiques. Il intervient encore régulièrement sur les listes de diffusion pour donner son avis et présenter ses dernières initiatives en faveur de Debian.
Dernier point anecdotique, c'est à lui que l'on doit l'inspiration des « noms de code » des différentes versions de Debian (1.1 — *rex*, 1.2 — *buzz*, 1.3 — *bo*, 2.0 — *hamm*, 2.1 — *slink*, 2.2 — *potato*, 3.0 — *woody*, 3.1 — *sarge*, *testing* — *etch*, *unstable* — *sid*). Ils correspondent tous à des personnages de *Toy Story*. Ce film d'animation entièrement réalisé en images de synthèse fut produit par Pixar, employeur de Bruce à l'époque où il était leader Debian. Le nom « Sid » a un statut particulier puisqu'il restera éternellement associé à *unstable* ; dans le film, il s'agit de l'enfant des voisins, incorrigible brise-tout — gare à vous donc si vous approchez *unstable* de trop près ! Par ailleurs, *sid* est l'acronyme de *Still In Development* (encore et toujours en cours de développement).

Fonctionnement du projet Debian

La richesse produite par le projet Debian résulte à la fois du travail sur l'infrastructure effectué par des développeurs Debian expérimentés, du travail individuel ou collectif de développeurs sur des paquets Debian, et des retours des utilisateurs.

Les développeurs Debian

Les développeurs Debian ont des responsabilités diverses : membres attitrés du projet, ils infléchissent grandement les directions qu'il prend. Un développeur Debian maintient au minimum un paquet, mais selon son temps disponible et ses envies il a le loisir de s'engager dans de nombreuses équipes, développant ainsi ses responsabilités.

- http://www.debian.org/devel/people
- http://www.debian.org/intro/organization

La maintenance des paquets est une activité relativement codifiée, largement documentée voire réglementée. Il faut en effet y respecter toutes les normes édictées par la *charte Debian* (connue en anglais sous le nom de *Debian Policy*). Fort heureusement, de nombreux outils facilitent le travail du mainteneur. Il peut ainsi se focaliser sur les particularités de son paquet et sur les tâches plus complexes, telles que la correction des bogues.

- http://www.debian.org/doc/debian-policy/

La charte, élément essentiel du projet Debian, énonce les normes assurant à la fois la qualité des paquets et la parfaite interopérabilité de l'ensemble. Grâce à elle, Debian reste cohérent malgré sa taille gigantesque. Cette charte n'est pas figée, mais évolue continuellement grâce aux propositions incessamment formulées sur la liste debian-policy@lists.debian.org. Les amendements emportant l'adhésion

B.A.-BA **Maintenance d'un paquet, le travail du développeur**

Maintenir un paquet suppose d'abord d'« empaqueter » un logiciel. Concrètement, il s'agit d'en définir les modalités d'installation, afin qu'une fois installé, ce logiciel soit fonctionnel et respecte l'ensemble des règles que Debian s'astreint à suivre. Le résultat de cette opération est conservé dans une archive .deb. L'installation effective du logiciel se limitera ensuite à l'extraction de cette archive, ainsi qu'à l'exécution de quelques scripts de pré- ou post-installation.
Après cette phase initiale, le cycle de la maintenance débute vraiment : préparation des mises à jour pour respecter la dernière version de la charte Debian, correction des bogues signalés par les utilisateurs, inclusion d'une nouvelle version « amont » du logiciel, qui continue naturellement d'évoluer en parallèle (ex : lors de l'empaquetage le logiciel en était la version 1.2.3. Après quelques mois de développement, ses auteurs originaux sortent une nouvelle version stable, numérotée 1.4.0. Il convient alors de mettre à jour le paquet Debian pour que les utilisateurs puissent bénéficier de sa dernière version stable).

OUTIL **Base de données des développeurs**

Debian dispose d'une base de données comprenant l'ensemble des développeurs enregistrés et les informations qui s'y rattachent (adresse, téléphone, coordonnées géographiques — latitude et longitude…). Certaines de ces informations (nom, prénom, pays, identifiant chez Debian, identifiant IRC, clé GnuPG…) sont publiques et disponibles sur le Web.

- http://db.debian.org

Les coordonnées géographiques permettent de générer une carte situant l'ensemble des développeurs sur le globe. On constate alors que Debian est vraiment un projet international : on trouve des développeurs sur tous les continents, même si la majorité proviennent de pays occidentaux.

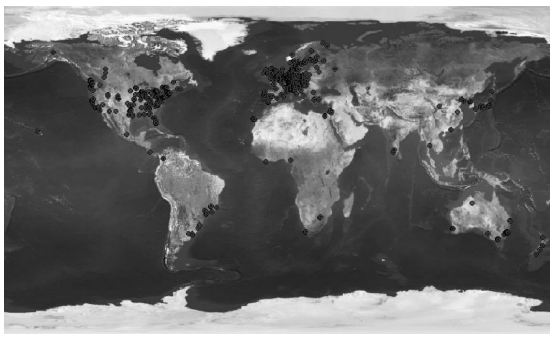


Figure 1–1 Répartition mondiale des développeurs Debian

de tous sont acceptés et appliqués au texte par un petit groupe de mainteneurs sans tâche éditoriale (ils se contentent d'inclure les modifications décidées par les développeurs Debian membres de la liste mentionnée ci-dessus). On peut consulter les actuelles propositions d'amendements via le système de suivi de bogues :

▸ http://bugs.debian.org/debian-policy

COMMUNAUTÉ **Processus éditorial de la charte**

Tout le monde peut proposer une modification de la charte Debian : il suffit de soumettre un rapport de bogue de « sévérité »*wishlist* (souhait) sur le paquet *debian-policy*. Tout développeur Debian peut alors en faire une proposition officielle en incluant la balise « [PROPOSAL] » (proposition) dans le titre du rapport de bogue. L'aval de deux autres développeurs Debian (en anglais, le verbe consacré est *to second*) assure ensuite un minimum de crédibilité à la proposition. Suit une phase de discussion publique sur la liste debian-policy@lists.debian.org, évaluant les tenants et les aboutissants de la proposition. Si aucune objection majeure n'apparaît et qu'un consensus en faveur du changement semble se dégager, le titre du bogue est à nouveau changé pour y inclure la balise « [ACCEPTED] » avec la date d'acceptation, à charge pour les mainteneurs du paquet *debian-policy* d'intégrer l'amendement dans le texte de référence.

La charte encadre très bien tout ce qui a trait au côté technique de la mise en paquet. La taille du projet soulève aussi des problèmes organisationnels ; ils sont traités par la constitution Debian, qui fixe une structure et des moyens de décision.

Cette constitution définit un certain nombre d'acteurs, de postes, les responsabilités et les pouvoirs de chacun. On retiendra que les développeurs Debian ont toujours le pouvoir ultime de décision par un vote de résolution générale — avec nécessité d'obtenir une majorité qualifiée de trois quarts pour les changements les plus importants (comme ceux portant sur les textes fondateurs). Cependant, les développeurs élisent annuellement un « leader » pour les représenter dans les congrès et assurer la coordination interne entre les différentes équipes. Son rôle n'est pas formellement défini par un document, et il est d'usage que chaque candidat à ce poste donne sa propre définition de la fonction. À mon sens, le leader a un rôle représentatif auprès des médias, un rôle de coordination entre les équipes « internes » et un rôle de visionnaire pour donner une ligne directrice au projet, dans laquelle les développeurs peuvent s'identifier.

Concrètement, le leader dispose de pouvoirs réels : sa voix est déterminante en cas d'égalité dans un vote, il peut prendre toute décision qui ne relève pas déjà d'un autre, et déléguer une partie de ses responsabilités.

La constitution définit également un « comité technique ». Son rôle essentiel est de trancher sur des points techniques lorsque les développeurs concernés ne sont pas parvenus à un accord entre eux. Par ailleurs, ce comité joue aussi un rôle de conseil vis-à-vis de chaque développeur qui n'arrive pas à prendre une décision qui lui revient. Il est important de noter qu'il n'intervient que lorsqu'une des parties concernées le lui a demandé.

CHARTE DEBIAN **La documentation**

La documentation de chaque paquet est stockée dans /usr/share/doc/<*paquet*>. Ce répertoire contient souvent un fichier README.Debian décrivant les aménagements spécifiques à Debian réalisés par le mainteneur. Il est donc sage de lire ce fichier avant toute configuration, pour tirer profit de son expérience. On trouve également un fichier changelog.Debian.gz décrivant les modifications effectuées au fil des versions par le mainteneur Debian. Le fichier changelog.gz (ou équivalent) décrit quant à lui les changements effectués au niveau des développeurs amont. Le fichier copyright rassemble les informations concernant les auteurs et la licence à laquelle le logiciel est soumis. Désormais, on trouve parfois un fichier NEWS.Debian.gz, qui permet au développeur Debian de communiquer quelques informations importantes sans utiliser le système de notice proposé par **debconf**. Tous les autres fichiers sont spécifiques au logiciel en question.

Enfin, la constitution définit le poste de « secrétaire du projet », qui a notamment en charge l'organisation des votes liés aux différentes élections et résolutions générales.

La procédure de « résolution générale » est entièrement détaillée dans la constitution, de la période de discussion préalable à l'analyse des résultats des votes. Pour plus de détails, je vous invite à en consulter le texte intégral :

▸ http://www.debian.org/devel/constitution.fr.html

Même si cette constitution instaure un semblant de démocratie, la réalité quotidienne est très différente : Debian suit naturellement les lois du logiciel libre, et sa politique du fait accompli. On peut longtemps débattre des mérites respectifs des différentes manières d'aborder un problème, la solution retenue sera la première fonctionnelle et satisfaisante... celle à la réalisation de laquelle une personne compétente aura consacré une partie de son temps.

C'est d'ailleurs la seule manière d'obtenir des galons : faire quelque chose d'utile et démontrer que l'on a bien travaillé. Beaucoup d'équipes « administratives » de Debian fonctionnent sur le mode de la cooptation, et favoriseront des volontaires ayant déjà effectivement contribué dans le sens de leur action et prouvé leur compétence à la tâche. Cette méthode est envisageable car l'essentiel du travail de ces équipes est public, donc a fortiori accessible à tout développeur intéressé. C'est pourquoi Debian est souvent qualifiée de « méritocratie ».

Ce mode de fonctionnement efficace garantit la qualité des contributeurs au sein des équipes « clés » de Debian. Tout n'est pas parfait pour autant, et il arrive fréquemment que certains n'acceptent pas cette manière de procéder. La sélection des développeurs acceptés dans ces équipes peut paraître quelque peu arbitraire voire injuste. Par ailleurs, tout le monde n'a pas la même définition du service attendu de ces équipes. Pour certains, il est inacceptable de devoir attendre 8 jours l'intégration d'un nouveau paquet Debian ; d'autres patienteront 3 semaines sans peine. Aussi, des esprits chagrins se plaignent régulièrement de la « qualité du service » de certaines équipes.

CULTURE **Flamewar, la discussion qui s'enflamme**

Une *flamewar*, littéralement « guerre enflammée », est une discussion (trop) passionnée qui finit souvent par des attaques personnelles lorsque tous les arguments raisonnés ont été épuisés de part et d'autre. Certains thèmes sont beaucoup plus sujets à polémique que d'autres (l'exemple type étant le choix d'un éditeur de texte, « préférez-vous `vi` ou `emacs` ? »). Ils provoquent de très rapides échanges de courrier électronique du fait du nombre de personnes concernées (tout le monde) et de l'aspect très personnel de cette question.
Rien de très utile ne sortant généralement de ces discussions, abstenez-vous d'y participer et ne survolez que rapidement leur contenu — sa lecture complète serait trop chronophage.

CULTURE **Méritocratie, le règne du savoir**

La méritocratie est une forme de gouvernement où le pouvoir est exercé par les plus « méritants ». Pour Debian, le mérite se mesure à la compétence, elle-même évaluée en observant les réalisations passées des uns et des autres au sein du projet. Leur simple existence prouve un certain niveau de compétence ; ces réalisations étant en général des logiciels libres, aux codes sources disponibles, il sera facile aux pairs d'en juger la qualité.

COMMUNAUTÉ **L'intégration des nouveaux mainteneurs**

L'équipe chargée de l'admission des nouveaux développeurs est la plus régulièrement critiquée. Il faut reconnaître qu'au fil des années le projet Debian est devenu de plus en plus exigeant avec les développeurs qu'il accepte en son sein. On peut y voir une certaine injustice, mais nous admettrons que ce qui n'était que de petits défis au départ prend l'allure de gageures dans une communauté de plus de 1000 personnes, où il s'agit de garantir la qualité et l'intégrité de tout ce que Debian produit pour ses utilisateurs.
Par ailleurs, la procédure d'acceptation se conclut par la revue de la candidature par une personne unique, le « Responsable des comptes Debian » (ou *DAM — Debian Account Manager*). Celui-ci est donc particulièrement exposé aux critiques, puisqu'il accepte ou refuse en dernier recours l'intégration d'un volontaire au sein de la communauté des développeurs Debian. Dans la pratique, il s'agit parfois de retarder l'acceptation d'une personne, le temps qu'elle connaisse mieux le fonctionnement du projet. On peut en effet contribuer à Debian avant d'y être accepté comme développeur officiel grâce à un mécanisme de parrainage par d'anciens développeurs.

OUTIL **Système de suivi de bogues**

Le système de suivi de bogues *Debian Bug Tracking System (Debian BTS)* encadre le projet. Son interface web, partie émergée, permet de consulter tous les bogues répertoriés, et propose d'afficher une liste (triée) de bogues sélectionnés sur de nombreux critères : paquet concerné, sévérité, statut, adresse du rapporteur, adresse du mainteneur concerné, étiquette ou *tag*, etc.). Il est encore possible de consulter l'historique complet et toutes les discussions se rapportant à chacun des bogues.

Sous la surface, Debian BTS communique par courrier électronique : toutes les informations qu'il stocke proviennent de messages émis par les différents acteurs concernés. Tout courrier envoyé à 12345@bugs.debian.org sera ainsi consigné dans l'historique du bogue 12345. Les personnes habilitées pourront « fermer » ce bogue en écrivant à 12345-done@bugs.debian.org un message exposant les motifs de cette décision (un bogue est fermé lorsque le problème signalé est corrigé ou plus valide). On signalera un nouveau bogue en transmettant à submit@bugs.debian.org un rapport respectant un format précis, permettant d'identifier le paquet concerné. L'adresse control@bugs.debian.org propose enfin de manipuler toutes les « méta-informations » relatives à un bogue.

Debian BTS offre bien d'autres fonctionnalités (notamment les *tags*, ou étiquettes) je vous invite à les découvrir en lisant sa documentation en ligne :

▸ http://www.debian.org/Bugs/index.fr.html

Le rôle actif des utilisateurs

Est-il pertinent de citer les utilisateurs parmi les acteurs du fonctionnement de Debian ? Oui : ils y jouent un rôle crucial. Loin d'être « passifs », certains de nos utilisateurs utilisent quotidiennement les versions de développement et nous envoient régulièrement des rapports de bogues signalant des problèmes. D'autres vont encore plus loin et formulent des améliorations (par l'intermédiaire d'un bogue de « sévérité » *wishlist* — littéralement « liste de vœux »), voire soumettent directement des correctifs du code source (patches).

VOCABULAIRE **Sévérité d'un bogue**

La « sévérité » (gravité ; *severity* en anglais) d'un bogue décrit de manière formelle la gravité du problème signalé. Tous n'ont en effet pas la même importance : une faute de frappe dans un manuel n'a rien de comparable à une faille de sécurité dans un logiciel serveur.

Debian utilise une échelle étendue de sévérités permettant d'exprimer assez finement la gravité d'un bogue. Elle définit par ailleurs très précisément chacun de ces niveaux afin de faciliter le choix de l'un ou l'autre.

▸ http://www.debian.org/Bugs/Developer.fr.html#severities

Par ailleurs, de nombreux utilisateurs satisfaits du service offert par Debian souhaitent à leur tour apporter une pierre à l'édifice. Pas toujours pourvus des compétences de programmation adéquates, ils choisissent parfois d'aider à la traduction de documents et aux relectures de celles-ci. Pour les francophones, tout ce travail est coordonné sur la liste debian-l10n-french@lists.debian.org.

▸ http://www.debian.org/intl/french/index.fr.html

B.A.-BA **i18n et l10n, qu'es aquò ?**

« i18n » et « l10n » sont les abréviations respectives des mots « internationalisation » et « localisation », dont elles ne conservent que l'initiale, la finale, et le nombre de lettres intermédiaires.

« Internationaliser » un logiciel consiste à le modifier pour qu'il puisse être traduit (localisé). Il s'agit de réécrire partiellement un programme prévu pour fonctionner dans une seule langue pour l'ouvrir à toutes les langues.

« Localiser » un programme consiste à en traduire les messages originels (souvent en anglais) dans une autre langue. Pour cela, il devra avoir été internationalisé.

En résumé, l'internationalisation met en place la structure nécessaire aux traductions, réalisées à proprement parler par la localisation.

OUTIL **Signaler un bogue avec `reportbug`**

L'outil **`reportbug`** facilite l'envoi d'un rapport de bogue sur un paquet Debian. Il peut vérifier au préalable que le bogue concerné n'a pas déjà été référencé, ce qui évite la création de doublons. Il rappelle la définition de la sévérité pour qu'elle soit aussi juste que possible (le développeur pourra toujours affiner par la suite le jugement de l'utilisateur). Il permet d'écrire un rapport de bogue complet sans en connaître la syntaxe précise, en l'écrivant puis en proposant de le retoucher. Ce rapport sera ensuite transmis via un serveur de courrier électronique (local par défaut, mais **`reportbug`** peut aussi utiliser un serveur distant).
Cet outil cible d'abord les versions de développement, seules concernées par les corrections de bogues. Une version stable de Debian est en effet figée dans le marbre, à l'exception des mises à jour de sécurité ou très importantes (si un par exemple un paquet n'est pas du tout fonctionnel). Une correction d'un bogue bénin dans un paquet Debian devra donc attendre la version stable suivante.

Tous ces mécanismes sont accentués par le comportement des utilisateurs. Loin d'être isolés, ils forment une vraie communauté, au sein de laquelle de nombreux échanges ont lieu. Citons notamment l'activité impressionnante de la liste de discussion des utilisateurs francophones debian-user-french@lists.debian.org (le chapitre 7 vous révélera plus d'informations sur cette dernière).

Non contents de s'entraider sur des problèmes techniques qui les concernent directement, ceux-ci traitent aussi de la meilleure manière d'aider Debian et de faire progresser le projet — discussions provoquant souvent des suggestions d'amélioration.

Debian ne finançant aucune campagne publicitaire d'auto-promotion, ses utilisateurs jouent un rôle essentiel dans sa diffusion, et en assurent la notoriété par le bouche à oreille.

Cette méthode fonctionne plutôt bien puisqu'on retrouve des inconditionnels de Debian à tous les niveaux de la communauté du logiciel libre : dans les *install parties* organisées par les groupes locaux d'utilisateurs de Linux, sur les stands associatifs des grands salons d'informatique traitant de Linux, etc.

B.A.-BA **Patch, le moyen d'envoyer un correctif**

Un patch est un fichier décrivant des changements à apporter à un ou plusieurs fichiers de référence. Concrètement, on y trouve une liste de lignes à supprimer ou à insérer, ainsi (parfois) que des lignes reprises du texte de référence et replaçant les modifications dans leur contexte (elles permettront d'en identifier l'emplacement si les numéros de lignes ont changé).
On utilise indifféremment les termes « correctif » et « patch » car la plupart des corrections de bogues sont envoyées sous forme de patch. L'utilitaire appliquant les modifications données par un tel fichier s'appelle simplement **`patch`**. L'outil qui le crée s'appelle **`diff`** (autre synonyme de « correctif ») et s'utilise comme suit :

```
$ diff -u file.old file.new >file.patch
```

Le fichier file.patch contient les instructions permettant de transformer le contenu de file.old en celui de file.new. On pourra le transmettre à un correspondant pour qu'il recrée file.new à partir des deux autres comme ci-dessous :

```
$ patch -p0 file.old <file.patch
```

Le fichier file.old est maintenant identique à file.new.

Signalons que des volontaires réalisent affiches, tracts et autres supports utiles pour la promotion du projet, qu'ils mettent à disposition de tous et que Debian fournit librement sur son site web :

▸ http://www.debian.org/events/material.fr.html

Équipes et sous-projets

Debian s'organisa d'emblée autour du concept de paquet source, chacun correspondant à un mainteneur. De nombreuses équipes de travail sont peu à peu apparues, assurant l'administration de l'infrastructure, la gestion des tâches transversales à tous les paquets (assurance qualité, charte Debian, programme d'installation...), les dernières s'articulant autour de sous-projets.

Sous-projets Debian existants

À chaque public sa Debian ! Un sous-projet est un regroupement de volontaires intéressés par l'adaptation de Debian à des besoins spécifiques. Au-delà de la sélection d'un sous-ensemble de logiciels dédiés à un usage particulier (éducation, médecine, création multimédia...), cela suppose d'améliorer les paquets existants, de mettre en paquet les logiciels manquants, d'adapter l'installateur, de créer une documentation spécifique, etc.

Voici une petite sélection des sous-projets actuels :

- Debian-Junior de Ben Armstrong, vise à proposer aux enfants un système Debian facile et attrayant ;
- Debian-Edu, de *Petter Reinholdtsen*, se focalise sur la création d'une distribution spécialisée pour le monde éducatif ;
- Debian-Med d'*Andreas Tille* se consacre au milieu médical ;
- Debian-Multimedia des créateurs d'Agnula traite de création multimédia ;
- Debian-Desktop de Colin Walters s'intéresse à la bureautique ;
- Debian-Ham, créé par Bruce Perens, cible les radio-amateurs ;
- Debian-NP (comme *Non-Profit*) concerne les associations à but non lucratif ;
- Debian-Lex enfin, travaille pour le cadre des cabinets juridiques.

Gageons que cette liste s'étoffera avec le temps et une meilleure perception des avantages des sous-projets Debian.

PERSPECTIVE **Debian en milieu scolaire**

Debian-Edu est à l'origine un projet francophone réalisé par Stéphane Casset et moi-même au sein de la société Logidée, pour le compte d'un centre départemental de documentation pédagogique. Je l'ai ensuite intégré à Debian en tant que sous-projet. Le temps manquant, il n'a plus progressé, comme c'est parfois le cas des logiciels libres dépourvus de contributeurs.
Parallèlement, une équipe de Norvégiens travaillait sur une distribution similaire, également basée sur **`debian-installer`**. Les progrès de SkoleLinux étant significatifs, je leur ai proposé de s'intégrer à Debian et de reprendre le flambeau de Debian-Edu.

PERSPECTIVE **Debian pour le multimédia**

Agnula est un projet européen mené sous la direction d'une équipe italienne. Il consiste, pour sa partie « DeMuDi », à développer une version de Debian dédiée aux applications multimédia. Certains membres du projet, et notamment Marco Trevisani, ont voulu le pérenniser en l'intégrant dans Debian. Le sous-projet *debian-multimedia* était né.

▸ http://www.agnula.org

Équipes administratives

La plupart des équipes administratives sont relativement fermées et ne recrutent que par cooptation. Le meilleur moyen d'y entrer est alors d'en aider intelligemment les membres actuels en montrant que l'on a compris leurs objectifs et leur mode de fonctionnement.

Les *ftpmasters* sont les responsables de l'archive de paquets Debian. Ils maintiennent le programme qui reçoit les paquets envoyés par les développeurs et les installe automatiquement, après quelques vérifications, sur le serveur de référence (ftp-master.debian.org).

Ils doivent aussi vérifier la licence des nouveaux paquets, pour savoir si Debian peut les distribuer, avant de les intégrer au corpus de paquets existants. Lorsqu'un développeur souhaite supprimer un paquet, c'est à eux qu'il s'adresse via le système de suivi de bogues et le « pseudo-paquet » *ftp.debian.org*.

VOCABULAIRE **Le pseudo-paquet, un outil de suivi**

Le système de suivi de bogues, initialement conçu pour associer des rapports de bogue à un paquet Debian, s'avère très pratique pour gérer d'autres cas de figure : liste de problèmes à résoudre ou de tâches à mener indépendamment de tout lien à un paquet Debian. Les « pseudo-paquets » permettent ainsi à certaines équipes d'utiliser le système de suivi de bogues sans y associer de paquet réel. Tout le monde peut ainsi leur signaler des éléments à traiter. Le BTS dispose ainsi d'une entrée *ftp.debian.org* pour signaler les problèmes de l'archive de paquets ou simplement y demander la suppression d'un paquet. Le pseudo-paquet *www.debian.org* correspond ainsi aux erreurs sur le site web de Debian, et *lists.debian.org* rassemble les soucis liés aux listes de diffusion.

L'équipe *debian-admin* (debian-admin@lists.debian.org), comme on peut s'y attendre, est responsable de l'administration système des nombreux serveurs exploités par le projet. Elle veille au fonctionnement optimal de l'ensemble des services de base (DNS, Web, courrier électronique, shell, CVS, etc.), installe les logiciels demandés par les développeurs Debian, et prend toutes les précautions en matière de sécurité.

Les *listmasters* administrent le serveur de courrier électronique gérant les listes de diffusion. Ils créent les nouvelles listes, gèrent les *bounces* (notices de non livraison), et maintiennent des filtres contre le *spam* (pourriel, ou publicités non sollicitées).

Chaque service spécifique dispose de sa propre équipe d'administration système, constituée généralement par les volontaires qui l'ont mise en place (et, souvent, programmé eux-mêmes les outils correspondants). C'est le cas du système de

CULTURE **Le trafic sur les listes de diffusion : quelques chiffres**

Les listes de diffusion sont de mon point de vue le meilleur témoin de l'activité d'un projet, car elles gardent la trace de tout ce qui s'y passe. Voici quelques statistiques concernant nos listes de diffusion, et qui parleront d'elles-mêmes : Debian héberge plus de 160 listes totalisant 170 000 abonnements individuels. Les 36 000 messages écrits tous les mois provoquent chaque jour l'envoi de 1,4 million de courriers électroniques.

OUTIL **GForge, le couteau suisse du développement collaboratif**

GForge est un logiciel permettant de créer des sites similaires à www.sourceforge.net, alioth.debian.org ou encore savannah.gnu.org. Il s'agit d'héberger des projets et de leur proposer un ensemble de services facilitant le développement collaboratif. Chaque projet dispose alors d'un espace virtuel dédié, regroupant un site web, un système de suivi de bogues, un système de suivi de patches, un outil de sondages, un espace de dépôt de fichiers, des forums, des archives CVS, des listes de diffusion et divers petits services annexes.

alioth.debian.org est le serveur GForge de Debian, administré par Wichert Akkerman, Roland Mas et moi-même. Tout projet impliquant un ou plusieurs développeurs Debian peut y être hébergé.

▸ http://alioth.debian.org

Très complexe de par l'étendue des services qu'il offre, GForge est désormais relativement facile à installer grâce au travail exceptionnel de Roland Mas et Christian Bayle sur le paquet Debian **`gforge`**.

suivi de bogues (BTS), du système de suivi de paquets (*Package Tracking System* — PTS), d'alioth.debian.org (serveur GForge, voir encadré), des services disponibles sur qa.debian.org, lintian.debian.org, arch.debian.org, subversion.debian.org, cdimage.debian.org, etc.

Équipes de développement, équipes transversales

Contrairement aux équipes administratives, les équipes de développement sont très largement ouvertes, même aux contributeurs extérieurs. Même si Debian n'a pas vocation à créer des logiciels, le projet a besoin de quelques programmes spécifiques pour atteindre ses objectifs. Évidemment développés sous une licence libre, ces outils font appel aux méthodes éprouvées par ailleurs dans le monde du logiciel libre.

Debian a développé peu de logiciels en propre, mais certains ont acquis un rôle capital, et leur notoriété dépasse désormais le cadre du projet. Citons notamment **`dpkg`**, programme de manipulation des paquets Debian (c'est d'ailleurs une abréviation de *Debian PacKaGe*), et **`apt`**, outil d'installation automatique de tout paquet Debian et de ses dépendances, garantissant la cohérence du système après la mise à jour (c'est l'acronyme d'*Advanced Package Tool*). Leurs équipes sont pourtant très réduites, car un très bon niveau en programmation est nécessaire à la compréhension globale du fonctionnement de ce type de programmes.

L'équipe la plus importante est probablement celle du nouveau programme d'installation de Debian, **`debian-installer`**, qui a accompli un travail titanesque en 3 ans. Il lui a fallu recourir à de nombreux contributeurs car il est difficile d'écrire un seul logiciel capable d'installer Debian sur une douzaine d'architectures différentes. Chacune a son propre mécanisme de démarrage et son propre

OUTIL **Système de suivi de paquets**

C'est l'une de mes réalisations. L'idée de base est de rassembler sur une seule page le maximum d'informations relatives à chaque paquet source. On peut ainsi visualiser rapidement l'état du logiciel, identifier les tâches à réaliser, et proposer son aide. C'est pourquoi cette page réunit en vrac les statistiques des bogues, les versions disponibles dans chaque distribution, la progression du paquet dans la distribution *testing*, l'état des traductions des descriptions et des *templates debconf*, l'éventuelle disponibilité d'une nouvelle version amont, des avertissements en cas de non conformité à la dernière version de la charte Debian, des renseignements sur le mainteneur, et toute autre information que celui-ci aura souhaité y intégrer.

▸ http://packages.qa.debian.org

Un système d'abonnement par courrier électronique complète cette interface web. Il envoie automatiquement une sélection d'informations choisies dans la liste suivante : bogues et discussions associées, notices de disponibilité d'une nouvelle version sur les serveurs Debian, traductions effectuées (pour les relire), etc.

Les utilisateurs avancés peuvent donc suivre tout cela de près, voire contribuer au projet après avoir bien compris son fonctionnement.

Igor Genibel a créé une interface web similaire, développée autour des mainteneurs plutôt que des paquets eux-mêmes. Elle donne à chaque développeur un synoptique de l'état de tous les paquets Debian placés sous sa responsabilité.

▸ http://qa.debian.org/developer.php

Ces deux sites web constituent des outils pour *Debian QA* (*Quality Assurance*), le groupe en charge de l'assurance qualité au sein de Debian.

chargeur d'amorçage (*bootloader*). Tout ce travail est coordonné sur la liste de diffusion debian-boot@lists.debian.org, sous la houlette de Joey Hess.

- http://www.debian.org/devel/debian-installer/
- http://www.kitenet.net/~joey/blog/entry/d-i_retrospective-2004-08-07-19-46.html

L'équipe du programme **`debian-cd`**, plus réduite, a un objet bien plus modeste. Signalons que de nombreux petits contributeurs se chargent de leur architecture, le développeur principal (votre serviteur) ne pouvant pas en connaître toutes les subtilités, ni la manière exacte de faire démarrer l'installateur depuis le cédérom.

De nombreuses équipes ont des tâches transversales à l'activité de mise en paquet : debian-qa@lists.debian.org essaye par exemple d'assurer la qualité à tous les niveaux de Debian. Quant à debian-policy@lists.debian.org, elle fait évoluer la charte Debian en fonction des propositions des uns et des autres. Les équipes responsables de chaque architecture (debian-*arch*@lists.debian.org) y compilent tous les paquets, qu'elles adaptent à leur architecture le cas échéant.

D'autres équipes encadrent les paquets les plus importants pour en assurer la maintenance sans faire peser une trop lourde responsabilité sur une seule paire d'épaules ; c'est le cas de la bibliothèque C avec debian-glibc@lists.debian.org, du compilateur C avec debian-gcc@lists.debian.org ou encore de XFree86 avec debian-x@lists.debian.org (groupe également connu sous le nom de *X Strike Force*, dirigé par Branden Robinson et Fabio Massimo Di Nitto).

CULTURE **CVS**

CVS (*Concurrent Versions System*) est un outil pour travailler simultanément à plusieurs sur des fichiers en conservant un historique des modifications. Il s'agit en général de fichiers texte, comme le code source d'un logiciel. Si plusieurs personnes travaillent de concert sur le même fichier, **`cvs`** ne pourra fusionner les modifications effectuées que si elles ont porté sur des portions distinctes du texte. Dans le cas contraire, il faudra résoudre ces « conflits » à la main. Ce système gère les modifications ligne par ligne en stockant des correctifs différentiels de type *diff* d'une « révision » (version) à l'autre.

CVS utilise une archive centrale (« dépôt » appelé *CVS repository* en anglais), stockant les fichiers et l'historique de leurs modifications (chaque révision est enregistrée sous la forme d'un fichier correctif de type *diff*, prévu pour être appliqué sur la version précédente). Chacun en extrait une version particulière (*working copy* ou « copie de travail ») pour travailler. L'outil permet notamment de consulter les modifications effectuées sur sa copie de travail (**`cvs diff`**), de les enregistrer dans l'archive centrale en créant une nouvelle entrée dans l'historique des versions (**`cvs commit`**), de mettre à jour sa copie de travail pour intégrer les modifications effectuées en parallèle par d'autres utilisateurs (**`cvs update`**), et d'enregistrer dans l'historique une configuration particulière afin de pouvoir facilement l'extraire plus tard (**`cvs tag`**). Les experts de **`cvs`** sauront mener de front plusieurs versions d'un projet en développement sans qu'elles n'interfèrent. Le terme consacré est *branches*. Cette métaphore de l'arbre est assez juste, car il s'agit d'abord de développer un programme sur un tronc commun. Parvenu à une étape importante (comme la version 1.0), le développement continue sur deux branches : la branche de développement prépare la version majeure suivante et la branche de maintenance gère les mises à jour corrigeant la version 1.0.

`cvs` souffre pourtant de quelques limitations. Incapable de gérer les liens symboliques, les changements de noms de fichiers ou de répertoires, la suppression de répertoires, etc. il a contribué à l'apparition de concurrents libres et plus modernes, corrigeant la plupart de ces défauts. Citons notamment **`subversion`** et **`arch`**.

- http://subversion.tigris.org/
- http://www.gnu.org/software/gnu-arch/

Rôle d'une distribution

Une distribution GNU/Linux a deux objectifs principaux : installer un système libre sur un ordinateur (vierge ou disposant déjà d'autres systèmes) et fournir une palette de logiciels couvrant tous les besoins de l'utilisateur.

L'installateur : `debian-installer`

`debian-installer`, conçu de manière très modulaire pour être le plus générique possible, répond au premier. Il couvre un grand nombre de scénarios d'installations et surtout facilite grandement la création d'un installateur dérivé correspondant à un cas particulier.

Cette modularité, qui le rend aussi plus complexe, pourra perturber les développeurs découvrant cet outil. Il déroutera encore les utilisateurs habitués à des installations plus graphiques et moins souples. De gros efforts ont pourtant été consentis pour leur éviter de devoir renseigner un maximum de champs — ce qui explique l'intégration d'un logiciel de détection automatique du matériel (**`discover`** de Progeny Inc.).

Il est intéressant de remarquer que les distributions dérivées de Debian se différencient beaucoup sur cet aspect, et fournissent un installateur plus limité (souvent confiné à l'architecture i386) mais bien plus convivial aux yeux des utilisateurs néophytes. En revanche, elles se gardent généralement de trop diverger sur les contenus des paquets pour profiter au maximum de la grande famille de logiciels proposés sans souffrir de problèmes de compatibilité.

NOTE **Cédérom fourni avec le livre**

Le cédérom joint à ce livre exploite **`debian-installer`**. Il permet d'installer Debian 3.1 « *release candidate 3* » simplement en démarrant l'ordinateur dessus. Il contient en outre la plupart des logiciels étudiés dans ce livre.

La bibliothèque de logiciels

Quantitativement, Debian est indiscutablement en tête avec plus de 8000 paquets sources. Qualitativement, sa charte et la longue période de tests préalable à toute version stable justifient sa réputation de cohérence et de stabilité. Sur le plan de la disponibilité, on trouve tout en ligne sur de nombreux miroirs mis à jour quotidiennement.

De nombreux commerçants vendent sur le Web des cédéroms à bas prix (parfois à prix coûtant), dont chacun est libre de télécharger et graver les « images ». Seule ombre au tableau : la faible fréquence de sortie des versions stables (leur élaboration dépasse parfois deux ans), qui ralentit l'intégration de tout nouveau logiciel.

La plupart des nouveaux logiciels libres sont rapidement pris en charge dans la version de développement, qui permet de les installer. Si cela implique trop de mises à jour par le jeu des dépendances, on peut aussi recompiler le programme pour la version stable de Debian (voir le chapitre 13 pour plus de détails sur le sujet).

Cycle de vie d'une *release*

Le projet dispose à tout instant de 3 ou 4 versions différentes de chaque logiciel, nommées *experimental*, *unstable*, *testing*, et *stable*. Chacune évoque un stade différent du développement. Pour bien les comprendre, suivons le parcours d'un programme, de sa première mise en paquet à son intégration dans une version stable de Debian.

> VOCABULAIRE **Release**
>
> Le terme « *release* » désigne chez Debian une version particulière d'une distribution (ex : « *the unstable release* » signifie « la version instable »). Il désigne aussi l'annonce publique de toute nouvelle version (stable).

Le statut *experimental*

Traitons d'abord le cas particulier de la distribution *experimental* : c'est un ensemble de paquets Debian correspondant à des logiciels en cours de développement, et pas forcément finalisés — d'où son nom. Tout ne transite pas par cette étape ; certains développeurs y créent des paquets pour obtenir un premier retour des utilisateurs les plus expérimentés (ou les plus courageux).

D'autre part, cette distribution abrite fréquemment des modifications importantes portant sur des paquets de base et dont l'intégration dans *unstable* avec des bogues gênants aurait des répercussions trop importantes et bloquantes. C'est donc une distribution totalement isolée, dont les paquets ne migrent jamais vers une autre (sauf intervention expresse du mainteneur ou des *ftpmasters*).

Le statut *unstable*

Revenons au cas d'un paquet type. Le mainteneur crée un premier paquet, qu'il compile pour *unstable* et place sur le serveur ftp-master.debian.org. Cette première manifestation implique inspection et validation par les *ftpmasters*. Le logiciel est alors disponible dans *unstable*, distribution risquée mais choisie par des utilisateurs préférant le dernier cri à l'assurance de l'absence de bogues graves. Ceux-ci découvrent alors le programme et le testent.

S'ils découvrent des bogues gênants dans ce paquet pas encore décanté, ils les décrivent à son mainteneur. Ce dernier prépare alors régulièrement des versions corrigées, qu'il place sur le serveur.

Toute nouvelle mise en ligne est répercutée sur tous les miroirs Debian du monde dans les 24 heures. Les utilisateurs valident alors la correction et cherchent d'autres problèmes, consécutifs aux modifications. Plusieurs mises à jour peuvent ainsi s'enchaîner rapidement. Pendant ce temps, les robots *autobuilders* sont entrés en action. Le plus souvent, le mainteneur ne dispose que d'un PC traditionnel et aura compilé son paquet pour architecture i386 ; les *autobuilders* ont donc pris le

relais et compilé automatiquement des versions pour toutes les autres architectures. Certaines compilations pourront échouer ; le mainteneur recevra alors un rapport de bogue signalant le problème, à corriger dans les prochaines versions. Lorsque le bogue est découvert par un spécialiste de l'architecture concernée, il arrive que ce rapport soit accompagné d'un correctif prêt à l'emploi.

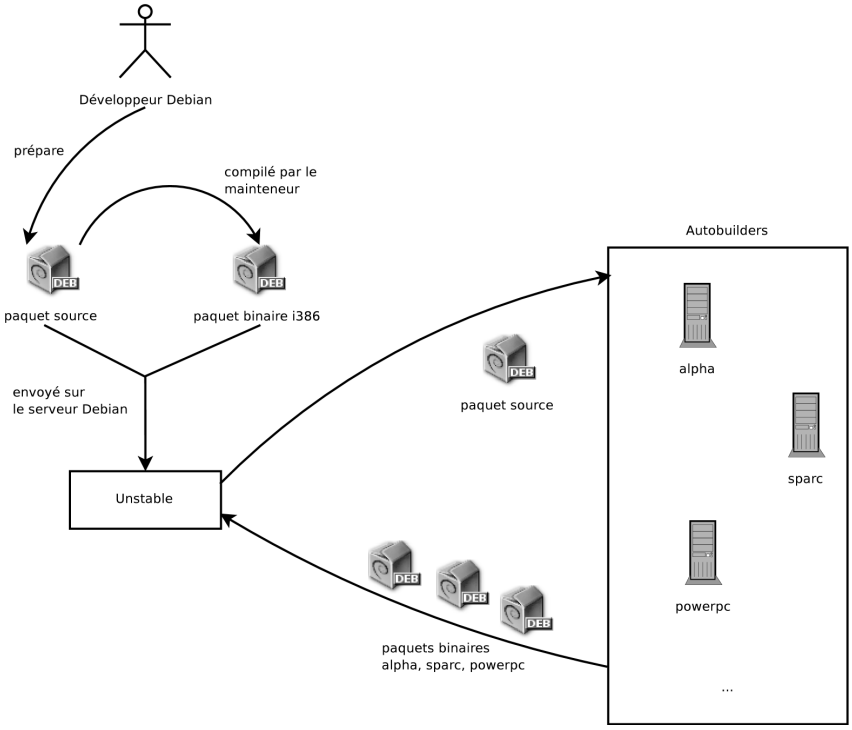


Figure 1–2 Compilation d'un paquet par les autobuilders

DÉCOUVERTE **`buildd`, le recompilateur de paquet Debian**

buildd est l'abréviation de *build daemon*. Ce logiciel recompile automatiquement les nouvelles versions des paquets Debian sur l'architecture qui l'accueille (la compilation croisée — *crosscompiling* — n'étant pas toujours satisfaisante).

Ainsi pour produire des binaires destinés à l'architecture `sparc`, le projet dispose de machines `sparc` (en l'occurrence de marque Sun). Le programme *buildd* y fonctionne en permanence afin de créer des paquets binaires pour `sparc` à partir des paquets sources expédiés par les développeurs Debian.

Ce logiciel est employé sur tous les ordinateurs servant d'*autobuilders* à Debian. Par extension, le terme *buildd* désigne souvent ces machines, en général réservées à cet usage.

NOTE **Limitations de testing**

Très intéressant dans son principe, *testing* pose quelques problèmes pratiques : l'enchevêtrement des dépendances croisées entre paquets est tel que jamais un paquet ne peut y progresser tout seul. Les paquets dépendant tous les uns des autres, il est nécessaire d'y faire progresser simultanément un grand nombre d'entre eux, ce qui est impossible tant que certains subissent des mises à jour régulières. D'autre part, le script identifiant les familles de paquets ainsi solidarisés peine beaucoup à les constituer (il s'agirait d'un problème NP-complet, auquel nous connaissons heureusement quelques bonnes heuristiques). C'est pourquoi on peut intervenir manuellement et conseiller ce script en lui suggérant des ensembles de paquets ou en imposant l'inclusion de certains d'entre eux — quitte à casser temporairement quelques dépendances. Cette fonctionnalité est accessible au *Release Manager* et à ses assistants.
Rappelons qu'un problème NP-complet est de complexité algorithmique exponentielle avec le nombre d'éléments concernés. La seule manière de le résoudre est souvent d'examiner toutes les configurations possibles, ce qui requiert parfois d'énormes moyens. Une heuristique en est une solution approchée et satisfaisante.

La migration vers *testing*

Un peu plus tard, le paquet aura mûri ; compilé sur toutes les architectures, il n'aura pas connu de modifications récentes. C'est alors un candidat pour l'intégration dans la distribution *testing* — ensemble de paquets *unstable* sélectionnés sur quelques critères quantifiables. Chaque jour, un programme choisit automatiquement les paquets à intégrer à *testing*, selon des éléments garantissant une certaine qualité :

1. absence de bogues critiques, ou tout du moins nombre inférieur à celui de la version actuellement intégrée dans *testing* ;
2. villégiature minimale de 10 jours dans *unstable*, ce qui laisse assez de temps pour trouver et signaler les problèmes graves ;
3. compilation réussie sur toutes les architectures officiellement prises en charge ;
4. dépendances toutes satisfaisables dans *testing*, ou qui peuvent du moins y progresser de concert avec le paquet ;

Ce système n'est évidemment pas infaillible ; il arrive régulièrement qu'on trouve des bogues critiques dans un paquet intégré à *testing*. Il est pourtant globalement efficace, et *testing* pose beaucoup moins de problèmes qu'*unstable*, représentant pour beaucoup un bon compromis entre la stabilité et la soif de nouveauté.

La promotion de *testing* en *stable*

Supposons notre paquet désormais intégré à *testing*. Tant qu'il est perfectible, son responsable doit persister à l'améliorer et recommencer le processus depuis *unstable* (mais ces inclusions ultérieures dans *testing* sont en général plus rapides : si elles n'ont pas trop évolué, toutes les dépendances sont déjà présentes). Quand il atteint la perfection, son mainteneur a fini son travail, et la prochaine étape est l'inclusion dans la distribution *stable*, en réalité une simple copie de *testing* à un moment choisi par le *Release Manager*. L'idéal est de prendre cette décision quand

COMMUNAUTÉ **Le Release Manager**

Release Manager (gestionnaire de version) est un titre important, associé à de lourdes responsabilités. Son porteur doit en effet gérer la sortie de la nouvelle version stable de Debian et définir le processus d'évolution de *testing* tant qu'elle ne répond pas aux critères de qualité de *stable*. Il définit également un calendrier prévisionnel (jamais respecté).
Colin Watson et Steve Langasek partagent cette responsabilité depuis août 2004. Anthony Towns l'avait auparavant assumée plusieurs années, après avoir programmé les scripts de gestion de *testing*.
On trouve aussi un *Stable Release Manager* (gestionnaire de version stable) : c'est lui qui gère et sélectionne les mises à jour de la version stable de Debian. Il y inclut systématiquement les correctifs de sécurité et examine au cas par cas toutes les autres propositions d'inclusion faites par des développeurs Debian soucieux de mettre à jour un de leurs paquets dans la version stable. Cette personne est actuellement Martin Schulze.

l'installateur est prêt et quand plus aucun programme de *testing* n'a de bogue critique répertorié.

Étant donné que ce moment ne survient jamais dans la pratique, Debian doit faire des compromis : supprimer des paquets dont le mainteneur n'a pas réussi à corriger les bogues à temps ou accepter de livrer une distribution comptant quelques bogues pour des milliers de logiciels. Le *Release Manager* aura préalablement prononcé une période de *freeze* (gel), où il devra approuver chaque mise à jour de *testing*. Le but est d'empêcher toute nouvelle version (et ses nouveaux bogues) et de n'approuver que des mises à jours correctives.

VOCABULAIRE **Freeze : la dernière ligne droite**

Pendant la période de *freeze* (« gel »), l'évolution du contenu de la distribution *testing* est bloquée : plus aucune mise à jour automatique n'a lieu. Seul le *Release Manager* est alors habilité à y changer des paquets, selon ses propres critères. L'objectif est d'éviter l'apparition de nouveaux bogues par l'introduction de nouvelles versions ; seules les mises à jour bien examinées sont acceptées lorsqu'elles corrigent des bogues importants.

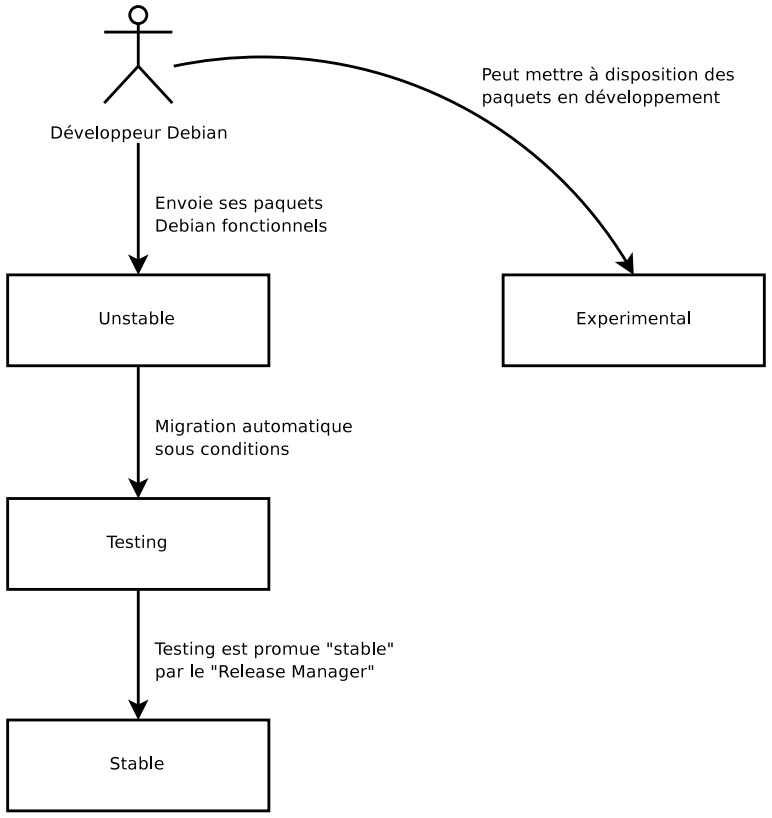


Figure 1–3 Parcours d'un paquet au sein des différentes versions de Debian

Après la sortie de la nouvelle version stable, le *Stable Release Manager* en gère les évolutions ultérieures (appelées « révisions ». ex : 3.0r1, 3.0r2, 3.0r3 pour la version 3.0). Ces mises à jour intègrent systématiquement tous les correctifs de sécurité. On y trouve également les corrections les plus importantes (le mainteneur du paquet doit prouver la gravité du problème qu'il souhaite corriger pour faire intégrer sa mise à jour).

Fin du voyage : notre hypothétique paquet est désormais intégré à la distribution stable. Ce trajet, non dépourvu de difficultés, explique les délais importants séparant les versions stables de Debian. Il contribue surtout à sa réputation de qualité. De plus, la majorité des utilisateurs est satisfaite par l'emploi de l'une des trois distributions disponibles en parallèle. Les administrateurs système, soucieux avant tout de la stabilité de leurs serveurs, se moquent de la dernière version de GNOME ; ils opteront pour Debian *stable* et en seront satisfaits. Les utilisateurs finaux, plus intéressés par la dernière version de GNOME ou de KDE que par une stabilité irréprochable, trouveront en Debian *testing* un bon compromis entre absence de problèmes graves et logiciels relativement à jour. Enfin, les développeurs et utilisateurs les plus expérimentés pourront ouvrir la voie en testant toutes les nouveautés de Debian *unstable* dès leur sortie, au risque de subir les affres et bogues inhérents à toute nouvelle version de logiciel. À chaque public sa Debian !

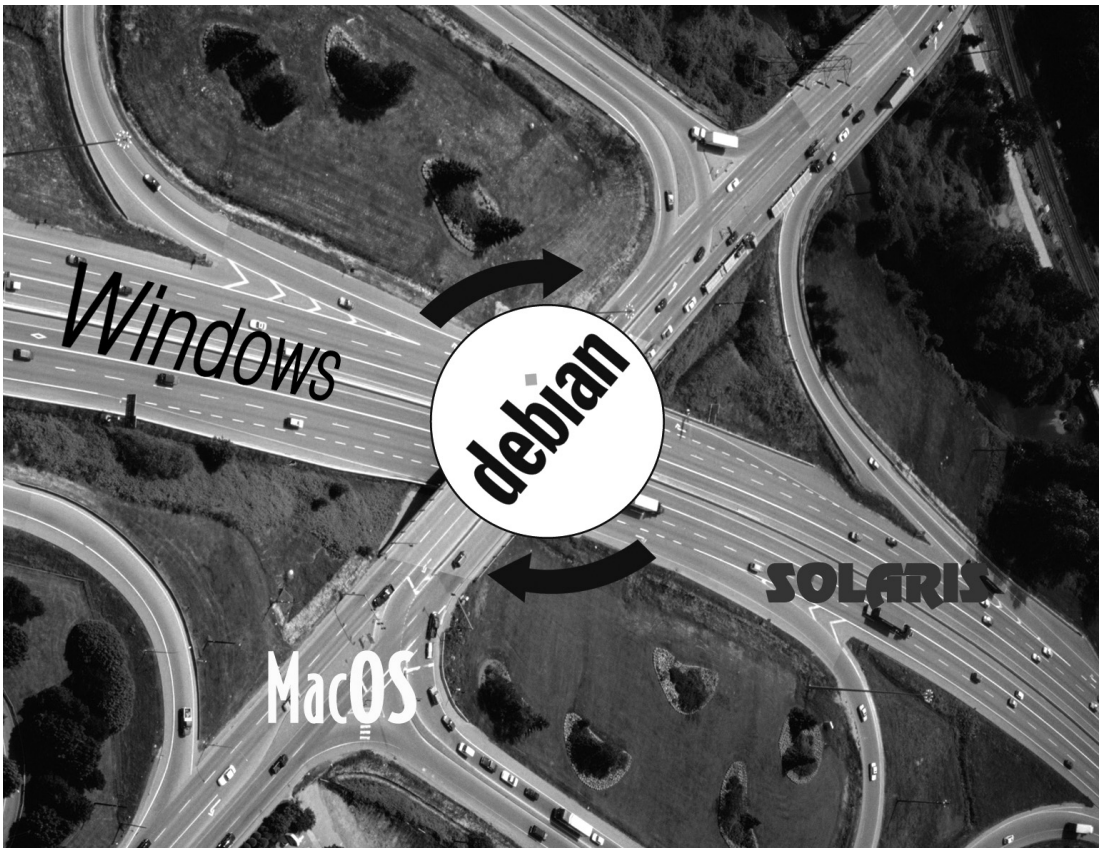
CULTURE GNOME et KDE, les bureaux graphiques

GNOME (« *GNU Network Object Model Environment* », ou environnement réseau de modèle objet de GNU) et KDE (« *K Desktop Environment* », ou environnement de bureau K) sont les deux « bureaux graphiques » les plus populaires dans le milieu du logiciel libre. On entend par là un ensemble de logiciels de bureautique permettant d'effectuer aisément les opérations les plus courantes au travers d'une interface graphique. Ils comportent notamment un gestionnaire de fichiers, une suite bureautique, un navigateur web, un logiciel de courrier électronique, des accessoires multimédias, etc. Leur différence la plus visible réside dans le choix de la bibliothèque graphique employée : GNOME a choisi GTK+ (logiciel libre sous licence LGPL) et KDE a opté pour Qt (de la société Trolltech, libre sous les systèmes d'exploitation libres mais pas sous Microsoft Windows).

- http://www.gnome.org
- http://www.kde.org



2



Présentation de l'étude de cas

Vous êtes administrateur système d'une PME en pleine croissance. En collaboration avec votre direction, vous venez de redéfinir le plan directeur du système informatique pour l'année qui vient, et avez choisi de migrer progressivement vers Debian pour des raisons tant pratiques qu'économiques. Détaillons ce qui vous attend...

SOMMAIRE

- Des besoins informatiques en forte hausse
- Plan directeur
- Pourquoi une distribution GNU/Linux ?
- Pourquoi la distribution Debian ?
  - Distributions communautaires et commerciales
- Pourquoi Debian Sarge ?

MOTS-CLEFS

- Falcot SA
- PME
- Forte croissance
- Plan directeur
- Migration
- Réduction des coûts

NOTE **Société fictive de l'étude de cas**

La société Falcot SA étudiée ici est totalement fictive. Toute ressemblance avec une société réelle est purement fortuite. De même, certaines données des exemples parsemant ce livre peuvent être fictives.

J'ai imaginé cette étude de cas pour aborder tous les services d'un système d'information moderne couramment utilisés dans une société de taille moyenne. Après la lecture de ce livre, vous disposerez de tous les éléments nécessaires pour effectuer vos propres installations de serveurs et voler de vos propres ailes. Vous aurez aussi appris comment trouver efficacement des informations en cas de blocage.

Des besoins informatiques en forte hausse

Falcot SA est un fabricant de matériel audio haut de gamme. C'est une PME en forte croissance qui dispose de deux sites : Saint-Étienne et Pau. Le premier compte environ 150 employés ; il héberge l'usine de fabrication des enceintes, un laboratoire de conception, et les bureaux de toute l'administration. Le site de Pau, plus petit, n'abrite qu'une cinquantaine de collaborateurs et produit les amplificateurs.

Le système informatique a peiné à suivre la croissance de l'entreprise et il a été convenu de le redéfinir entièrement pour atteindre différents objectifs fixés par la direction :

- infrastructure moderne capable de monter en puissance facilement ;
- baisse du coût des licences logicielles grâce à l'emploi de logiciels Open Source ;
- mise en place d'un site de commerce électronique, voire de B2B ;
- amélioration importante de la sécurité en vue de mieux protéger les secrets industriels relatifs aux nouveaux produits.

Derrière ces objectifs se dessine une refonte globale du système d'information.

Plan directeur

Avec votre collaboration, la direction informatique a réalisé une étude un peu plus poussée, permettant d'identifier quelques contraintes et de définir un plan de migration vers le système Open Source retenu, Debian.

Parmi les contraintes, il faut noter que la comptabilité utilise un logiciel spécifique ne fonctionnant que sous Microsoft Windows™. Le laboratoire utilise quant à lui un logiciel de conception assistée par ordinateur fonctionnant sous MacOS X™.

Le passage vers Debian sera bien entendu progressif ; une PME, aux moyens limités, ne peut pas tout changer rapidement. Dans un premier temps, c'est le personnel informatique qui doit être formé à l'administration de Debian. Les serveurs seront ensuite basculés, en commençant par l'infrastructure réseau (routeur, pare-feu, etc.) pour enchaîner sur les services utilisateurs (partage de fichiers,

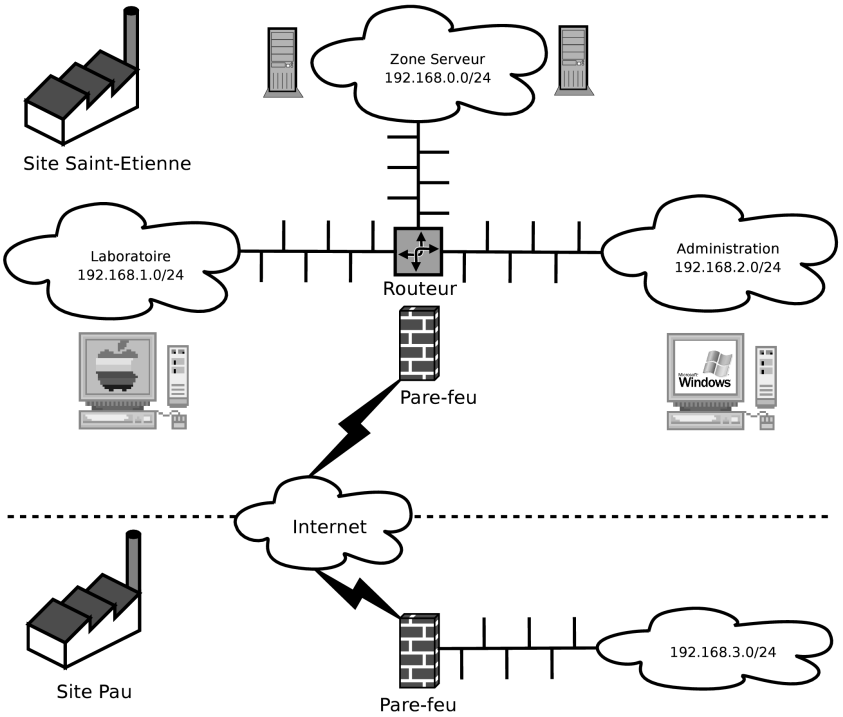


Figure 2–1 Aperçu global du réseau de Falcot SA

Web, SMTP, etc.). Ce n'est qu'ensuite que les ordinateurs de bureau seront progressivement migrés sous Debian, pour que chaque service puisse être formé (en interne) lors du déploiement du nouveau système.

Pourquoi une distribution GNU/Linux ?

Plusieurs facteurs ont dicté ce choix. L'administrateur système, qui connaissait cette distribution, l'a fait inclure dans les candidats à la refonte du système d'information. Des conditions économiques difficiles et une compétition féroce ont limité le budget de cette opération, malgré son importance capitale pour l'avenir de l'entreprise. C'est pourquoi les solutions Open Source ont rapidement séduit : plusieurs études récentes les donnent bien moins onéreuses que les solutions propriétaires malgré une qualité de service équivalente voire supérieure, à condition d'avoir du personnel qualifié pour leur administration.

Parmi les systèmes d'exploitation libres, le service informatique a recensé les BSD libres (dont OpenBSD, FreeBSD et NetBSD), GNU Hurd, et les distributions Linux. GNU Hurd, qui n'a pas encore publié de version stable, fut immédiatement rejeté. Le choix est moins simple entre BSD et Linux. Les premiers sont très

B.A.-BA **Linux ou GNU/Linux ?**

Linux, comme vous le savez déjà, n'est qu'un noyau. Les expressions « distribution Linux » ou « système Linux » sont donc incorrectes : il s'agit en réalité de distributions ou de systèmes *basés sur* Linux. Ces expressions omettent de mentionner les logiciels qui complètent toujours ce noyau, parmi lesquels les programmes développés par le projet GNU. Richard Stallman, créateur de ce dernier, souhaite bien entendu que l'expression « GNU/Linux » soit systématiquement employée, afin que l'importance de la contribution du projet GNU soit mieux reconnue et ses idées de liberté mieux véhiculées.
Debian a choisi de suivre cette recommandation et nomme donc ses distributions en mentionnant GNU (exemple : Debian GNU/Linux 3.1).

méritants, notamment sur les serveurs. Le pragmatisme pousse pourtant à opter pour un système Linux car sa base installée et sa popularité, bien plus importantes, ont de nombreuses conséquences positives. Il est ainsi plus facile de trouver du personnel qualifié pour administrer des machines Linux que des techniciens rompus à BSD. D'autre part, la prise en charge des matériels récents est plus rapide sous Linux que sous BSD (même si les deux se suivent souvent de peu). Enfin, les distributions Linux sont souvent plus adaptées à l'installation de « cliquodromes » graphiques, indispensables aux utilisateurs débutants lors de la migration de toutes les machines de bureau vers ce nouveau système.

Pourquoi la distribution Debian ?

Le choix de Linux entériné, il fallait opter pour une offre précise. À nouveau, les critères à considérer abondent. La distribution retenue doit pouvoir fonctionner plusieurs années, car la migration de l'une à l'autre représente des coûts supplémentaires (moins élevés toutefois que s'il s'agissait d'un système totalement différent, comme Windows ou Mac OS).

La pérennité est donc primordiale, et il faut une garantie d'existence et de publication régulière de correctifs de sécurité pendant plusieurs années. Le calendrier de mises à jour compte lui aussi : avec son important parc informatique, Falcot SA ne peut mener cette opération complexe trop souvent. Le service informatique exige pourtant d'employer la dernière version stable de la distribution, bénéficiant de la meilleure assistance technique, et aux correctifs de sécurité assurés. En effet, les mises à jour de sécurité ne sont généralement assurées que pour une durée limitée sur les anciennes versions d'une distribution.

Enfin, pour des raisons d'homogénéité du parc et de facilité d'administration, il est demandé que la même distribution assure le fonctionnement des serveurs (dont certains sont des machines Sparc actuellement sous Solaris) et des PC de bureau.

EN PRATIQUE **Le coût total de possession (TCO)**

Le *Total Cost of Ownership* (coût total de possession) est la somme d'argent dépensée suite à la possession ou à l'acquisition d'un bien : dans le cas présent, il s'agit de systèmes d'exploitation. Ce prix inclut l'éventuelle licence, la formation du personnel au nouveau logiciel, le changement de toute machine trop peu puissante, les réparations supplémentaires, etc. Tout ce qui découle directement du choix initial est pris en compte.
Ce TCO, qui varie selon les critères retenus dans son évaluation, est rarement significatif par lui-même. En revanche, il est très intéressant de comparer des TCO calculés en suivant les mêmes règles. Cette grille d'appréciation revêt donc une importance capitale, et il est facile de la manipuler pour en tirer une conclusion prédéfinie. Ainsi, le TCO d'une seule machine n'a pas de sens puisque le coût d'un administrateur se répercute sur la quantité totale de postes qu'il encadre, nombre qui dépend évidemment du système d'exploitation et des outils proposés.

Distributions communautaires et commerciales

On trouve deux grandes catégories de distributions Linux : les commerciales et les communautaires. Les premières, développées par des entreprises, sont vendues associées à des services. Les secondes sont élaborées suivant le même modèle de développement ouvert que les logiciels libres dont elles sont constituées.

Une distribution commerciale aura donc tendance à publier de nouvelles versions assez fréquemment pour mieux en commercialiser les mises à jour et services associés. Son avenir est directement lié au succès commercial de son entreprise, et beaucoup ont déjà disparu (Caldera Linux, StormLinux, etc.).

Une distribution communautaire ne suit quant à elle aucun planning. À l'instar du noyau Linux, les nouvelles versions sortent lorsqu'elles sont stables, jamais avant. Sa survie est garantie tant qu'il y aura assez de développeurs individuels ou de sociétés tierces pour la faire vivre.

Une comparaison des diverses distributions Linux a fait retenir Debian pour de nombreuses raisons :

- C'est une distribution communautaire, au développement assuré indépendamment de toute contrainte commerciale ; ses objectifs sont donc essentiellement d'ordre technique, ce qui semble favoriser la qualité globale du produit.
- De toutes les distributions communautaires, c'est la plus importante à tout point de vue : en nombre de contributeurs, en nombre de logiciels disponibles, en années d'existence. La taille de sa communauté représente évidemment un indiscutable gage de pérennité.
- Statistiquement, ses nouvelles versions sortent tous les 18 à 24 mois, calendrier qui convient aux administrateurs.
- Une enquête auprès de plusieurs sociétés de services françaises spécialisées dans le logiciel libre a montré que toutes proposent une assistance technique pour Debian ; c'est même pour beaucoup d'entre elles la distribution retenue en interne. Cette diversité de fournisseurs potentiels est un atout majeur pour l'indépendance de Falcot SA.
- Enfin, Debian est disponible sur une multitude d'architectures, dont Sparc ; il sera donc possible de l'installer sur les quelques serveurs Sun de Falcot SA.

Une fois Debian retenue, il reste à choisir la version à employer. Voyons pourquoi les administrateurs ont tranché en faveur de Debian Sarge.

PARTICIPER

N'hésitez pas à me signaler toute erreur à l'adresse hertzog@debian.org.

Pourquoi Debian Sarge ?

À l'heure où j'écris ces lignes, Debian Sarge est encore la distribution « *testing* », mais la publication de ce livre devrait coïncider plus ou moins avec son officialisation comme nouvelle version « stable » de Debian. C'est d'ailleurs la raison pour laquelle j'évoquerai « Debian Sarge » plutôt que « Debian 3.1 »... l'usage veut que le numéro de version ne soit pas employé avant sa sortie effective.

Vous pourrez ainsi parfois remarquer quelques différences entre ce qui est décrit ici et ce que vous constaterez en pratique — même si j'ai limité ces incohérences de mon mieux.

Le choix de Debian Sarge se justifie bien sûr par le fait que tout administrateur soucieux de la qualité de ses serveurs s'orientera naturellement vers la version stable de Debian. De plus, cette distribution introduit de nombreux changements intéressants : qu'il s'agisse du nouvel installateur ou de la généralisation de **`debconf`**, tous apportent des améliorations qui concernent directement les administrateurs.

COMMUNAUTÉ **Fin de la gestation de Sarge**

À l'heure de cette nouvelle édition, Debian Sarge n'est toujours pas publiée... mais je maintiens ma prévision et sais qu'elle se réalisera. À l'époque de la première édition, l'absence de serveurs capables de recompiler les mises à jour de sécurité pour la distribution Sarge bloquait le processus. Les *Release Managers*, ne sachant pas quand ce problème serait résolu, décidèrent de ne pas fermer les vannes qui alimentent Sarge en nouvelles versions des logiciels. De nombreuses mises à jour progressent donc régulièrement (GNOME 2.8, KDE 3.3, Firefox 1.0, etc.). Dans ces conditions, il est difficile de réduire le nombre de bogues critiques. Par ailleurs, ces mises à jour rendent parfois nécessaire une adaptation de l'installateur, dont la vaste campagne de tests dure trois semaines. Grâce à ces nombreuses évolutions, Sarge s'enrichit de logiciels récents, et la prochaine version stable de Debian ne sera peut-être pas obsolète le jour de sa parution.

En février 2005, le problème des serveurs pour les mises à jour de sécurité est en passe d'être réglé et les *Release Managers* vont à nouveau pouvoir travailler efficacement en gelant progressivement la distribution au fur et à mesure que les bogues sont corrigés. Par ailleurs, la version *release candidate 3* de l'installateur est prête — et cela devrait être une version stable. On croise les doigts !



3



Prise en compte de l'existant et migration

Toute refonte du système d'information doit se baser sur l'existant pour réexploiter au maximum les ressources disponibles et garantir l'interopérabilité des différents éléments constituant le système. Cette étude fera apparaître une trame générique, à suivre dans chaque migration d'un service sous Linux.

SOMMAIRE

- Coexistence en environnement hétérogène
  - Intégration avec des machines Windows
  - Intégration avec des machines Mac OS
  - Intégration avec d'autres machines Linux/Unix
- Démarche de migration
  - Recenser et identifier les services
  - Conserver la configuration
  - Prendre en main un serveur Debian existant
  - Installer Debian
  - Installer et configurer les services sélectionnés

MOTS-CLEFS

- Existant
- Réutilisation
- Migration

Coexistence en environnement hétérogène

Debian s'intègre très bien dans tous les types d'environnements existants et cohabite avec tous les systèmes d'exploitation. Cette quasi-parfaite harmonie provient des pressions du marché qui contraignent les éditeurs à développer des logiciels respectueux des normes et standards, donc avec lesquels les autres programmes, libres ou pas, serveurs comme clients, peuvent interagir.

OUTIL **Samba**

La version 2 de Samba se comporte comme un serveur Windows NT (authentification, fichiers, files d'impression, téléchargement des pilotes d'impression, DFS, etc.). La version 3 honore l'annuaire Active Directory, l'interopérabilité avec les contrôleurs de domaines NT4 et les appels distants d'administration (RPC — *Remote Procedure Call*).

Intégration avec des machines Windows

La prise en charge de SMB/CIFS par Samba assure une communication parfaite dans un contexte Windows. Il sert des fichiers et des files d'impression aux clients Windows et intègre des logiciels grâce auxquels une machine Linux utilisera des ressources publiées par des serveurs Windows.

Intégration avec des machines Mac OS

Le logiciel Netatalk, employant le protocole Appletalk (lui-même géré par le noyau Linux), interfacera Debian avec un réseau Mac OS. Il assure les fonctions de serveur de fichiers et de files d'impression ainsi que de temps (synchronisation des horloges). Sa fonction de routeur permet d'interconnecter des réseaux Appletalk.

Intégration avec d'autres machines Linux/Unix

Enfin, NFS et NIS, eux aussi compris, garantiront les interactions avec des systèmes Unix. NFS assure la fonctionnalité de serveur de fichiers, tandis que NIS permet de créer un annuaire des utilisateurs. Signalons également que la couche d'impression BSD, employée par la majorité des Unix, permet aussi de partager des files d'impression.

Démarche de migration

Pour chaque ordinateur à migrer, il faut suivre une démarche garantissant une continuité dans les services offerts. Quel que soit le système d'exploitation utilisé, le principe ne change pas.

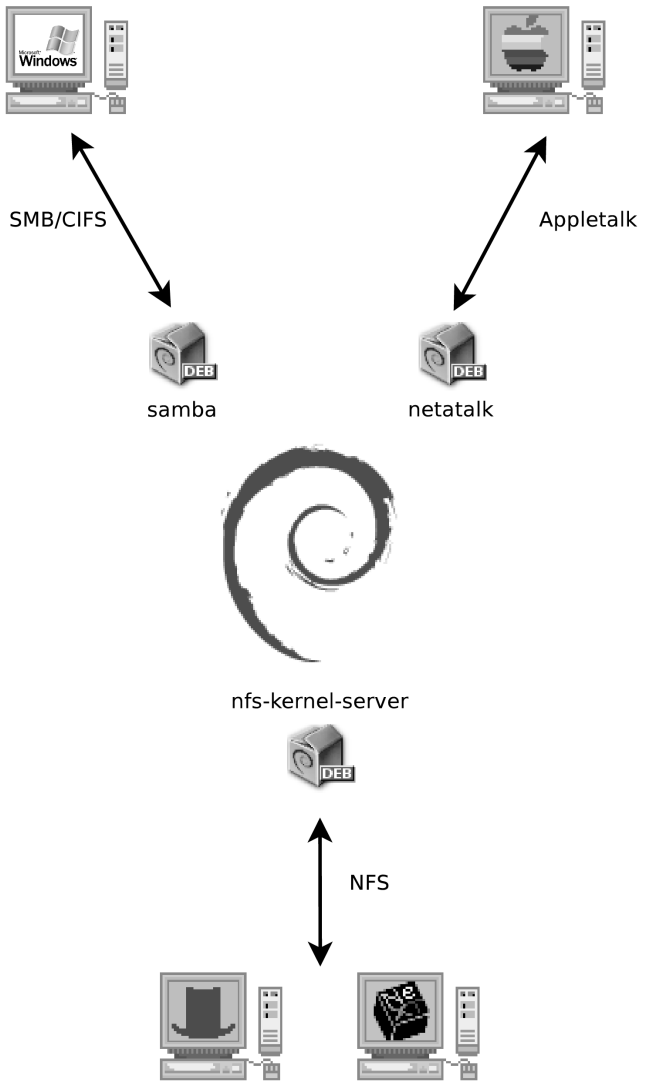


Figure 3–1 Cohabitation de Debian avec MacOS, Windows et les systèmes Unix

Recenser et identifier les services

Aussi simple qu'elle paraisse, cette étape est indispensable. Un administrateur sérieux connaît vraisemblablement les rôles principaux de chaque serveur, mais ceux-ci évoluent et quelques utilisateurs expérimentés auront parfois installé des services « sauvages ». Connaître leur existence vous permettra au moins de décider de leur sort au lieu de les supprimer par mégarde.

À ce titre, il est souhaitable d'informer vos utilisateurs de votre projet quelque temps avant la migration effective du serveur.

Réseau et processus

L'outil **nmap** (paquet Debian du même nom) identifiera rapidement les services Internet hébergés par une machine reliée au réseau sans même nécessiter de s'y connecter (**login**). Il suffit d'invoquer la commande suivante sur une autre machine connectée au même réseau :

```
$ nmap nomserveur
Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) at 2004-08-
  02 13:50 CEST
Interesting ports on nomserveur (127.0.0.1):
(The 1651 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
9/tcp   open  discard
13/tcp  open  daytime
22/tcp  open  ssh
25/tcp  open  smtp
37/tcp  open  time
80/tcp  open  http
111/tcp open  rpcbind
608/tcp open  sift-uft

Nmap run completed -- 1 IP address (1 host up) scanned in 0.503 seconds
```

ALTERNATIVE **Utiliser `netstat` pour trouver la liste des services disponibles**

Sur une machine Linux, la commande **`netstat -tupan`** dresse la liste de ses sessions TCP actives ou en attente ainsi que des ports UDP que les programmes actifs écoutent. Cela facilite le recensement des services proposés sur le réseau.

Si le serveur est une machine Unix offrant un compte shell aux utilisateurs, il est intéressant de déterminer si des processus s'exécutent en tâche de fond, en l'absence de leur propriétaire. La commande **`ps auxw`** affiche tous les processus et leur identifiant utilisateur associé. En croisant ces informations avec la sortie de la commande **who** (donnant la liste des utilisateurs connectés), il est possible de retrouver d'éventuels serveurs sauvages ou des programmes fonctionnant en tâche de fond. La consultation des tables de processus planifiés (crontabs) fournira souvent des renseignements intéressants sur les fonctions remplies par le serveur (l'explication complète des commandes de **cron** se trouve dans la section « Planification synchrone » du chapitre 9).

Dans tous les cas, il est indispensable de prévoir une sauvegarde du serveur : elle permettra de récupérer des informations a posteriori, quand les utilisateurs feront état de problèmes concrets dus à la migration.

Conserver la configuration

Il convient de conserver la configuration de chaque service identifié afin de pouvoir installer l'équivalent sur le serveur mis à jour. Le strict minimum est d'imprimer les fichiers de configuration et d'en faire une copie de sauvegarde.

Pour des machines Unix, la configuration se trouve habituellement sous /etc/ mais il se peut qu'elle soit placée dans un sous-répertoire de /usr/local. C'est le cas lorsqu'un logiciel est installé depuis ses sources plutôt qu'avec un paquet.

Pour les services gérant des données (comme les bases de données), il est fortement recommandé d'exporter celles-ci dans un format standard, plus facile à reprendre par le nouveau logiciel. Un tel format est généralement en mode texte

et documenté : il s'agira par exemple d'un *dump* SQL pour une base de données et d'un fichier LDIF pour un serveur LDAP.

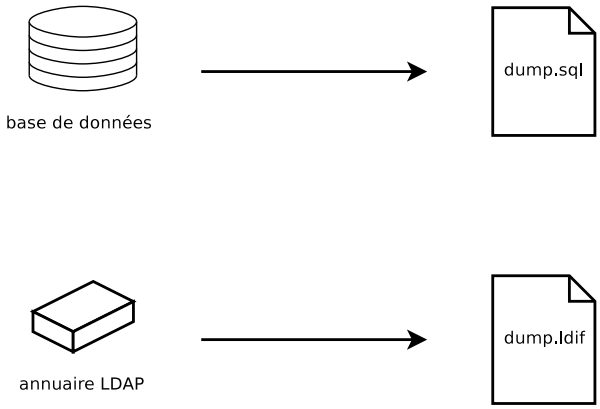


Figure 3–2 Sauvegarde des bases de données

Chaque logiciel serveur est différent et il est impossible de détailler tous les cas existants. Reportez-vous aux documentations du logiciel actuel et du nouveau logiciel pour identifier les portions exportables puis réimportables et celles qui nécessiteront un travail manuel. La lecture de ce livre vous éclairera déjà sur la configuration des principaux logiciels serveurs sous Linux.

Prendre en main un serveur Debian existant

Pour en reprendre efficacement l'administration, on pourra analyser une machine déjà animée par Debian.

Le premier fichier à vérifier est /etc/debian_version, qui contient habituellement le numéro de version du système Debian installé (il fait partie du paquet *base-files*). S'il indique `testing/unstable`, c'est que le système a été mis à jour avec des paquets provenant d'une de ces deux distributions en développement.

Le programme **`apt-show-versions`** (du paquet Debian éponyme) consulte la liste des paquets installés et identifie les versions disponibles. **`aptitude`** permet aussi d'effectuer ces tâches, d'une manière moins systématique.

Un coup d'œil au fichier /etc/apt/sources.list montrera la provenance probable des paquets Debian déjà installés. Si beaucoup de sources inconnues apparaissent, l'administrateur pourra choisir de réinstaller complètement l'ordinateur pour s'assurer une compatibilité optimale avec les logiciels fournis par Debian.

Ce fichier sources.list est souvent un bon indicateur : la majorité des administrateurs gardent au moins en commentaires les sources APT employées par le passé. Mais il est toujours possible que des sources employées par le passé aient été supprimées, voire que l'ancien administrateur ait installé manuellement (avec **`dpkg`**) des paquets téléchargés sur l'Internet. Dans ce cas, la machine trompe par

son apparence de Debian « standard ». C'est pourquoi il convient d'être attentif à tout indice pouvant trahir la présence de paquets externes (apparition de fichiers .deb dans des répertoires inhabituels, numéros de version de paquet dotés d'un suffixe particulier représentant l'origine du paquet — comme `ubuntu` ou `ximian`, etc.)

De même, il est intéressant d'analyser le contenu du répertoire /usr/local/, prévu pour contenir des programmes compilés et installés manuellement. Répertorier les logiciels installés de cette manière est riche d'enseignements, car cela incite à s'interroger sur la raison justifiant le non-emploi du paquet Debian correspondant — si un tel paquet existe.

Installer Debian

Toutes les informations relatives au serveur actuel étant maintenant connues, il est possible de mettre celui-ci hors service et d'y installer Debian.

Pour en choisir la version adéquate, il faut connaître l'architecture de l'ordinateur. S'il s'agit d'un PC, ce sera vraisemblablement i386. Dans les autres cas, on pourra restreindre les possibilités en fonction du système précédemment employé.

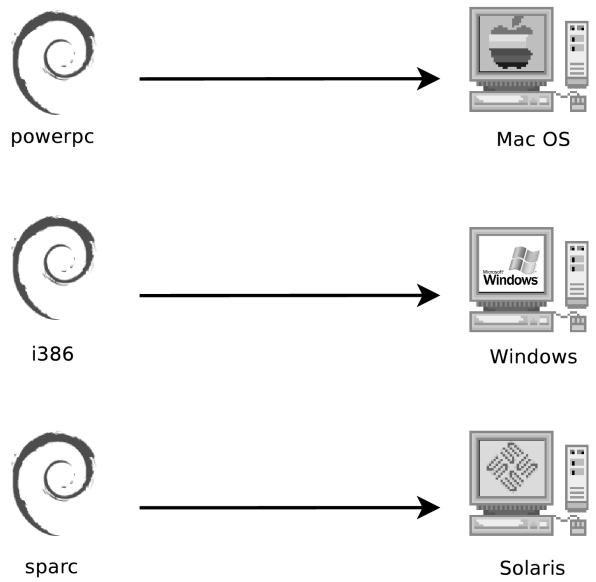


Figure 3–3 Installer la Debian adaptée à chaque ordinateur

MATÉRIEL **PC de dernière génération**

Certains ordinateurs récents sont pourvus de processeurs 64 bits d'Intel et d'AMD, compatibles avec les anciens processeurs 32 bits : les logiciels compilés pour architecture « i386 » fonctionneront donc. En revanche, ce mode compatibilité ne met pas à profit les capacités de ces nouveaux processeurs. C'est pourquoi Debian prévoit les architectures « ia64 » pour les processeurs Itanium d'Intel et « amd64 » pour ceux d'AMD. Cette dernière, bien que déjà prête et fonctionnelle, n'est pas encore officiellement intégrée sur les serveurs Debian. Cela sera chose faite dès la sortie de Sarge.

Le tableau 3.1 ne prétend pas être exhaustif mais pourra vous aider. Dans tous les cas, la documentation d'origine de l'ordinateur vous renseignera de manière plus certaine.

Tableau 3–1 Correspondance entre système d'exploitation et architecture

Système d'exploitation	Architecture(s)
DEC Unix (OSF/1)	alpha, mipsel
HP Unix	hppa
IBM AIX	powerpc
Irix	mips
MacOS	powerpc, m68k
MVS	s390
Solaris, SunOS	sparc, m68k, i386
Ultrix	mips
VMS	alpha
Windows NT	i386, alpha, mipsel

Installer et configurer les services sélectionnés

Une fois Debian installée, il s'agit d'installer et de configurer un à un tous les services que cet ordinateur doit héberger. La nouvelle configuration devra prendre en compte la précédente pour assurer une transition tout en douceur. Toutes les informations récoltées dans les deux premières étapes sont nécessaires pour mener à bien celle-ci.

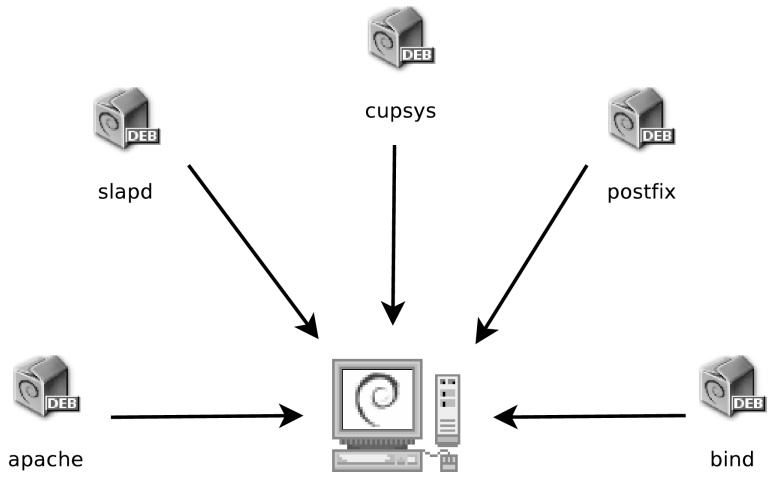


Figure 3–4 Installer les services sélectionnés

Avant de se lancer corps et âme dans cet exercice, il est fortement recommandé de consulter le reste de ce livre, grâce auquel vous aurez une idée plus précise sur la manière de configurer les services prévus.

4


PROD. NO.
Debian
SCENE
1-01
TAKE
install
DATE
ROLL
A012
SOUND
OPEN

Installation

Pour utiliser Debian, il faut l'avoir installée sur un ordinateur, tâche complexe prise en charge par le programme *debian-installer*. Une bonne installation implique de nombreuses opérations. Ce chapitre les passe en revue dans l'ordre dans lequel elles sont habituellement effectuées.

SOMMAIRE

- Méthodes d'installation
- Étapes du programme d'installation
- Le premier démarrage

MOTS-CLEFS

- Installation
- Partitionnement
- Formatage
- Système de fichiers
- Secteur d'amorçage
- Détection de matériel

Sarge est la toute première version de Debian à bénéficier du nouvel installateur **`debian-installer`**, que sa conception modulaire rend opérationnel dans de nombreux scénarios différents. Cette nouvelle version introduit également la détection automatique du matériel. D'une manière générale, cet installateur est beaucoup moins déroutant pour le débutant puisqu'il supprime de nombreuses questions et assiste l'utilisateur à chaque étape du processus, notamment le partitionnement, bête noire des nouveaux venus.

En revanche, il est beaucoup plus gourmand que son prédécesseur, et requiert 32 Mo de mémoire (64 Mo sont toutefois conseillés). Ces limitations auraient pu poser problème il y a quelques années mais plus à l'heure actuelle, chaque nouvel ordinateur comptant au minimum 128 Mo de mémoire RAM. Même les plus vieux ordinateurs de Falcot disposent de 64 Mo de RAM suite à une mise à jour effectuée l'année dernière.

Méthodes d'installation

L'installation d'un système Debian est possible depuis divers types de médias, pour peu que le BIOS de la machine le permette. On pourra ainsi amorcer grâce à un cédérom, une clé USB, voire à travers un réseau.

B.A.-BA **BIOS, l'interface matériel/logiciel**

Le BIOS (abréviation de *Basic Input/Output System*, ou système élémentaire d'entrées-sorties) est un logiciel intégré à la carte mère et exécuté au démarrage du PC pour charger un système d'exploitation (par l'intermédiaire d'un chargeur d'amorçage adapté). Il reste ensuite présent en arrière-plan pour assurer une interface entre le matériel et le logiciel (en l'occurrence, le noyau Linux).

Installation depuis un cédérom

Le média d'installation le plus employé est le cédérom : l'ordinateur s'amorce sur ce dernier et le programme d'installation prend la main.

Divers cédéroms ciblent chacun des usages différents : *netinst* (installation réseau) contient l'installateur et le système de base de Debian ; tous les autres logiciels seront téléchargés. Son « image », c'est-à-dire le système de fichiers ISO-9660 présentant le contenu exact du disque, occupe environ 110 Mo. Quant au cédérom de type carte de visite (*businesscard* ou *bizcard*), il ne fournit que l'installateur, tous les paquets Debian (y compris ceux du système de base) devant être téléchargés. Son image n'occupant que 50 Mo, elle peut être gravée sur un cédérom au format « carte de visite » — ce qui explique ce nom. Enfin, le jeu complet propose tous les paquets et permet d'installer un ordinateur sans accès à l'Internet — ce qui suppose tout de même plus d'une dizaine de cédéroms (ou deux dévédéroms). Mais les logiciels sont répartis sur les disques en fonction de leur popularité et de

leur importance ; les trois premiers suffiront donc à la majorité des installations car ils contiennent la plupart des logiciels les plus utilisés.

Pour se procurer des cédéroms Debian, on peut bien sûr télécharger et graver leur image. Mais il est aussi possible de les acheter, et de faire par la même occasion un petit don au projet. Consultez le site web pour connaître la liste des vendeurs et des sites de téléchargement des images de ces cédéroms.

▸ http://www.debian.org/CD/index.fr.html

PRATIQUE **Debian sur Cédérom**

Le cédérom offert avec ce livre contient Debian 3.1 « *release candidate 3* » (architecture « i386 »). Il a été préparé quelques semaines à peine avant la sortie officielle prévue de *Debian Sarge*. Son contenu a été adapté pour intégrer la majorité des logiciels étudiés ou cités (mis à part GNOME et KDE, trop volumineux pour y tenir).

Des cédéroms Debian sont également proposés à la vente ; je recommande notamment la société Ikarios, qui propose aussi de nombreux autres articles intéressants en rapport avec le logiciel libre (vêtements avec le logo Debian, autocollants, livres, etc.).

▸ http://www.ikarios.com/

Vous trouverez également quelques offres spéciales sur mon site, n'hésitez pas à le consulter :

▸ http://www.ouaza.com/livre/admin-debian/

Démarrage depuis une clé USB

Les ordinateurs récents étant capables de démarrer depuis des périphériques USB, il est également possible d'installer Debian depuis une clé USB (ce n'est qu'un petit disque de mémoire Flash) — mais cette technique n'est pas encore documentée. Tous les BIOS ne sont pas de même facture : certains savent démarrer sur des périphériques USB 2.0, d'autres ne gèrent que l'USB 1.1. Un développeur a rédigé un petit document expliquant (en anglais) comment créer une clé USB contenant **`debian-installer`**.

▸ http://d-i.pascal.at

Installation par *boot* réseau

De nombreux BIOS permettent d'amorcer directement sur le réseau en téléchargeant le noyau à démarrer. Cette méthode peut être salvatrice si l'ordinateur ne dispose pas de lecteur de cédérom ou si le BIOS ne peut amorcer sur un tel média.

Tous les détails de cette méthode sont disponibles dans le Wiki de Debian ou dans le guide d'installation.

▸ http://wiki.debian.net/index.cgi?DebianInstallerNetbootPXE

▸ http://d-i.alioth.debian.org/manual/fr.i386/index.html

COMMUNAUTÉ **Le site `debian.net` et son Wiki**

Le domaine *debian.net* ne constitue pas une ressource officielle du projet Debian. Chaque développeur Debian a la possibilité d'employer ce nom de domaine pour l'usage de son choix. On y trouve des services officieux (parfois des sites personnels) hébergés sur une machine n'appartenant pas au projet et mis en place par des développeurs Debian, voire des prototypes attendant d'être migrés sur *debian.org*. Deux raisons peuvent expliquer cet état de fait : soit personne ne souhaite faire l'effort nécessaire à sa transformation en service officiel (hébergé dans le domaine *debian.org*), soit le service est trop controversé pour être officialisé.

Le Wiki (`wiki.debian.net`), site collaboratif où même de simples visiteurs peuvent faire des suggestions depuis un navigateur, répond probablement du premier cas.

Étapes du programme d'installation

Exécution du programme d'installation

Quand le BIOS a démarré sur le cédérom, l'invite du chargeur d'amorçage Isolinux apparaît (`boot:`). À ce stade, le noyau Linux n'est pas encore chargé ; cette invite permet justement de choisir le noyau à démarrer et de saisir d'éventuelles options à lui passer.

Pour une installation standard, une pression sur la touche **`Entrée`** suffit pour enchaîner sur la suite de l'installation.

Les touches **`F1`** à **`F10`** affichent différents écrans d'aide détaillant les options possibles à l'invite. L'une d'entre elles permet par exemple de mettre en place un noyau de la série 2.6 (au lieu de la série 2.4, employée par défaut) ; on l'activera en tapant **`linux26`** et en validant par **`Entrée`**.

Le mode « expert » détaille toutes les options possibles au cours de l'installation et permet de naviguer entre les différentes étapes sans qu'elles s'enchaînent automatiquement. Attention, ce mode très verbeux pourra dérouter par la multitude des choix de configuration qu'il propose. Pour l'employer, il suffit de saisir **`expert`** et de valider par **`Entrée`**. Pour installer un noyau 2.6 en mode expert, on tapera **`expert26`**.

Une fois démarré, le programme d'installation nous guide d'étape en étape tout au long du processus. Cette section détaille chacune d'entre elles, leurs tenants et leurs aboutissants. Nous reproduisons le cas d'un cédérom *netinst* — les autres types d'installation pouvant varier quelque peu.

B.A.-BA Chargeur d'amorçage

Le chargeur d'amorçage (ou de démarrage), *bootloader* en anglais, est un programme de bas niveau chargé de démarrer le noyau Linux juste après que le BIOS lui a passé la main. Pour mener cette mission à bien, il doit être capable de « retrouver » sur le disque le noyau Linux à démarrer. Sur architecture i386, les deux programmes les plus employés pour effectuer cette tâche sont LILO, le plus ancien, et GRUB, un challenger plus moderne.

B.A.-BA Naviguer grâce au clavier

Certaines étapes du processus d'installation nécessitent une saisie d'informations. Ces écrans disposent alors de plusieurs zones qui peuvent « avoir le *focus* » (zone de saisie de texte, cases à cocher, liste de choix, boutons OK et Annuler), et la touche tabulation permet de circuler entre elles.

Choix de la langue

Le programme d'installation débute en anglais mais la toute première étape consiste à choisir la langue utilisée par la suite. Opter pour le français fournira une installation entièrement traduite (et un système configuré en français à l'issue du processus). Ce choix sert également à proposer des choix par défaut plus pertinents lors des étapes suivantes (la disposition du clavier notamment).

B.A.-BA Versions du noyau Linux

Les versions de noyau de deuxième nombre pair correspondent aux noyaux d'une série dite « stable » (ex : « 2.4.27 », « 2.6.8 »). Les autres sont des noyaux de développement (ex : « 2.3.55 », « 2.5.30 »). Une série de travail dédiée au développement coexiste traditionnellement avec une série stable pour que le travail puisse perpétuellement continuer. Des exceptions existent cependant : quelques temps après la création d'une nouvelle série stable. Linus Torvalds, auteur de Linux et son mainteneur principal, souhaite que les efforts de tous soient axés sur la stabilisation de cette nouvelle série plutôt que dispersés sur de nouveaux développements.
Les noyaux des séries 2.4 et 2.6 sont donc tous stables, mais les seconds sont plus récents et donc peut-être un peu moins éprouvés. En contrepartie, ils intègrent plus de pilotes de périphériques récents.



Figure 4–1 Choix de la langue

Choix du pays

La deuxième étape consiste à choisir le pays. Associée à la langue, cette information permettra de proposer une disposition de clavier encore plus adaptée. Elle influera aussi sur la configuration ultérieure du fuseau horaire. Dans le cas de la France, un clavier de type « azerty » sera proposé et le fuseau horaire sera « Europe/Paris ».

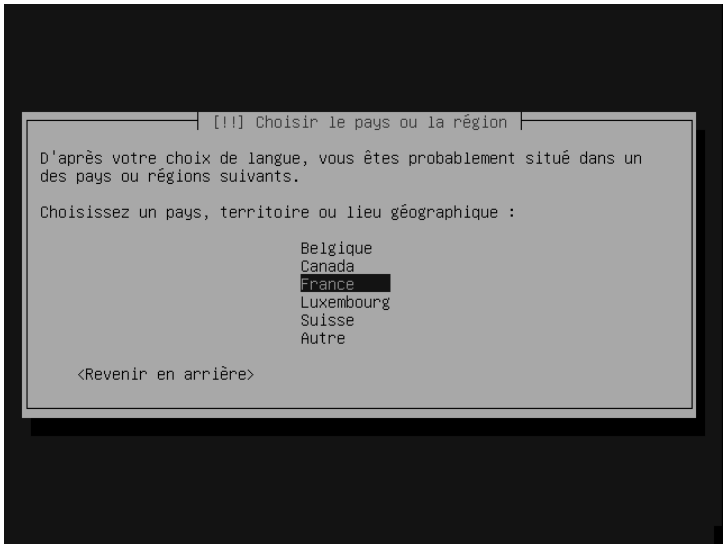


Figure 4–2 Choix du pays

Choix de la disposition du clavier

Le clavier « Français » proposé convient pour les claviers azerty traditionnels. Il prend en charge le symbole euro. « Français (OBSOLÈTE) » est le même clavier, sans ce symbole.

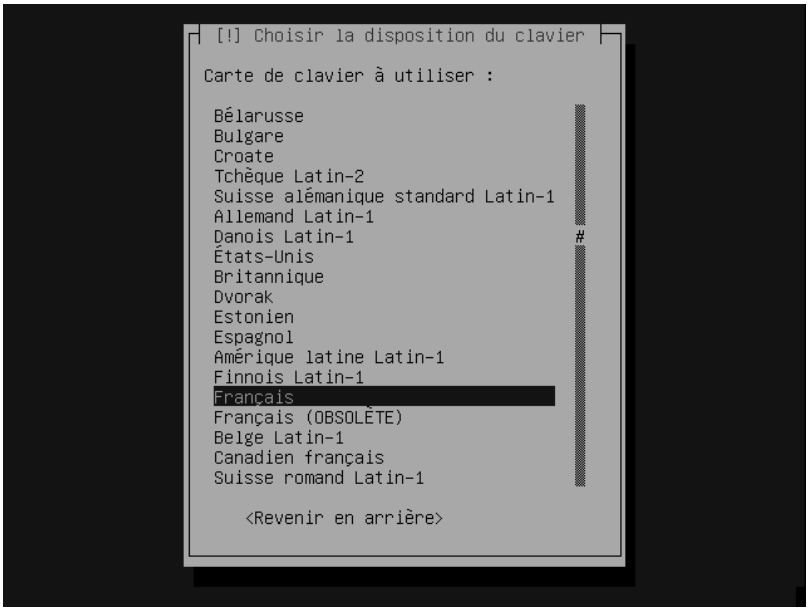


Figure 4–3 Choix du clavier

Détection du matériel

Cette étape est entièrement automatique dans l'immense majorité des cas. L'installateur détecte le matériel et cherche notamment à identifier le lecteur de cédérom employé afin de pouvoir accéder à son contenu. Il charge les modules correspondant aux différents éléments détectés puis « monte » le cédérom afin de pouvoir le lire. Les étapes précédentes étaient entièrement contenues dans l'image de démarrage intégrée au CD, fichier de taille limitée et chargé en mémoire par le BIOS lors de l'amorçage du CD.

L'installateur gère l'immense majorité des lecteurs, en particulier les périphériques ATAPI (parfois appelés IDE ou EIDE) standards. Toutefois, si la détection du lecteur de cédérom échoue, l'installateur propose de charger (par exemple depuis une disquette) un module noyau correspondant au pilote du cédérom.

Chargement des composants

Le contenu du CD désormais accessible, l'installateur rapatrie tous les fichiers nécessaires à la poursuite de sa tâche. Cela comprend des pilotes supplémentaires pour le reste du matériel (et notamment la carte réseau) ainsi que tous les composants du programme d'installation.

Détection du matériel réseau

Cette étape automatique cherche à identifier la carte réseau et à charger le module correspondant. À défaut de reconnaissance automatique, il est possible de sélectionner manuellement le module à charger. Si aucun module ne fonctionne, il est possible de charger un module spécifique depuis une disquette. Cette dernière solution ne sert réellement que si le pilote adéquat n'est pas intégré au noyau Linux standard, et qu'il est disponible par ailleurs, par exemple sur le site du fabricant.

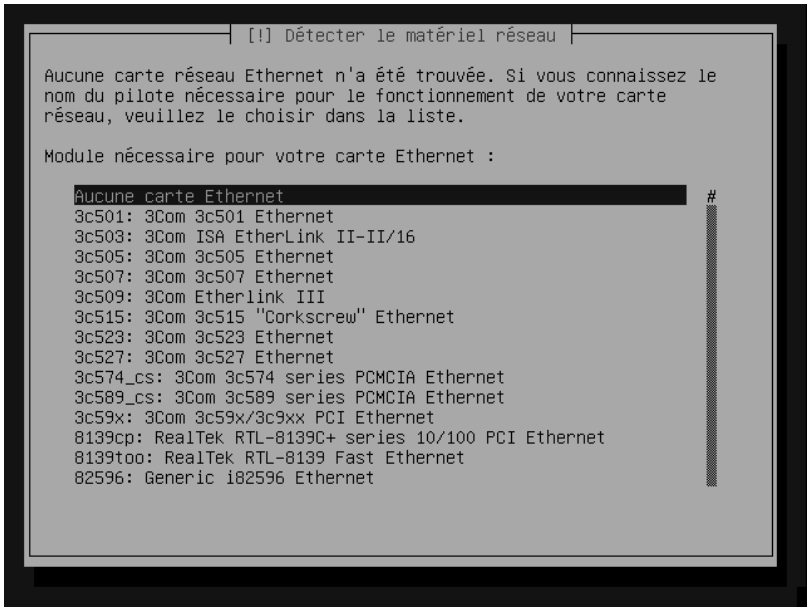


Figure 4–4 Choix manuel d'un module pour la carte réseau

Cette étape doit absolument réussir pour les installations de type *netinst* ou *businesscard*, puisque les paquets Debian doivent y être chargés sur le réseau.

Configuration du réseau

Soucieux d'automatiser au maximum le processus, l'installateur tente une configuration automatique du réseau par DHCP. Si celle-ci échoue, il propose plusieurs

choix : réessayer une configuration DHCP normale, effectuer une configuration DHCP en annonçant un nom de machine, ou mettre en place une configuration statique du réseau.

Cette dernière demande successivement une adresse IP, un masque de sous-réseau, une adresse IP pour une éventuelle passerelle, un nom de machine et un nom de domaine.

ATTENTION **Ne pas improviser**

Beaucoup de réseaux locaux reposant sur une confiance concédée a priori à toutes les machines, une configuration inadéquate d'un ordinateur y cause souvent des dysfonctionnements. Par conséquent, ne connectez pas votre machine à un réseau sans convenir au préalable avec son administrateur des modalités correspondantes (par exemple le numéro IP, le masque réseau, l'adresse de *broadcast*…).

ASTUCE **Configuration sans DHCP**

Si le réseau local est équipé d'un serveur DHCP que vous ne souhaitez pas utiliser car vous préférez définir une adresse IP statique pour la machine en cours d'installation, vous pourrez lors du démarrage sur le cédérom indiquer l'option **`netcfg/use_dhcp=false`**. Pour une installation standard, il convient de taper **`linux netcfg/use_dhcp=false`** à l'invite `isolinux` (**`linux`** deviendra, *mutatis mutandi*, **`expert`**, **`linux26`** ou encore **`expert26`** selon le type d'installation souhaité).

Détection des disques et autres périphériques

Cette étape automatique détecte les disques pouvant potentiellement accueillir Debian. Ils seront proposés dans l'étape suivante : le partitionnement.

Démarrage de l'outil de partitionnement

CULTURE **Utilité du partitionnement**

Le partitionnement, étape indispensable de l'installation, consiste à diviser l'espace disponible sur les disques durs (chaque sous-partie étant alors appelée une « partition ») en fonction des données à stocker et de l'usage prévu de l'ordinateur. Cette étape intègre aussi le choix des systèmes de fichiers employés. Toutes ces décisions ont une influence en termes de performances, de sécurité des données, et d'administration du serveur.

L'étape du partitionnement est traditionnellement difficile pour les utilisateurs débutants. Il faut en effet définir les différentes portions des disques (ou « partitions ») qui accueilleront le système de fichiers de Linux et sa mémoire virtuelle (*swap*). La tâche se complique si un autre système d'exploitation existe déjà et qu'on souhaite le conserver. En effet, il faudra alors veiller à ne pas altérer ses partitions (ou à les redimensionner de manière indolore).

L'ancien installateur invoquait l'utilitaire de partitionnement et l'utilisateur devait décider seul des partitions à créer. Le nouveau est beaucoup plus convivial de ce point de vue : le logiciel de partitionnement dispose d'un mode « assisté » qui propose à l'utilisateur les partitions à créer — dans la majorité des cas il suffit de valider ses propositions.

Le premier écran de l'utilitaire de partitionnement propose d'employer un disque complet pour créer les diverses partitions. Pour un ordinateur (neuf) qui sera dédié à Linux, cette option est vraisemblablement la plus simple. Si l'ordinateur compte deux disques pour deux systèmes d'exploitation, consacrer un disque à chacun est également une solution facilitant le partitionnement. Dans ces deux cas, choisissez le disque à consacrer à Linux en validant l'option correspondante (par exemple, « Effacer le disque IDE1 maître » pour installer Debian sur le premier disque). Vous débutez alors un partitionnement assisté. Dans les autres cas, quand Linux doit cohabiter avec des partitions déjà présentes, il faudra opter pour un partitionnement manuel.

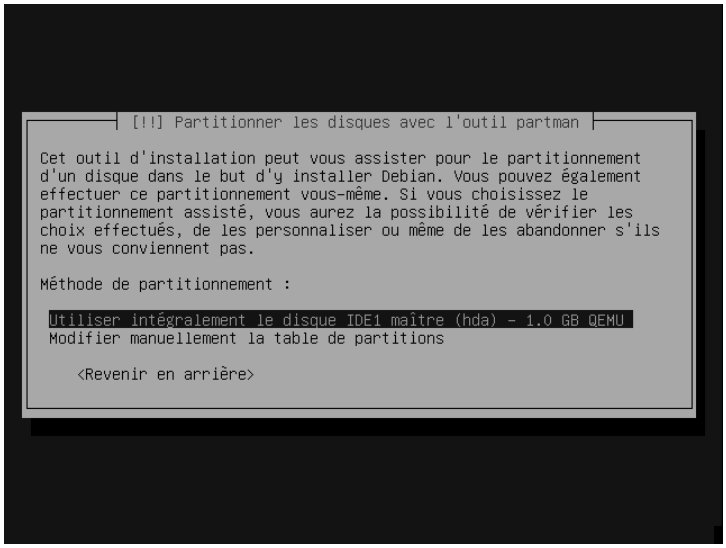


Figure 4–5 Début du partitionnement

Partitionnement assisté

L'outil de partitionnement assisté propose trois méthodes de partitionnement, qui correspondent à des usages différents.

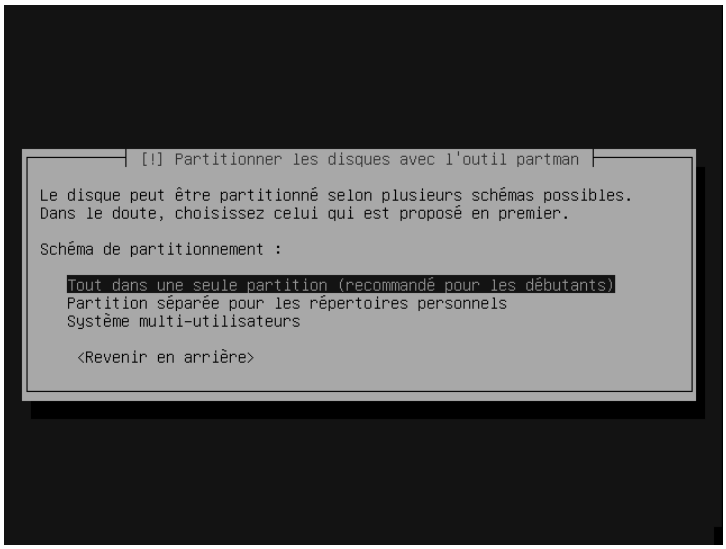


Figure 4–6 Partitionnement assisté

La première méthode s'intitule « Tout dans une seule partition ». Toute l'arborescence du système Linux est stockée dans un seul système de fichiers, correspondant à la racine /. Ce partitionnement simple et robuste convient parfaitement

pour des ordinateurs personnels ou mono-utilisateurs. En réalité, deux partitions verront le jour : la première abritera le système complet ; la seconde, la mémoire virtuelle.

La deuxième méthode, « Partition séparée pour les répertoires personnels », est similaire mais découpe l'arborescence en deux : une partie contient le système Linux (/) et la seconde les données des utilisateurs (c'est-à-dire tous les fichiers placés sous /home).

La dernière méthode de partitionnement, intitulée « Système multi-utilisateur », convient pour les serveurs. Elle découpe l'arborescence en de nombreuses partitions : en plus de la racine (/) et des comptes utilisateurs (/home), elle prévoit des partitions pour les applications (/usr), pour les données des logiciels serveurs (/var) et pour les fichiers temporaires (/tmp). Ces divisions ont plusieurs avantages. Les utilisateurs ne pourront pas bloquer le serveur en consommant tout l'espace disque disponible (ils ne pourront remplir que /tmp et /home). Les données des démons (et notamment les logs) ne pourront pas non plus bloquer le reste du système.

Après le choix du type de partitionnement, le logiciel calcule une proposition, qu'il détaille à l'écran, et qu'on peut au besoin modifier. On peut notamment choisir un autre système de fichiers si le choix standard (*ext3*) ne convient pas. Dans la plupart des cas, il suffit cependant d'accepter la proposition de partitionnement en validant l'option « Terminer le partitionnement et appliquer les changements ».

Partitionnement manuel

Le partitionnement manuel offre plus de souplesse et permet de choisir le rôle de chaque partition. Par ailleurs, ce mode est inévitable pour employer le RAID logiciel ou le gestionnaire de volumes logiques (LVM, ou *Logical Volume Manager*). Ces deux technologies sont présentées dans les sections qui suivent.

B.A-BA **Choisir un système de fichiers**

Un système de fichiers définit la manière d'organiser les données sur un disque. Chaque système de fichiers existant a ses mérites et ses limitations. Certains sont plus robustes, d'autres plus efficaces : si l'on connaît bien ses besoins, le choix d'un système de fichiers plus adapté est possible. De nombreuses comparaisons ont déjà été réalisées ; il semblerait que ReiserFS soit particulièrement efficace pour la lecture de nombreux petits fichiers ; *XFS*, quant à lui, est plus véloce avec de gros fichiers. Ext3, le système employé par défaut chez Debian, est un bon compromis, par ailleurs basé sur les deux précédentes versions du système de fichiers historiquement utilisé par Linux (*ext* et *ext2*).
Un système de fichiers journalisé (comme *ext3*, *reiserfs* ou *xfs*) prend des dispositions particulières afin qu'en cas d'interruption brutale, il soit toujours possible de revenir dans un état cohérent sans être contraint à une analyse complète du disque (comme c'était le cas avec le système *ext2*). Cette fonctionnalité est obtenue en remplissant un journal décrivant les opérations à effectuer avant de les exécuter réellement. Si une opération est interrompue, il sera possible de la « rejouer » à partir du journal. Inversement, si une interruption a lieu en cours de mise à jour du journal, le dernier changement demandé est simplement ignoré : les données en cours d'écriture sont peut-être perdues, mais les données sur le disque n'ayant pas changé, elles sont restées cohérentes. Il s'agit ni plus ni moins d'un mécanisme transactionnel appliqué au système de fichiers.

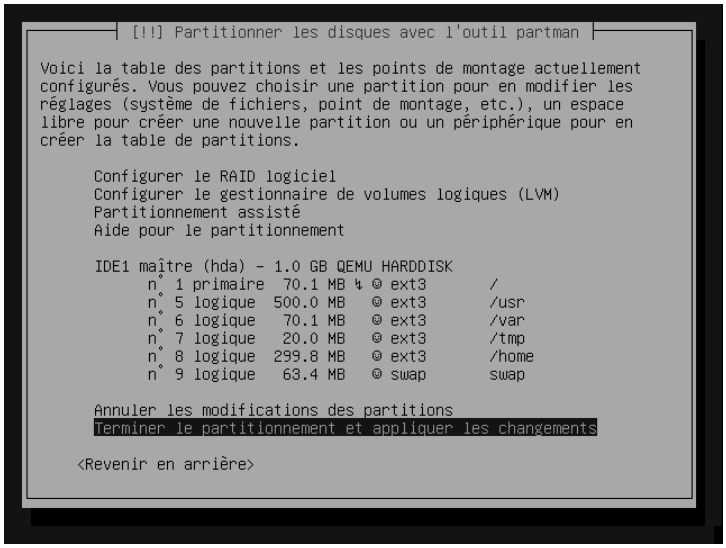


Figure 4–7 Valider le partitionnement

Le premier écran affiche les disques, les partitions qui les composent, et tout éventuel espace libre non encore partitionné. On peut sélectionner chaque élément affiché ; une pression sur la touche **Entrée** donne alors une liste d'actions possibles.

On peut effacer toutes les partitions d'un disque en sélectionnant celui-ci.

En sélectionnant un espace libre d'un disque, on peut créer manuellement une nouvelle partition. Il est également possible d'y effectuer un partitionnement assisté, solution intéressante pour un disque contenant déjà un système d'exploitation mais que l'on souhaite partitionner pour Linux de manière standard. Reportez-vous à la section précédente pour plus de détails sur le partitionnement assisté.

En sélectionnant une partition, on peut indiquer la manière dont on va l'utiliser :

- la formater et l'intégrer à l'arborescence en choisissant un point de montage ;
- l'employer comme partition d'échange (« *swap* ») ;
- en faire un « volume physique » pour LVM (notion détaillée plus loin dans ce chapitre) ;
- l'utiliser comme périphérique RAID (voir plus loin dans ce chapitre) ;
- ou ne pas l'exploiter et la laisser inchangée.

Emploi du RAID logiciel

Le RAID logiciel permet de dupliquer les informations stockées sur des disques durs pour éviter toute perte de données en cas de problème matériel de l'un

B.A.-BA **Point de montage**

Le point de montage est le répertoire de l'arborescence qui abritera le contenu du système de fichiers de la partition sélectionnée. Ainsi, une partition montée sur /home est traditionnellement prévue pour contenir les données des utilisateurs.
Si ce répertoire se nomme « / », on parle alors de la *racine* de l'arborescence, donc de la partition qui va réellement accueillir le système Debian.

B.A.-BA **Mémoire virtuelle, swap**

La mémoire virtuelle permet au noyau Linux en manque de mémoire vive (RAM) de libérer un peu de place en stockant sur la partition d'échange, donc sur le disque dur, une partie du contenu de la RAM restée inactive un certain temps.
Pour simuler la mémoire supplémentaire, Windows emploie un fichier d'échange contenu directement sur un système de fichiers. À l'inverse, Linux emploie une partition dédiée à cet usage, d'où le terme de « partition d'échange ».

d'entre eux. Le RAID de niveau 1 maintient une simple copie fidèle (miroir) d'un disque sur un autre alors que le RAID de niveau 5 répartit sur plusieurs disques des informations redondantes qui permettront de reconstituer intégralement un disque défaillant.

Nous traiterons ici du RAID de niveau 1, le plus simple à mettre en œuvre. La première étape est de créer deux partitions de taille identique situées sur deux disques différents et de les étiqueter « volume physique pour RAID ».

Il faut ensuite choisir dans l'outil de partitionnement l'élément « Configuration du RAID logiciel » pour transformer ces deux partitions en un nouveau disque virtuel et sélectionner « Création d'un périphérique multi-disques (MD) » dans cet écran de configuration. Suit alors une série de questions concernant ce nouveau périphérique. La première s'enquiert du niveau de RAID à employer — « RAID1 » dans notre cas. La deuxième demande le nombre de périphériques actifs — deux ici, soit le nombre de partitions à intégrer dans ce périphérique RAID logiciel. La troisième question concerne le nombre de périphériques de réserve — zéro ; on n'a prévu aucun disque supplémentaire pour prendre immédiatement la relève d'un éventuel disque défectueux. La dernière question demande de choisir les partitions retenues pour le périphérique RAID — soit les deux qu'on a prévues à cet usage (et qui devraient être les seuls choix possibles).

Au retour dans le menu principal, un nouveau disque virtuel « RAID » apparaît. On pourra y créer des partitions habituelles comme s'il s'agissait d'un disque réel.

Emploi de LVM (*Logical Volume Manager*)

LVM permet de créer des partitions « virtuelles » s'étendant sur plusieurs disques. L'intérêt est double : les tailles des partitions ne sont plus limitées par celles des disques individuels mais par leur volume cumulé, et on peut à tout moment augmenter la taille d'une partition existante, en ajoutant au besoin un disque supplémentaire.

LVM emploie une terminologie particulière : une partition virtuelle est un « volume logique », lui-même compris dans un « groupe de volumes », ou association de plusieurs « volumes physiques ». Chaque volume physique correspond en fait à une partition « réelle » (ou une partition RAID logicielle).

Cette technique fonctionne assez simplement : chaque volume, physique ou logique, est découpé en blocs de même taille, que LVM fait correspondre entre eux. L'ajout d'un nouveau disque entraîne la création d'un nouveau volume physique, et ses nouveaux blocs pourront être associés à n'importe quel groupe de volumes. Toutes les partitions du groupe de volumes ainsi agrandi disposeront alors d'espace supplémentaire pour s'étendre.

L'outil de partitionnement configure LVM en plusieurs étapes. Il faut d'abord créer sur les disques existants des partitions qui seront les « volumes physiques LVM ». Pour activer LVM, on choisira « Configuration de LVM », puis dans cet

écran de configuration, « Créer un groupe de volumes » — auquel on associera les volumes physiques existants. Enfin, on pourra créer des volumes logiques au sein de ce groupe de volumes.

Dans le menu du partitionneur, chaque groupe de volumes apparaît comme un disque, et chaque volume logique apparaît comme une partition (du groupe de volumes dont il fait partie).

Installation du système de base Debian

Cette étape installe les paquets du « système de base » de Debian. Celui-ci comprend les outils **`dpkg`** et **`apt`**, qui gèrent les paquets Debian, ainsi que les utilitaires nécessaires pour démarrer le système et commencer à l'exploiter. Les paquets Debian sont lus sur le disque (cas d'un cédérom *netinst* ou d'un cédérom complet) ou téléchargés (cas d'un cédérom *businesscard*).

Installation du chargeur d'amorçage GRUB

Le chargeur d'amorçage est le premier programme démarré par le BIOS. Ce programme charge en mémoire le noyau Linux puis l'exécute. Souvent, il propose un menu permettant de choisir le noyau à charger et/ou le système d'exploitation à démarrer.

Le menu proposé par GRUB contient par défaut les deux derniers noyaux Linux installés ainsi que tous les autres systèmes d'exploitation détectés (Windows principalement). Le fait de pouvoir choisir entre les deux derniers noyaux permet d'être capable de démarrer le système même si le dernier noyau installé est défectueux ou mal adapté à votre matériel.

Le seul élément de configuration nécessaire à GRUB est la sélection de son disque d'installation, mais la proposition par défaut convient dans la majorité des cas. Opter pour le premier disque (par exemple « `(hd0)` ») donnera le contrôle de l'amorçage à GRUB, puisque le BIOS démarre systématiquement sur le premier disque.

GRUB est le chargeur d'amorçage installé en standard par Debian, en raison de sa supériorité technique : il traite la plupart des systèmes de fichiers et n'a donc pas besoin d'être mis à jour à chaque installation d'un nouveau noyau car, lors de l'amorçage, il lit sa configuration et retrouve la position exacte du nouveau noyau. En revanche, il ne gère ni LVM ni le RAID logiciel — situations où il faut alors recommander LILO (autre chargeur d'amorçage). L'installateur propose automatiquement LILO dans ces situations.

ATTENTION **Chargeurs d'amorçage et architectures**
LILO et GRUB, mentionnés dans ce chapitre, sont des chargeurs d'amorçage pour l'architecture *i386*. Si vous installez Debian sur une autre architecture, c'est un autre chargeur qui sera employé. Citons entre autres **`yaboot`** ou **`quik`** pour *powerpc*, **`silo`** pour *sparc*, **`elilo`** pour *ia64*, **`aboot`** pour *alpha*, **`arcboot`** pour *mips*, **`atari-bootstrap`** ou **`vme-lilo`** pour *m68k*.

Terminer l'installation et redémarrer

La première phase de l'installation maintenant terminée, le programme vous invite à sortir le cédérom de son lecteur puis à redémarrer le PC.

Le premier démarrage

Après cette phase initiale, il reste encore quelques étapes pour achever l'installation — en particulier, la configuration du système de base (automatiquement exécutée). Il est toujours possible d'y revenir plus tard en tapant la commande **`base-config`**.

Horloge et fuseau horaire

La première question concerne l'horloge : il faut indiquer si elle correspond ou non à l'heure UTC (GMT). L'heure du système est donnée à titre indicatif. Si elle correspond à l'heure locale, répondez « non » ; sinon répondez « oui ». Une machine où Linux est le seul système d'exploitation a généralement une horloge réglée sur l'heure universelle (la conversion en heure locale étant réalisée par le système lui-même).

Un fuseau horaire vous est alors proposé. Il s'agit normalement du fuseau « Europe/Paris » si vous avez choisi la France comme pays au cours de l'installation. Si le fuseau horaire proposé n'est pas correct, sélectionnez-le dans la liste.

Mot de passe administrateur

Le compte super-utilisateur « root », réservé à l'administrateur de la machine, est automatiquement créé lors de l'installation : c'est pourquoi un mot de passe est demandé. Une confirmation (ou deuxième saisie identique) évitera toute erreur de saisie, difficile à retrouver ensuite !

Création du premier utilisateur

Debian impose également de créer un compte utilisateur standard pour que l'administrateur ne prenne pas la mauvaise habitude de travailler en tant que « root ». Le principe de précaution veut en effet que chaque tâche soit effectuée avec le minimum de droits nécessaires, pour limiter l'impact d'une mauvaise manipulation. C'est pourquoi on vous demandera successivement le nom complet de ce premier utilisateur, son identifiant, et son mot de passe.

Configuration de l'outil Debian de gestion de paquets (`apt`)

Pour installer des logiciels supplémentaires, il convient aussi de configurer APT, en lui indiquant où trouver les paquets Debian.

ATTENTION **Cédérom Debian dans le lecteur**

Si **`base-config`** détecte un disque d'installation Debian dans le lecteur de cédérom, cette étape est court-circuitée ; APT est automatiquement configuré pour lire les paquets depuis ce lecteur. Il proposera cependant d'« explorer » ainsi d'autres disques afin de référencer tous les paquets qu'ils stockent. C'est primordial si le disque fait partie d'un jeu de plusieurs cédéroms.

La première question demande une méthode d'accès. La méthode la plus populaire et retenue par les administrateurs de Falcot, « `http` », indique à APT de télécharger ses paquets depuis un serveur web. Les deux questions suivantes permettent de sélectionner ce serveur web en choisissant successivement un pays puis un miroir disponible dans ce pays (il s'agit d'un serveur public qui met à disposition une copie de tous les fichiers du serveur de Debian).

Enfin, le programme propose de recourir à un mandataire (proxy) HTTP. En son absence, l'accès à l'Internet sera direct. Si l'on tape `http://proxy.falcot.com:3128`, APT fera appel au *proxy/cache* de Falcot, un programme « Squid ». Il est possible de retrouver ces paramètres en consultant la configuration d'un navigateur web sur une autre machine connectée au même réseau.

Les fichiers Packages.gz et Sources.gz sont ensuite automatiquement téléchargés pour mettre à jour la liste des paquets reconnus par APT.

B.A.-BA **Mandataire HTTP, proxy**

Un mandataire (ou proxy) HTTP est un serveur effectuant une requête HTTP pour le compte des utilisateurs du réseau. Il permet parfois d'accélérer les téléchargements en gardant une copie des fichiers ayant transité par son biais (on parle alors de *proxy/cache*). Dans certains cas, c'est le seul moyen d'accéder à un serveur web externe ; il est alors indispensable de renseigner la question correspondante de l'installation pour que le programme puisse récupérer les paquets Debian par son intermédiaire.
Squid est le nom du logiciel serveur employé par Falcot SA pour offrir ce service.

Installation de logiciels supplémentaires

Pour faciliter l'installation d'ensembles logiciels cohérents, Debian crée des « tâches » dédiées à des usages spécifiques (serveur de messagerie, serveur de fichiers, etc.). On peut les sélectionner à l'installation ; tous les programmes les composant seront alors automatiquement installés.

Lors du choix des tâches à installer, on peut quand même sélectionner l'option « choix manuel des paquets ». Dans ce cas, le programme **`aptitude`** sera exécuté et permettra d'affiner la liste des paquets à installer (les paquets des tâches retenues étant présélectionnés).

Aptitude est une interface à APT en mode texte plein écran. Elle permet de naviguer dans la liste des paquets disponibles selon différents classements (paquets installés ou non, par tâche, par section, etc.) et de consulter toutes les informations

ASTUCE **Debian pense aux francophones**

Il existe une tâche dédiée à la localisation du système en français. Elle comprend de la documentation en français, des dictionnaires français et divers autres paquets utiles aux francophones. Elle est automatiquement pré-sélectionnée si le « Français » a été retenu pour la langue d'installation.

CULTURE **`dselect`, l'ancienne interface pour installer des paquets**

Avant **`aptitude`**, le programme standard pour sélectionner les paquets à installer était **`dselect`**, ancienne interface graphique associée à **`dpkg`**. Difficile d'emploi pour les débutants, il est donc déconseillé.

disponibles à propos de chacun d'entre eux (dépendances, conflits, description, etc.). Chaque paquet peut être marqué « à installer » (touche « **+** ») ou « à supprimer » (touche « **-** »). Toutes ces opérations seront effectuées simultanément après confirmation par appui sur la touche « **g** » (comme *go*, ou « allez ! »). En cas d'oubli de certains logiciels, aucun souci, il sera toujours possible d'exécuter à nouveau **`aptitude`** une fois l'installation initiale achevée.

Bien entendu, il est possible de ne sélectionner aucune tâche à installer. Dans ce cas, il vous suffira d'installer manuellement les logiciels souhaités avec la commande **`apt-get`** ou **`aptitude`** (également accessible en ligne de commande) en dehors du programme d'installation proprement dit.

VOCABULAIRE **Dépendance, conflit d'un paquet**

Dans le jargon des paquets Debian, une « dépendance » est un autre paquet nécessaire au bon fonctionnement du paquet concerné. Inversement, un « conflit » est un paquet qui ne peut pas cohabiter avec celui-ci.
Ces notions sont traitées plus en détail dans le chapitre 5.

Mise à jour du système

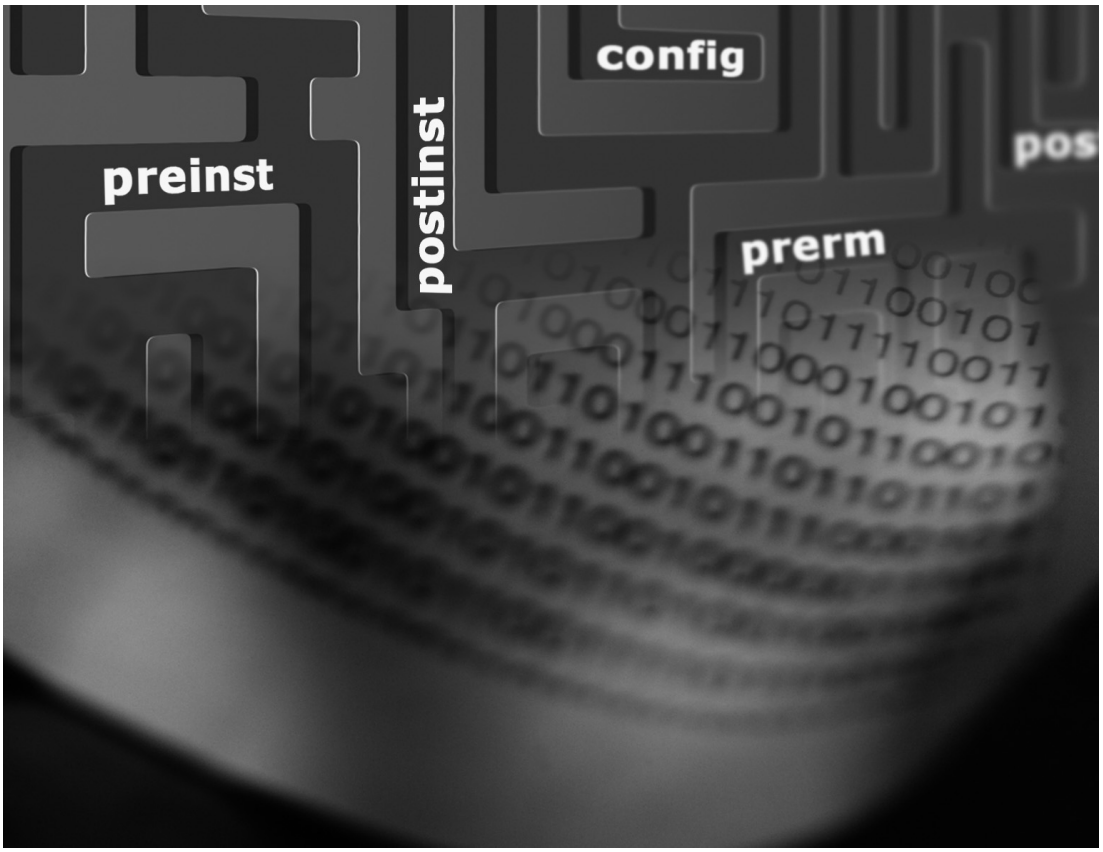
Un premier **`apt-get upgrade`** (commande de mise à jour automatique des versions des logiciels installés) est automatiquement exécuté, notamment en raison d'éventuelles mises à jour de sécurité. Ces mises à jour pourront impliquer quelques questions supplémentaires via **`debconf`**, l'outil de configuration standard de Debian.

Fin de l'installation

L'installation est terminée : l'écran d'identification apparaît. Vous pouvez vous identifier et commencer à travailler avec votre ordinateur.



5



Système de paquetage, outils et principes fondamentaux

En tant qu'administrateur de système Debian, vous allez régulièrement manipuler des paquets (fichiers .deb) car ils abritent des ensembles fonctionnels cohérents (applications, documentations...) dont ils facilitent l'installation et la maintenance. Mieux vaut donc savoir de quoi ils sont constitués et comment on les utilise.

SOMMAIRE

- Structure d'un paquet binaire
- Méta-informations d'un paquet
  - Description : fichier **`control`**
  - Scripts de configuration
  - Sommes de contrôle, liste des fichiers de configuration
- Structure d'un paquet source
  - Format
  - Utilité chez Debian
- Manipuler des paquets avec **`dpkg`**
  - Installation de paquets
  - Suppression de paquet
  - Autres fonctionnalités de **`dpkg`**
- Cohabitation avec d'autres systèmes de paquetages

MOTS-CLEFS

- Paquet binaire
- Paquet source
- dpkg
- dépendances
- conflit

Vous trouverez ci-après la description des structures et contenus des fichiers de paquets de type « binaire » puis « source ». Les premiers sont les fichiers .deb directement utilisables par **dpkg** alors que les seconds contiennent les codes sources des programmes ainsi que les instructions pour créer les paquets binaires.

Structure d'un paquet binaire

Le format d'un paquet Debian est conçu de telle sorte que son contenu puisse être extrait sur tout système Unix disposant des commandes classiques **ar**, **tar**, et **gzip**. Cette propriété anodine est importante du point de vue de la portabilité et de la récupération en cas de catastrophe.

Imaginons par exemple que vous ayez supprimé le programme **dpkg** par erreur et que vous ne puissiez donc plus installer de paquets Debian. **dpkg** étant lui-même un paquet Debian, votre système semble condamné... Heureusement, vous connaissez le format d'un paquet et pouvez donc télécharger le fichier .deb du paquet **dpkg** et l'installer manuellement (voir encadré « OUTILS »). Si par malheur un ou plusieurs des programmes **ar**, **tar** ou **gzip** avaient disparu, il suffirait de copier le programme manquant depuis un autre système (chacun fonctionnant de manière totalement autonome, une simple copie est suffisante).

Examinons le contenu d'un fichier .deb :

```
$ ar t dpkg_1.10.18_i386.deb
debian-binary
control.tar.gz
data.tar.gz
$ ar x dpkg_1.10.18_i386.deb
$ ls
control.tar.gz  data.tar.gz  debian-binary  dpkg_1.10.18_i386.deb
$ tar ztf data.tar.gz | head
./
./usr/
./usr/share/
./usr/share/doc/
./usr/share/doc/dpkg/
./usr/share/doc/dpkg/THANKS.gz
./usr/share/doc/dpkg/TODO.gz
./usr/share/doc/dpkg/changelog.gz
./usr/share/doc/dpkg/dpkg.cfg
./usr/share/doc/dpkg/pseudo-tags.gz
$ tar ztf control.tar.gz
./
./conffiles
./preinst
./prerm
./postinst
./postrm
./control
$ cat debian-binary
2.0
```

Comme vous le voyez, l'archive **ar** d'un paquet Debian est constituée de trois fichiers :

OUTILS **dpkg, APT et ar**

dpkg est le programme qui permet de manipuler des fichiers .deb, notamment de les extraire, analyser, décompacter, etc.

APT est un ensemble logiciel pour effectuer des modifications globales sur le système : installation ou suppression d'un paquet en gérant les dépendances, mise à jour du système, consultation des paquets disponibles, etc.

Quant au programme **ar**, il permet de manipuler les archives du même nom : **ar t *archive*** donne la liste des fichiers contenus dans l'archive, **ar x *archive*** extrait les fichiers de l'archive dans le répertoire courant, **ar d *archive fichier*** supprime un fichier de l'archive, etc. Sa page de manuel documente ses nombreuses autres opérations. **ar** est un outil très rudimentaire, qu'un administrateur Unix n'emploiera qu'à de rares occasions. Mais il se sert régulièrement de **tar**, programme de gestion d'archives et de fichiers plus évolué. C'est pourquoi il est facile de restaurer **dpkg** en cas de suppression involontaire. Il suffit de télécharger son paquet Debian et d'extraire le contenu de son archive data.tar.gz dans la racine du système (/) :

```
# ar x dpkg_1.10.18_i386.deb
# tar -C / -p -zxf data.tar.gz
```

- debian-binary. Il s'agit d'un fichier texte ne renfermant que le numéro de version du format .deb employé (en 2004 : version 2.0).
- control.tar.gz. Ce fichier d'archive rassemble les diverses méta-informations disponibles. Les outils de gestion des paquets y trouvent, entre autres, le nom et la version de l'ensemble abrité. Certaines de ces méta-informations leur permettent de déterminer s'il est ou non possible de l'installer ou de le désinstaller, par exemple en fonction de la liste des paquets déjà présents sur la machine.
- data.tar.gz. Cette archive contient tous les fichiers à extraire du paquet, c'est là que sont stockés les exécutables, la documentation, etc.

Méta-informations d'un paquet

Le paquet Debian n'est pas qu'une archive de fichiers destinés à l'installation. Il inclut le paquet dans un ensemble plus vaste en décrivant des relations avec les autres paquets Debian (dépendances, conflits, suggestions). Il fournit également des scripts permettant d'exécuter des commandes lors des différentes étapes du parcours du paquet (installation, suppression, mise à jour). Ces données employées par les outils de gestion des paquets ne font pas partie du logiciel empaqueté mais constituent, au sein du paquet, ce que l'on appelle ses « méta-informations » (informations portant sur les informations).

Description : fichier `control`

Ce fichier utilise une structure similaire à un en-tête de mail (défini par la RFC 822), qui ressemble pour l'exemple d'**`apt`** à :

```
$ apt-cache show apt
Package: apt
Priority: important
Section: base
Installed-Size: 2476
Maintainer: APT Development Team <deity@lists.debian.org>
Architecture: i386
Version: 0.5.21
Replaces: libapt-pkg-doc (<< 0.3.7), libapt-pkg-dev (<< 0.3.7)
Provides: libapt-pkg-libc6.3-5-3.3
Depends: libc6 (>= 2.3.2.ds1-4), libgcc1 (>= 1:3.3.2-1), libstdc
  ++5 (>= 1:3.3.2-1)
Suggests: aptitude | synaptic | gnome-apt | wajig, dpkg-dev, apt-doc
Description: Advanced front-end for dpkg
 This is Debian's next generation front-end for the dpkg package
 manager. It provides the apt-get utility and APT dselect method that
 provides a simpler, safer way to install and upgrade packages.
 .
 APT features complete installation ordering, multiple source
 capability and several other unique features, see the Users Guide in
 apt-doc.
```

B.A.-BA **RFC — les normes de l'Internet**

RFC abrège *Request For Comments* (appel à commentaires en anglais). Une RFC est un document généralement technique exposant ce qui deviendra une norme de l'Internet. Avant d'être standardisées et figées, ces normes sont soumises à une revue publique (d'où leur nom). C'est l'IETF (*Internet Engineering Task Force*) qui décide de l'évolution du statut de ces documents (*proposed standard*, *draft standard* ou *standard*, respectivement « proposition de standard », « brouillon de standard », « standard »). La RFC 2026 définit le processus de standardisation de protocoles de l'Internet.

▸ http://www.faqs.org/rfcs/rfc2026.html

Dépendances : champ *Depends*

Les dépendances sont définies dans le champ *Depends* des en-têtes du paquet. Il s'agit d'une liste de conditions à remplir pour que le paquet fonctionne correctement, informations utilisées par des outils comme **`apt`** pour installer les versions des bibliothèques dont dépend le programme à installer. Pour chaque dépendance, il est possible de restreindre l'espace des versions qui satisfont la condition. Autrement dit, il est possible d'exprimer le fait que l'on a besoin du paquet libc6 dans une version supérieure à « 2.3.0 » (cela s'écrit « **`libc6 (>> 2.3.0)`** »). Les opérateurs de comparaison de versions sont les suivants :

- **`<<`** : strictement inférieur à
- **`<=`** : inférieur ou égal à
- **`=`** : égal à (attention « `2.6.1` » n'est pas égal à « `2.6.1-1` »)
- **`>=`** : supérieur ou égal à
- **`>>`** : strictement supérieur à

Dans la liste des conditions à remplir, la virgule sert de séparateur. Son sens y est celui d'un « et » logique. Au sein d'une condition il est possible d'utiliser une barre verticale (« | ») pour introduire un « ou » logique. Ainsi la dépendance « (A ou B) et C » s'écrit **`A | B, C`**. En revanche, l'expression « A ou (B et C) » doit être développée en « (A ou B) et (A ou C) » puisque le champ *Depends* ne dispose pas de parenthèses pour changer les priorités entre les opérateurs logiques « ou » et « et ». Elle s'écrira donc **`A | B, A | C`**.

▸ http://www.debian.org/doc/debian-policy/ch-relationships.html

Le système de dépendances est un bon mécanisme pour garantir le fonctionnement d'un logiciel, mais il trouve un autre usage avec les « méta-paquets ». Il s'agit de paquets vides, décrivant uniquement des dépendances. Ils facilitent l'installation d'un ensemble cohérent de logiciels présélectionnés par le mainteneur du méta-paquet ; en effet **`apt-get install`** ***`méta-paquet`*** installera automatiquement l'ensemble de ces logiciels grâce aux dépendances du méta-paquet. Les paquets *gnome-desktop-environment*, *kde* et *kernel-image-2.6-686* sont des exemples de méta-paquets.

CHARTE DEBIAN **Pre-Depends, un Depends plus exigeant**

Des « pré-dépendances », données dans le champ « *Pre-Depends* » de l'en-tête du paquet, complètent les dépendances normales ; leur syntaxe est identique. Une dépendance normale indique que le paquet concerné doit être décompacté et configuré avant que le paquet la déclarant ne soit lui-même configuré. Une pré-dépendance stipule que le paquet concerné doit être décompacté et configuré avant même d'exécuter le script de pré-installation du paquet la déclarant, c'est-à-dire avant son installation proprement dite.
Une pré-dépendance est très contraignante pour **`apt`**, qui doit ordonnancer la liste des paquets à installer. Elles sont donc déconseillées en l'absence de nécessité stricte. Il est même recommandé de consulter l'avis des (autres) développeurs sur debian-devel@lists.debian.org avant d'ajouter une pré-dépendance. En règle générale, il est possible de trouver une solution de substitution qui permet de l'éviter.

CHARTE DEBIAN **Champs Recommends, Suggests, et Enhances**

Les champs *Recommends* (recommande) et *Suggests* (suggère) décrivent des dépendances non obligatoires. Les dépendances « recommandées », les plus importantes, améliorent considérablement les fonctionnalités offertes par le paquet sans pour autant être indispensables à son fonctionnement. Les dépendances « suggérées », secondaires, indiquent que certains paquets peuvent se compléter et augmenter leur utilité respective — mais il est parfaitement raisonnable d'installer l'un sans les autres.
Il faut systématiquement installer les paquets « recommandés » sauf si vous savez précisément pourquoi vous n'en avez pas besoin. Inversement, il est inutile d'installer les paquets « suggérés » sauf si vous savez pourquoi vous en aurez besoin.

Le champ *Enhances* (améliore) décrit lui aussi une suggestion, mais dans un contexte différent. Il se situe en effet dans le paquet suggéré, et non pas dans celui qui profite de la suggestion. Son intérêt est de pouvoir ajouter une suggestion sans devoir modifier le paquet concerné par elle. Ainsi, tous les *add-ons* (ajouts), *plugins* (greffons) et autres extensions d'un logiciel pourront ensuite prendre place dans la liste des suggestions liées au logiciel. Ce dernier champ — récemment créé — est encore largement ignoré par des programmes comme **apt-get** ou **synaptic**. L'objectif est cependant qu'une suggestion faite par le biais d'un champ *Enhances* apparaisse à l'utilisateur en complément des suggestions traditionnelles — réalisées avec le champ *Suggests*.

Conflits : champ *Conflicts*

Le champ *Conflicts* permet d'indiquer qu'un paquet ne peut être installé en même temps qu'un autre. Les raisons les plus courantes sont les suivantes : les deux paquets incluent un fichier portant le même nom, fournissent le même service sur le même port TCP, ou gênent mutuellement leur bon fonctionnement (par exemple dans le cas de mises à jour sans compatibilité ascendante : c'est le cas si la nouvelle version ne fonctionne plus comme l'ancienne et entraîne un dysfonctionnement en l'absence de dispositions particulières — comme l'emploi du champ *Conflicts* pour refuser une cohabitation indésirable).

dpkg refusera d'installer un paquet s'il déclenche un conflit avec un autre paquet déjà présent, sauf si le nouveau paquet précise qu'il « remplace » le paquet installé — auquel cas **dpkg** choisira de remplacer l'ancien par le nouveau. **apt-get** suit toujours vos instructions : si vous choisissez d'installer le nouveau paquet, il proposera automatiquement de désinstaller le paquet qui pose problème.

Éléments fournis : champ *Provides*

Ce champ introduit le concept très intéressant de « paquet virtuel ». Il a de nombreux rôles, mais on en distingue deux principaux. Le premier consiste à utiliser un paquet virtuel pour lui associer un service générique (le paquet « fournit » le service). Le second indique qu'un paquet en remplace complètement un autre, et qu'à ce titre il peut également satisfaire les dépendances déclarées sur celui-ci. Il est ainsi possible de créer un paquet de substitution sans avoir de contrainte sur son nom.

VOCABULAIRE **Méta-paquet et paquet virtuel**

Distinguons bien les méta-paquets des paquets virtuels. Les premiers sont des paquets réels (dotés de fichiers .deb), dont le seul intérêt est d'exprimer des dépendances.
Les paquets virtuels, quant à eux, n'existent pas physiquement ; il s'agit juste d'un moyen d'identifier des paquets réels sur la base d'un critère logique commun (service fourni, compatibilité avec un programme standard ou un paquet préexistant, etc.).

La fourniture d'un « service »

Détaillons le premier cas par un exemple : tous les serveurs de courrier électronique tels que **postfix** ou **sendmail** déclarent « fournir » le paquet

virtuel *mail-transport-agent*. Ainsi, tout paquet qui a besoin de ce service pour fonctionner (ce peut être un gestionnaire de listes de diffusion, comme **`majordomo`**, **`smartlist`**, **`sympa`**) se contentera de déclarer dans ses dépendances *mail-transport-agent* au lieu d'y préciser une grande liste de choix toujours incomplète (**`postfix`** | **`sendmail`** | **`exim`** | **`...`**). Par ailleurs, il ne sert à rien d'installer deux serveurs de courrier électronique ; c'est pourquoi chacun de ces paquets déclare un conflit avec le paquet virtuel *mail-transport-agent*. Le conflit avec soi-même est ignoré par le système, mais cette technique interdira d'installer de concert deux serveurs de courrier électronique.

CHARTE DEBIAN **Liste des paquets virtuels**

Pour que les paquets virtuels soient utiles, il faut que tout le monde s'entende sur leur nom. C'est pourquoi ils sont standardisés par la charte Debian. La liste comprend entre autres *mail-transport-agent* pour les serveurs de courrier électronique, *c-compiler* pour les compilateurs C, *www-browser* pour les navigateurs web, *httpd* pour les serveurs web, *ftp-server* pour les serveurs FTP, *x-terminal-emulator* pour les émulateurs de terminal en mode graphique (**`xterm`**) et *x-window-manager* pour les gestionnaires de fenêtres.
Retrouvez-en la liste complète sur le Web :
▸ http://www.debian.org/doc/packaging-manuals/virtual-package-names-list.txt

L'interchangeabilité avec un autre paquet

L'autre rôle principal du champ *Provides* consiste à indiquer qu'un paquet est capable d'offrir strictement le même service qu'un autre, ce qui arrive fréquemment pour des paquets renommés ou intégrés dans des logiciels plus vastes. Deux exemples illustreront ces cas de figure. Le paquet *libmd5-perl* contient un module Perl dont le développement est abandonné puisqu'une autre version plus évoluée et totalement compatible existe : le paquet *libdigest-md5-perl*. Cependant, de nombreux paquets déclarent encore des dépendances sur *libmd5-perl*, qui ne seront pas satisfaites si l'on dispose uniquement de *libdigest-md5-perl*. Pour éviter ce problème, on modifie ce dernier pour qu'il déclare `Provides: libmd5-perl` et le tour est joué. Par ailleurs, on déclarera sûrement un conflit et un remplacement pour que *libmd5-perl*, désormais obsolète, soit automatiquement supprimé. *Provides* s'avère encore intéressant lorsque le contenu d'un paquet est intégré dans un paquet plus vaste. Ainsi le module Perl *libstorable-perl* était un module facultatif en Perl 5.6, intégré en standard dans Perl 5.8. Le paquet *perl* dans sa version 5.8 déclare donc `Provides: libstorable-perl` afin que les dépendances sur ce paquet soient satisfaites si l'utilisateur dispose de Perl 5.8.

Limitations actuelles

Les paquets virtuels souffrent malgré tout de quelques limitations gênantes, dont la plus importante est l'absence de numéro de version. Pour reprendre l'exemple précédent, une dépendance `Depends: libstorable-perl (>= 1.6)` ne sera donc jamais satisfaite, pour le système de paquetage, par la présence de Perl 5.8 — bien qu'en réalité elle le soit probablement. Ne le sachant pas, le système

B.A.-BA **Perl, un langage de programmation**

Perl (*Practical Extraction and Report Language*, ou langage pratique d'extraction et de rapports), est un langage de programmation très populaire. Il dispose de nombreux modules prêts à l'emploi fournissant des fonctionnalités couvrant un spectre très large d'applications, et diffusés par le réseau de serveurs CPAN (*Comprehensive Perl Archive Network*, ou réseau exhaustif d'archives de Perl).

▸ http://www.perl.org

▸ http://www.cpan.org

S'agissant d'un langage interprété, un programme rédigé en Perl ne requiert pas de compilation préalable à son exécution. C'est pourquoi l'on parle de « scripts Perl ».

de paquetage opte pour une politique du moindre risque en supposant que les versions ne correspondent pas.

POUR ALLER PLUS LOIN **Versions de paquet virtuel**

Si actuellement les paquets virtuels ne peuvent pas avoir de versions, il n'en sera pas forcément toujours ainsi. En effet, **apt** est déjà capable de gérer les versions de paquets virtuels et il est probable que **dpkg** le sera aussi dans un avenir proche. On pourra alors écrire des champs `Provides: libstorable-perl (= 1.7)` pour indiquer qu'un paquet fournit les même fonctionnalités que *libstorable-perl* dans sa version 1.7.

Remplacements : champ *Replaces*

Le champ *Replaces* indique que le paquet contient des fichiers également présents dans un autre paquet, mais qu'il a légitimement le droit de les remplacer. En l'absence de cette précision, **dpkg** échoue en précisant qu'il ne peut pas écraser les fichiers d'un autre paquet (en fait, il est possible de lui forcer la main avec l'option **`--force-overwrite`**). Cela permet d'identifier les problèmes potentiels et contraint le mainteneur à étudier la question avant de choisir d'ajouter ou non ce champ.

L'emploi de ce champ se justifie dans le cas de changements de noms de paquets ou lorsqu'un paquet est intégré dans un autre. Cela se produit également quand le mainteneur a décidé de répartir différemment les fichiers entre différents paquets binaires produits depuis le même paquet source : un fichier remplacé n'appartient plus à l'ancien paquet, mais uniquement au nouveau.

Si tous les fichiers d'un paquet installé ont été remplacés il est considéré comme supprimé. Enfin, ce champ incite aussi **dpkg** à supprimer le paquet remplacé en cas de conflit.

Scripts de configuration

En plus du fichier control, l'archive control.tar.gz de chaque paquet Debian peut contenir un certain nombre de scripts, appelés par **dpkg** à différentes étapes du traitement d'un paquet. La charte Debian détaille longuement les cas possibles en précisant les scripts appelés et les arguments qu'ils reçoivent. Ces séquences peuvent être compliquées puisque si l'un des scripts échoue, **dpkg** essaiera de revenir dans un état satisfaisant en annulant l'installation ou la suppression en cours (tant que cela est possible).

ASTUCE **Diagrammes d'états**

Le projet DebianWomen a réalisé des diagrammes expliquant comment les scripts de configuration sont appelés par **dpkg**. Le lien ci-dessous pointant sur un Wiki, j'ai ajouté le lien vers les diagrammes eux-mêmes (au cas où la page du Wiki disparaîtrait).

- http://women.alioth.debian.org/wiki/index.php/English/MaintainerScripts
- http://www.marga.com.ar/~marga/debian/diagrams/

POUR ALLER PLUS LOIN **Répertoire privé de dpkg**

Tous les scripts de configuration des paquets installés sont stockés dans le répertoire /var/lib/dpkg/info sous la forme d'un fichier préfixé par le nom du paquet. On y trouve également, pour chaque paquet, un fichier d'extension .list contenant la liste des fichiers appartenant au paquet.
Le fichier /var/lib/dpkg/status contient une série de blocs d'informations (au format des fameux en-têtes de courriers électroniques, RFC 822) décrivant le statut de chaque paquet. On y trouve également les informations contenues dans le fichier control des différents paquets installés.

D'une manière générale, le script preinst est exécuté préalablement à l'installation du paquet alors que le postinst la suit. De même, prerm est invoqué avant la suppression du paquet et postrm après. Une mise à jour d'un paquet est équivalente à en supprimer la version précédente puis à installer la nouvelle. Il n'est pas possible de détailler ici tous les scénarios d'actions réussies, mais évoquons quand même les deux plus courants : une installation/mise à jour et une suppression.

ATTENTION **Noms symboliques des scripts**

Les séquences décrites dans cette section font appel à des scripts de configuration aux noms particuliers, comme **`ancien-prerm`** ou **`nouveau-postinst`**. Il s'agit respectivement du script **`prerm`** contenu dans l'ancienne version du paquet (installé avant la mise à jour) et du script **`postinst`** de sa nouvelle version (mis en place par la mise à jour).

Installation et mise à jour

Voici les étapes d'une installation (ou mise à jour) :

1. En cas de mise à jour, on appelle la commande **`ancien-prerm upgrade`** ***`nouvelle-version`***.
2. Pour une mise à jour, **dpkg** exécute **`nouveau-preinst upgrade`** ***`nouvelle-version`***; pour une première installation, il exécute **`nouveau-preinst install`**. Il peut ajouter l'ancienne version en dernier paramètre si le paquet avait déjà été installé et supprimé entre-temps (et que les fichiers de configuration avaient donc été conservés).
3. Les fichiers du nouveau paquet sont décompactés. Si un fichier existait au préalable, il est remplacé mais une copie de sauvegarde est temporairement réalisée.
4. En cas de mise à jour, **dpkg** exécute **`ancien-postrm upgrade`** ***`nouvelle-version`***.
5. **dpkg** met à jour toutes ses données internes (liste de fichiers, scripts de configuration) et supprime les copies de sauvegarde des fichiers remplacés. C'est un point de non retour : **dpkg** ne dispose plus désormais de tous les éléments nécessaires pour revenir à l'état antérieur.
6. **dpkg** va mettre à jour les fichiers de configuration en demandant à l'utilisateur de trancher s'il est incapable de tout gérer automatiquement. Les

détails de cette procédure se trouvent dans la section suivante (voir les explications sur le fichier *conffiles*).

7. Enfin, **dpkg** configure le paquet en exécutant **nouveau-postinst configure *dernière-version-configurée***.

Suppression de paquet

Voici les étapes pour une suppression de paquet :

1. **dpkg** appelle **prerm remove**.
2. **dpkg** supprime tous les fichiers du paquet, à l'exception des fichiers de configuration et des scripts de configuration.
3. **dpkg** exécute **postrm remove**. Tous les scripts de configuration, sauf le postrm, sont effacés. Si l'utilisateur n'a pas demandé la « purge » du paquet, les opérations s'arrêtent là.
4. En cas de purge complète du paquet (demandée avec **dpkg --purge** ou **dpkg -P**), les fichiers de configuration sont supprimés, ainsi qu'un certain nombre de copies (*.dpkg-tmp, *.dpkg-old, *.dpkg-new) et de fichiers temporaires ; **dpkg** exécute ensuite **postrm purge**.

Les 4 scripts évoqués ci-dessus sont complétés par un script config, fourni par les paquets utilisant **debconf** pour obtenir de l'utilisateur des informations de configuration. Lors de l'installation, ce script définit en détail les questions posées par **debconf**. Les réponses sont enregistrées dans la base de données de **debconf** pour référence ultérieure. Le script est généralement exécuté par **apt** avant d'installer un à un tous les paquets afin de regrouper en début de processus toutes les questions posées à l'utilisateur. Les scripts de pré- et post-installation pourront ensuite exploiter ces informations pour effectuer un traitement conforme aux vœux de l'utilisateur.

VOCABULAIRE **La purge, une suppression complète**

Lorsqu'un paquet Debian est supprimé, les fichiers de configuration sont conservés afin de faciliter une éventuelle réinstallation. De même, les données gérées par un démon (comme le contenu de l'annuaire d'un serveur LDAP, ou le contenu de la base de données pour un serveur SQL) sont habituellement conservées.
Pour supprimer toute donnée associée au paquet, il faut procéder à sa « purge » avec la commande **dpkg -P *paquet*** ou **apt-get remove --purge *paquet***.

OUTIL **debconf**

debconf fut créé pour résoudre un problème récurrent chez Debian. Tous les paquets Debian incapables de fonctionner sans un minimum de configuration posaient des questions à l'utilisateur avec des appels à **echo** et **read** dans les scripts shell postinst et similaires. Mais cela impliquait également, lors d'une grosse installation ou mise à jour, de rester à côté de son ordinateur pour renseigner ces requêtes qui pouvaient se produire à tout moment. Ces interactions manuelles ont désormais presque totalement disparu au profit de l'outil **debconf**.
debconf offre de nombreuses caractéristiques intéressantes : il contraint le développeur à spécifier les interactions avec l'utilisateur, il permet de localiser les différentes chaînes de caractères affichées (toutes les traductions sont stockées dans le fichier templates décrivant les interactions), il dispose de différents modules d'affichage pour présenter les questions à l'utilisateur (modes texte, graphique, non interactif), et il permet de créer une base centrale de réponses pour partager la même configuration entre plusieurs ordinateurs... Mais la plus importante est qu'il est maintenant possible de présenter toutes les questions d'un bloc à l'utilisateur avant de démarrer une longue installation ou mise à jour.

Sommes de contrôle, liste des fichiers de configuration

En plus des fichiers déjà cités dans les deux sections précédentes, l'archive control.tar.gz d'un paquet Debian en contient d'autres. Le premier, md5sums, contient la liste des empreintes numériques de tous les fichiers du paquet. Son principal avantage est de permettre à un utilitaire comme **debsums** (que nous verrons plus loin) de vérifier que ces fichiers n'ont pas été modifiés depuis leur installation.

conffiles liste les fichiers du paquet qu'il faudra gérer comme des fichiers de configuration. Un fichier de configuration a cela de particulier qu'il peut être modifié par l'administrateur et que ses changements seront normalement conservés lors d'une mise à jour du paquet.

En effet, dans une telle situation, **dpkg** se comporte aussi intelligemment que possible : si le fichier de configuration standard n'a pas évolué entre les deux versions, il ne fait rien. Sinon, il va essayer de mettre ce fichier à jour. Deux cas sont possibles : soit l'administrateur n'a pas touché à ce fichier de configuration, auquel cas **dpkg** installe automatiquement la nouvelle version disponible, soit le fichier a été modifié, auquel cas **dpkg** demande à l'administrateur quelle version il souhaite utiliser (l'ancienne avec les modifications, ou la nouvelle fournie par le paquet). Pour l'aider à prendre sa décision, **dpkg** lui propose de consulter un « **diff** » présentant les différences entre les deux versions. S'il choisit de conserver l'ancienne version, la nouvelle sera stockée au même emplacement dans un fichier suffixé de .dpkg-new. S'il choisit la nouvelle version, l'ancienne sera conservée dans un fichier .dpkg-old. La dernière possibilité offerte consiste à interrompre momentanément **dpkg** pour éditer le fichier et tenter d'y reprendre les modifications pertinentes (préalablement identifiées grâce au *diff*).

POUR ALLER PLUS LOIN **Éviter les questions sur les fichiers de configuration**

dpkg gère la mise à jour des fichiers de configuration mais interrompt régulièrement ses opérations pour solliciter l'avis de l'administrateur. Cette caractéristique est relativement désagréable pour qui souhaite obtenir une mise à jour non interactive. C'est pourquoi ce programme propose des options permettant de répondre systématiquement la même chose : **--force-confold** conserve l'ancienne version du fichier ; **--force-confnew** utilise la nouvelle version du fichier. Si de plus vous précisez **--force-confdef**, il fera le choix automatique quand c'est possible (c'est-à-dire lorsque le fichier de configuration original n'a pas été modifié) et se rabattra sur **--force-confnew** ou **--force-confold** dans les autres cas.
Ces options s'appliquent à **dpkg**, mais la plupart du temps un administrateur travaillera directement avec les programmes **aptitude** ou **apt-get**. Il est donc nécessaire de connaître la syntaxe qui permet de leur indiquer les options à passer à **dpkg** (leurs interfaces en ligne de commande sont très similaires).

```
# apt-get -o DPkg::Options::="--force-confdef" -o DPkg::options::="--force-confold" dist-upgrade
```

On peut placer ces options directement dans la configuration du programme **apt** plutôt que de les lui spécifier à chaque fois en ligne de commande. Pour cela, il suffit d'écrire la ligne suivante dans le fichier /etc/apt/apt.conf.d/local :

```
DPkg::Options { "--force-confdef"; "--force-confold"; }
```

Intégrer cette option dans le fichier de configuration permettra d'en profiter même dans le cadre d'une interface graphique telle qu'**aptitude**.

Audit des paquets : l'outil **debsums** et ses limites

debsums est un outil intéressant du point de vue de la sécurité puisqu'il permet de trouver facilement quels fichiers installés ont été modifiés (suite par exemple à des interventions malignes). Mais il convient de nuancer fortement cette affirmation : d'abord, tous les paquets Debian ne fournissent pas les empreintes nécessaires au fonctionnement de ce programme (quand elles existent, elles se trouvent dans un fichier md5sums). Rappelons qu'une empreinte est une valeur, généralement numérique (même si elle est codée en hexadécimal), constituant une sorte de signature caractéristique du contenu d'un fichier. Elle est calculée au moyen d'algorithmes (comme le célèbre « MD5 » ou le moins connu « SHA1 ») qui garantissent dans la pratique que (presque) toute modification du fichier, aussi minime soit-elle, entraînera un changement de l'empreinte ; c'est l'« effet d'avalanche ». C'est pourquoi une empreinte numérique permet de vérifier que le contenu d'un fichier n'a pas été altéré. Ces algorithmes ne sont pas réversibles, c'est-à-dire que pour la plupart d'entre eux il est impossible de retrouver un contenu inconnu à partir de la seule empreinte. De récentes découvertes scientifiques tendent à infirmer l'inviolabilité de ces principes, mais cela ne remet pas encore en cause leur usage puisque la création de contenus différents générant la même empreinte semble être très contraignante.

D'autre part, les fichiers md5sums sont stockés sur le disque dur : un intrus consciencieux modifiera ces fichiers pour leur faire refléter les nouvelles sommes de contrôle des fichiers sur lesquels il sera intervenu.

On peut contourner le premier inconvénient en demandant à **debsums** d'utiliser directement un paquet .deb pour effectuer le contrôle au lieu de se reposer sur le fichier md5sums. Mais il faut au préalable télécharger les fichiers .deb correspondants :

```
# apt-get --reinstall -d install 'debsums -l'
[ ... ]
# debsums -p /var/cache/apt/archives -g
```

ASTUCE **apt-get --reinstall**

Il arrive que le système soit endommagé suite à la suppression ou à la modification de fichiers appartenant à un paquet. Le moyen le plus simple de récupérer ces fichiers est alors de réinstaller le paquet concerné. Malheureusement, le système de paquetage considère que ce dernier est déjà installé et refuse poliment de s'exécuter ; l'option **--reinstall** de la commande **apt-get** permet précisément d'éviter cet écueil. La commande ci-dessous réinstalle *postfix* même si ce dernier est déjà présent.

```
#  apt-get --reinstall install postfix
```

Le problème ne se pose pas avec **dpkg**, mais il est rare que l'administrateur emploie directement ce dernier.
Attention, recourir à **apt-get --reinstall** pour restaurer des paquets modifiés au cours d'une attaque ne suffit certainement pas à garantir un système identique à ce qu'il était au préalable.

L'autre souci se contourne de la même manière : il suffit d'effectuer la vérification par rapport à un fichier .deb intègre. Mais cela impose de disposer de tous les fichiers .deb des paquets installés et d'être assuré de leur intégrité. Pour cela, le plus simple est de les reprendre depuis un miroir Debian. Cette opération étant plutôt lente et fastidieuse, ce n'est donc pas une technique à suivre systématiquement dans un but de prévention.

```
# apt-get --reinstall -d install 'grep-status -e 'Status: install ok
   installed' -n -s Package'
[ ... ]
# debsums -p /var/cache/apt/archives --generate=all
```

Attention, cet exemple a employé la commande **grep-status** du paquet *grep-dctrl*, qui n'est pas installé en standard.

debsums n'est pas le seul détecteur de modifications. Le programme *AIDE* (paquet Debian *aide*), par exemple, détecte de manière fiable toute modification inhabituelle par rapport à une image du système préalablement enregistrée et validée.

Structure d'un paquet source

Format

Un paquet source est habituellement constitué de 3 fichiers : un .dsc, un .orig.tar.gz, et un .diff.gz. Ils permettent de créer les paquets binaires (fichiers .deb) du programme à partir de ses codes sources, écrits en langages de programmation.

Le fichier .dsc (*Debian Source Control*, ou contrôle des sources de Debian) est un court fichier texte contenant un en-tête RFC 822 (tout comme le fichier control précédemment étudié) qui décrit le paquet source et indique quels autres fichiers en font partie. Il est signé par son mainteneur, ce qui en garantit l'authenticité.

EXEMPLE **Un fichier .dsc**

```
-----BEGIN PGP SIGNED MESSAGE-----

Format: 1.0
Source: libcarp-datum-perl
Version: 1:0.1.3-1
Binary: libcarp-datum-perl
Maintainer: Raphael Hertzog <hertzog@debian.org>
Architecture: all
Standards-Version: 3.5.2
Build-Depends-Indep: debhelper (>> 3.0.18), perl (>= 5.6.0-16)
Files:
 f4008370407069f3a3adfacb5a39d5dc 39576 libcarp-datum-perl_0.1.3.orig.tar
   .gz
 644c1a68b48e92fa85975c2fbea403c3 8697 libcarp-datum-perl_0.1.3-1.diff.gz

-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.0.6 (GNU/Linux)
```

```
Comment: Pour information voir http://www.gnupg.org

iD8DBQE85DFcvPbGD26BadIRAQqJAJwPaF4q0v/wR6Z5N6s8NbPM9YkfRACff0H9
VGQlouAje7209QHcGrFEGAM=
=4uLo
-----END PGP SIGNATURE-----
```

On notera au passage que le paquet source compte lui aussi des dépendances (*Build-Depends*), totalement distinctes de celles des paquets binaires, puisqu'il s'agit d'outils nécessaires pour compiler le logiciel concerné et construire son paquet binaire.

ATTENTION **Espaces de noms distincts**

Il est important de voir qu'il n'y a pas forcément correspondance entre le nom du paquet source et le nom du ou des paquets binaires qu'il génère — c'est assez facile à comprendre si l'on sait que chaque paquet source peut générer plusieurs paquets binaires. C'est pourquoi le fichier .dsc dispose des champs *Source* et *Binary* pour nommer explicitement le paquet source et stocker la liste des paquets binaires qu'il génère.

Le fichier .orig.tar.gz est une archive contenant les codes sources du programme tels qu'ils ont été fournis par son auteur. Il est demandé aux développeurs de ne pas modifier cette archive afin de pouvoir vérifier facilement la provenance et l'intégrité du fichier (par simple comparaison d'une somme de contrôle) et par respect pour la volonté de certains auteurs.

Le fichier .diff.gz contient quant à lui l'ensemble des modifications apportées par le mainteneur Debian, notamment l'ajout d'un répertoire debian contenant les instructions à exécuter pour construire un paquet Debian.

OUTIL **Décompresser un paquet source**

Si l'on dispose d'un paquet source, on peut employer la commande **dpkg-source** (du paquet *dpkg-dev*) pour le décompacter :

```
$ dpkg-source -x paquet_0.7-1.dsc
```

On peut également employer **apt-get** pour télécharger un paquet source et le décompacter dans la foulée. Il faut cependant disposer de lignes *deb-src* adéquates dans le fichier /etc/apt/sources.list. Ces dernières sont employées pour lister des « sources » de paquets source (c'est-à-dire de serveurs mettant à disposition un ensemble de paquets sources).

```
$ apt-get source paquet
```

Utilité chez Debian

Le paquet source est à la base de tout chez Debian. Tous les paquets Debian proviennent d'un paquet source, et chaque changement dans un paquet Debian est la conséquence d'une modification réalisée au niveau du paquet source. Les mainteneurs Debian travaillent au niveau du paquet source, en connaissant cependant les conséquences de leurs actions sur les paquets binaires. Le fruit de

leur travail se retrouve donc dans les paquets sources disponibles chez Debian : on peut y remonter facilement et tout en découle.

Lorsqu'une nouvelle version d'un paquet (paquet source et un ou plusieurs paquets binaires) parvient sur le serveur Debian, c'est le paquet source qui est le plus important. En effet, il sera ensuite utilisé par tout un réseau de machines d'architectures différentes pour compilation sur les différentes architectures prises en charge par Debian. Le fait que le développeur envoie également un ou plusieurs paquets binaires pour une architecture donnée (en général i386) est relativement secondaire, puisque tout aurait aussi bien pu être généré automatiquement.

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. Mais 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, 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.

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 **Installation d'un paquet avec dpkg**

```
# dpkg -i man-db_2.3.20-18.woody.4_i386.deb
(Lecture de la base de données... 122388 fichiers et répertoires déjà
   installés.)
Préparation du remplacement de man-db 2.3.20-18.woody.4 (en utilisant
   man-db_2.3.20-18.woody.4_i386.deb) ...
Dépaquetage de la mise à jour de man-db ...
Paramétrage de man-db (2.3.20-18.woody.4) ...
```

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 **Dépaquetage et configuration séparée**

```
# dpkg --unpack man-db_2.3.20-18.woody.4_i386.deb
(Lecture de la base de données... 122388 fichiers et répertoires déjà
  installés.)
Préparation du remplacement de man-db 2.3.20-18.woody.4 (en utilisant
  man-db_2.3.20-18.woody.4_i386.deb) ...
Dépaquetage de la mise à jour de man-db ...
# dpkg --configure man-db
Paramétrage de man-db (2.3.20-18.woody.4) ...
```

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 page de manuel de cette commande donneront la liste complète de ces options. L'erreur la plus fréquente, et qui ne manquera de vous concerner tôt ou tard, est la collision de fichiers. Lorsqu'un paquet contient un fichier déjà installé par un autre paquet, **dpkg** refusera de l'installer. Les messages suivants apparaissent alors :

```
Préparation du remplacement de kdepim-libs 4:3.1.0-0woody2 (en utilisant
   .../kdepim-libs_4%3a3.1.1-0woody1_i386.deb) ...
Dépaquetage de la mise à jour de kdepim-libs ...
dpkg : erreur de traitement de /var/cache/apt/archives/kdepim-libs_4%3a3
  .1.1-0woody1_i386.deb (--unpack) :
 tentative de remplacement de « /usr/lib/libkcal.so.2.0.0 », qui
 appartient aussi au paquet libkcal2
```

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.

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 **Suppression puis purge du paquet debian-cd**

```
# dpkg -r debian-cd
(Lecture de la base de données... 122387 fichiers et répertoires déjà
  installés.)
Suppression de debian-cd ...
dpkg : avertissement : lors de la suppression de debian-cd, le
  répertoire
« /etc/debian-cd » n'était pas vide, donc il n'a pas été supprimé.
# dpkg -P debian-cd
(Lecture de la base de données... 122250 fichiers et répertoires déjà
  installés.)
Suppression de debian-cd ...
Purge des fichiers de configuration de debian-cd ...
```

Autres fonctionnalités de dpkg

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.

Pour conclure cette section, signalons qu'un certain nombre d'options de **dpkg** permettent 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 *paquet*** (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 **Diverses requêtes avec dpkg**

```
$ dpkg -L base-passwd
/.
/usr
/usr/sbin
/usr/sbin/update-passwd
/usr/share
/usr/share/doc
/usr/share/doc/base-passwd
/usr/share/doc/base-passwd/README
/usr/share/doc/base-passwd/copyright
/usr/share/doc/base-passwd/changelog.gz
/usr/share/man
/usr/share/man/man8
/usr/share/man/man8/update-passwd.8.gz
/usr/share/base-passwd
/usr/share/base-passwd/group.master
```

```
/usr/share/base-passwd/passwd.master

$ dpkg -S /bin/date
coreutils: /bin/date
$ dpkg -s coreutils
Package: coreutils
Essential: yes
Status: install ok installed
Priority: required
Section: base
Installed-Size: 6732
Maintainer: Michael Stone <mstone@debian.org>
Version: 5.0.91-2
Replaces: textutils, shellutils, fileutils, stat, debianutils (<= 2.3.1)
Provides: textutils, shellutils, fileutils
Pre-Depends: libacl1 (>= 2.2.11-1), libattr1 (>= 2.4.4-1), libc6
   (>= 2.3.2-1)
Conflicts: stat
Description: The GNU core utilities
 This package contains the essential basic system utilities.
 .
 Specifically, this package includes:
 basename cat chgrp chmod chown chroot cksum comm cp csplit cut date dd
 df dir dircolors dirname du echo env expand expr factor false fmt fold
 groups head hostid id install join link ln logname ls md5sum mkdir
 mkfifo mknod mv nice nl nohup od paste pathchk pinky pr printenv
 printf ptx pwd readlink rm rmdir sha1sum seq shred sleep sort split
 stat stty sum sync tac tail tee test touch tr true tsort tty uname
 unexpand uniq unlink users vdir wc who whoami yes
$ dpkg -l 'b*' | head

Souhait=inconnU/Installé/suppRimé/Purgé/H=à garder
| État=Non/Installé/fichier-Config/dépaqUeté/échec-conFig/H=semi-
  installé
|/ Err?=(aucune)/H=à garder/besoin Réinstallation/X=les deux (État,Err:
  majuscule=mauvais)
||/ Nom            Version        Description
+++-==============-==============-
  ========================================
un  babel          <néant>        (aucune description n'est disponible)
un  balsa          <néant>        (aucune description n'est disponible)
un  base           <néant>        (aucune description n'est disponible)
ii  base-files     3.0.2          Debian base system miscellaneous files
ii  base-passwd    3.4.1          Debian Base System Password/Group
  Files
$ dpkg -c /var/cache/apt/archives/cvs_1.11.1p1debian-9_i386.deb
drwxr-xr-x root/root       0 2004-01-06 20:51:06 ./
drwxr-xr-x root/root       0 2004-01-06 20:51:05 ./etc/
drwxr-xr-x root/root       0 2004-01-06 20:51:05 ./etc/cron.weekly/
-rwxr-xr-x root/root    1362 2004-01-06 20:51:05 ./etc/cron.weekly/cvs
drwxr-xr-x root/root       0 2004-01-06 20:51:01 ./usr/
drwxr-xr-x root/root       0 2004-01-06 20:51:07 ./usr/bin/
-rwxr-xr-x root/root  504232 2004-01-06 20:51:07 ./usr/bin/cvs
-rwxr-xr-x root/root   13996 2004-01-06 20:51:03 ./usr/bin/cvsbug
-rwxr-xr-x root/root   17663 2004-01-06 20:51:04 ./usr/bin/rcs2log
[ ... ]
$ dpkg -I /var/cache/apt/archives/cvs_1.11.1p1debian-9_i386.deb
 nouveau paquet Debian, version 2.0.
 taille 1085664 octets : archive de contrôle= 17793 octets.
      21 octets,     1 lignes      conffiles
    6277 octets,   285 lignes   *  config               #!/bin/sh
     972 octets,    24 lignes      control
    8517 octets,   113 lignes      md5sums
    1349 octets,    45 lignes   *  postinst             #!/bin/sh
     525 octets,    20 lignes   *  postrm               #!/bin/bash
     103 octets,     9 lignes   *  preinst              #!/bin/sh
     849 octets,    25 lignes   *  prerm                #!/bin/sh
   31326 octets,   602 lignes      templates
 Package: cvs
```

```
Version: 1.11.1p1debian-9
Section: devel
Priority: optional
Architecture: i386
Depends: libc6 (>= 2.2.4-4), zlib1g (>= 1:1.1.4), debconf
Recommends: netbase (>= 2.08-1), info-browser
Conflicts: cvs-doc
Replaces: cvs-doc (<< 1.11-2)
Provides: cvs-doc
Installed-Size: 2564
Maintainer: Steve McIntyre <93sam@debian.org>
Description: Concurrent Versions System
 CVS is a version control system, which allows you to keep old
 versions of files (usually source code), keep a log of who, when, and
 why changes occurred, etc., like RCS or SCCS.  Unlike the simpler
 systems, CVS does not just operate on one file at a time or one
 directory at a time, but operates on hierarchical collections of
 directories consisting of version controlled files.
 .
 CVS helps to manage releases and to control the concurrent editing of
 source files among multiple authors.  CVS allows triggers to
 enable/log/control various operations and works well over a wide area
 network.
```

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-version 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` ».

Cohabitation avec d'autres systèmes de paquetages

Les paquets Debian ne sont pas les seuls paquetages logiciels exploités dans le monde du logiciel libre. Le principal concurrent est le format RPM de la distribution Red Hat Linux et de ses nombreuses dérivées. C'est la distribution commerciale de référence, il est donc fréquent que des logiciels fournis par des tierces parties soient proposés sous forme de paquets RPM plutôt que Debian.

Dans ce cas, il faut savoir que le programme **`rpm`** existe en paquet Debian ; il est donc possible de manipuler des paquets RPM sur une machine Debian. On

veillera en revanche à limiter ces manipulations à l'extraction des informations du paquet ou à la vérification de son intégrité (avant par exemple d'employer **alien**, mentionné plus bas). Il est en effet déraisonnable de faire appel à **rpm** pour installer un paquet RPM sur un système Debian — RPM emploie ses propres bases de données, distinctes de celles des logiciels natifs (comme **dpkg**). C'est pourquoi il n'est pas possible d'assurer une coexistence saine des deux systèmes de paquetage.

D'autre part, l'utilitaire *alien* permet de convertir des paquets RPM en paquets Debian et vice versa.

```
$ fakeroot alien --to-deb phpMyAdmin-2.0.5-1.noarch.rpm
phpmyadmin_2.0.5-2_all.deb generated
$ ls -s phpmyadmin_2.0.5-2_all.deb
  64 phpmyadmin_2.0.5-2_all.deb
```

Vous constaterez que ce processus est extrêmement simple. Il faut cependant savoir que le paquet généré ne dispose d'aucune information de dépendances, puisque les dépendances des deux formats de paquetage n'ont pas de rapports systématiques. C'est donc à l'administrateur de s'assurer manuellement que le paquet converti fonctionnera correctement, et c'est pourquoi il faut éviter autant que possible les paquets Debian générés ainsi. Heureusement, Debian dispose de la plus grosse collection de paquets logiciels de toutes les distributions et il est probable que ce que vous cherchez y existe déjà.

En consultant la page de manuel de la commande **alien**, vous constaterez également que ce programme gère d'autres formats de paquetages, notamment celui de la distribution Slackware (il s'agit simplement d'une archive .tar.gz).

La stabilité des logiciels déployés grâce à l'outil **dpkg** contribue à la célébrité de Debian. La suite des outils APT, décrite dans le chapitre suivant, préserve cet avantage tout en soulageant l'administrateur de la gestion de l'état des paquets, nécessaire mais difficile.

COMMUNAUTÉ **Encourager l'adoption du .deb**

Si vous employez régulièrement **alien** pour installer des paquets RPM provenant d'un de vos fournisseurs, n'hésitez pas à lui écrire pour exprimer aimablement votre vive préférence pour le format .deb.

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 empressés d'étudier les possibilités.

SOMMAIRE

- Renseigner le fichier **`sources.list`**
- Commande **`apt-get`**
- Commande **`apt-cache`**
- Frontaux : **`aptitude`**, **`synaptic`**, **`gnome-apt`**
- Vérification d'authenticité des paquets
- Mise à jour automatique

MOTS-CLEFS

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

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).

VOCABULAIRE **Source de paquets et paquet source**

Le terme *source* est source d'ambiguïté. Il ne faut pas confondre un paquet source — paquet contenant le code source d'un programme — et une source de paquets — emplacement (site web, serveur FTP, cédérom, répertoire local, etc.) contenant des paquets.

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 a besoin qu'on lui fournisse une « liste de sources de paquets » : 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 du moment — voir la liste dans l'encadré « COMMUNAUTÉ » page 9) 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 :

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 dont vous disposez. 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 — ces sources sont donc gérées un peu différemment. On ajoutera ces entrées à l'aide du petit programme **apt-cdrom**, habituellement invoqué avec le paramètre **add**. Ce dernier demande alors d'insérer le disque dans le lecteur et 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 cédérom en question s'il a besoin de l'un de ses paquets.

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 d'accorder la préférence aux logiciels 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.

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, qui prévoyait à l'origine une interface graphique. Il repose sur une bibliothèque contenant le cœur de l'application, et **`apt-get`** est la première interface — en ligne de commande — développée dans le cadre du projet.

De nombreuses interfaces graphiques sont ensuite apparues en tant que projets extérieurs : **`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 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

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 vous assurer de leur origine et intégrité puis à bien les tester avant d'envisager de les déployer.

Installer un paquet revient à donner les droits *root* à son concepteur, car il décide du contenu de scripts d'initialisation qui sont exécutés sous cette identité. Les paquets officiels Debian sont réalisés par des volontaires cooptés et examinés capables de sceller leurs paquets pour en vérifier l'origine et l'intégrité.

Mais défiez-vous a priori d'un paquet dont l'origine est incertaine et, hors des serveurs officiels du projet Debian : évaluez le degré de confiance que vous accordez au concepteur et vérifiez l'intégrité du paquet.

▸ http://mentors.debian.net

chaque cas, APT installera automatiquement les dépendances nécessaires ou supprimera les paquets dépendant du paquet en cours de désinstallation. L'option **`--purge`** demande une désinstallation complète — les fichiers de configuration sont alors également supprimés.

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
[...]
```

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 se contentera en général d'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
```

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).

Mise à jour

Des mises à jour régulières sont recommandées, car elles mettront en place les derniers correctifs de sécurité. Pour cela, on invoquera **`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 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`**.

B.A.-BA **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 inclura 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 plus récentes 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 lors de l'exécution suivante 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 contenu du répertoire /etc/modutils).

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 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 la source d'où il provient. Chaque source de paquets 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 la distribution (habituellement *stable*, *testing*, *unstable* ou *experimental* pour les distributions 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 la distribution 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

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 1. 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
```

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)`.

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
```

Travailler avec plusieurs distributions

L'outil formidable qu'est **`apt-get`** incite fortement à mettre en place des paquets provenant d'autres distributions. Ainsi, après avoir installé une version *stable*, vous voulez tester un logiciel présent dans *testing* ou *unstable*, sans trop vous éloigner de son état initial.

Même si vous n'êtes pas complètement à l'abri de bogues d'interactions entre les paquets de différentes distributions, **`apt-get`** se révèle fort heureusement très habile pour gérer une telle cohabitation et en minimiser les risques. La meilleure manière de procéder est de préciser toutes les distributions employées dans le fichier /etc/apt/sources.list (à titre personnel, j'y place toujours les trois distributions, mais rappelons que l'utilisation d'*unstable* est réservée aux utilisateurs expérimentés) et de préciser votre distribution de référence avec le paramètre `APT::Default-Release` (voir section « Mise à jour » page 86).

Supposons que *stable* soit votre distribution de référence, mais que *testing* et *unstable* apparaissent également dans votre fichier sources.list. Dans ce cas, vous pouvez employer **`apt-get install paquet/testing`** pour installer un paquet depuis *testing*. Si l'installation échoue parce que certaines dépendances ne sont pas satisfaisables, autorisez-le à satisfaire ces dernières dans *testing* en ajoutant le paramètre **`-t testing`**. Il en ira évidemment de même pour *unstable*.

Dans cette situation, les mises à jour (« **`upgrade`** » et « **`dist-upgrade`** ») ont lieu dans le cadre de *stable* sauf pour les paquets mis à jours depuis une autre distribution : ces derniers suivront les dernières évolutions dans celles-là. Nous

donnons ci-dessous l'explication de ce comportement grâce aux priorités automatiques employées par APT. N'hésitez pas à employer **apt-cache policy** (voir encadré) pour vérifier les priorités indiquées.

Tout est lié au fait que **apt-get** ne considère que les paquets de version supérieure ou égale à la version installée (sauf configuration particulière dans /etc/apt/preferences forçant la priorité de certains paquets au-delà de 1000).

Considérons un premier paquet installé depuis *stable* et qui en est à la version 1, dont la version 2 se trouve dans *testing* et la 3 dans *unstable*. La version installée a une priorité de 100, mais la version disponible dans *stable* (la même) a une priorité de 990 (en tant que version dans la distribution cible). Les paquets de *testing* et *unstable* ont une priorité de 500 (priorité par défaut d'une version non installée). Le vainqueur est donc la version 1 avec une priorité de 990. Le paquet « reste dans *stable* ».

Prenons le cas d'un autre paquet, dont la version 2 a été installée depuis *testing* ; la version 1 est disponible dans *stable* et la 3 dans *unstable*. La version 1 (de priorité 990 — donc inférieure à 1000) est ignorée car plus petite que la version installée. Restent donc les versions 2 et 3, toutes deux de priorité 500. Face à ce choix, APT choisit la version plus récente, celle de la distribution *unstable*. Si vous ne souhaitez pas qu'un paquet installé depuis *testing* puisse migrer vers *unstable* il faut associer une priorité inférieure à 500 (par exemple, 490) aux paquets provenant d'*unstable* en modifiant /etc/apt/preferences :

```
Package: *
Pin: release a=unstable
Pin-Priority: 490
```

ASTUCE **apt-cache policy**

Pour mieux comprendre le mécanisme des priorités, n'hésitez pas à employer **apt-cache policy** pour voir la priorité par défaut associée à chaque source de paquets, et **apt-cache policy *paquet*** pour consulter les priorités des différentes versions disponibles d'un paquet donné.

Commande apt-cache

La commande **apt-cache** permet de consulter un certain nombre d'informations stockées dans la base de données interne d'APT. Ces informations — qui constituent une sorte de *cache* — sont rassemblées depuis les différentes sources données dans le fichier sources.list au cours de l'opération **apt-get update**.

Le programme **apt-cache** permet notamment 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`**. Il est donc relativement facile de réaliser un frontal en employant le code placé dans la bibliothèque.

`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`** mémorise les noms des paquets installés explicitement ou par le jeu des dépendances, ce qui lui permet de supprimer automatiquement ceux qui sont rendus superflus par une mise à jour.

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

`synaptic`, le plus avancé des deux, dispose d'une interface graphique efficace et propre. Ses nombreux filtres prêts à l'emploi permettent de voir rapidement les nouveaux paquets disponibles, les paquets installés, ceux que l'on peut mettre à jour, les paquets obsolètes, etc. En naviguant ainsi dans les différentes listes, on indique progressivement les opérations à effectuer (installer, mettre à jour, supprimer, purger). Un simple clic suffit à valider l'ensemble de ces choix, et toutes les opérations enregistrées sont alors effectuées en une seule passe.



Figure 6–1 Gestionnaire de paquets synaptic

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 provenant de Debian et non altérés en cours de route. En effet, un pirate pourrait tenter d'agir indirectement sur des machines en modifiant un paquet Debian diffusé afin d'y ajouter les instructions de son choix. Lorsque le paquet ainsi modifié sera installé, ces instructions agiront, par exemple afin de dérober les mots de passe. C'est pourquoi Debian offre un moyen de s'assurer que le paquet installé provient bien de son mainteneur et qu'il n'a subi aucune modification par un tiers : il existe un mécanisme de scellement 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 une version *experimental* implémentant cette fonctionnalité 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_2005.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_2005.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/4F368D5D 2005-01-31 Debian Archive Automatic Signing Key
   (2005) <ftpmaster@debian.org>
 Empreinte de la clé = 4C7A 8E5E 9454 FE3F AE1E  78AD F1D5 3D8C 4F36 8
   D5D
```

`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 prochain **`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: Tue, 08 Feb 2005 14:52:51 UTC
  o Description: Debian 3.0 Security Updates
  o Signed by: Debian Archive Automatic Signing Key (2005) <
     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, 30 Dec 2004 23:23:14 UTC
  o Description: Debian 3.0r4 Released 31st December 2004
  o Signed by: Debian Archive Automatic Signing Key (2005) <
     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 ftp://ftp.nerim.net/debian-marillat/ stable main
  o Origin: Christian Marillat/Unofficial Marillat Packages
  o Suite: stable/woody
  o Date: Mon, 07 Feb 2005 16:14:05 UTC
  o Description: Unofficial Multimedia Packages
  * COULDN'T CHECK SIGNATURE BY KEYID: 07DC563D1F41B907
```

SÉCURITÉ **Vérifier l'empreinte 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

```
   * 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
    ftp.nerim.net_debian-marillat_dists_stable_main_binary-i386_Release
    ftp.nerim.net_debian-marillat_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

Les administrateurs de Falcot SA souhaitant automatiser au maximum les mises à jour, les programmes chargés de ces opérations doivent fonctionner sans intervention humaine.

Configuration de `dpkg`

Nous avons déjà vu comment interdire à **`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.

Configuration de debconf

Pour **`debconf`**, la réponse mérite un plus long développement. Dès sa naissance, ce programme fut prévu pour permettre de contrôler la pertinence et le volume des questions posées à l'utilisateur, ainsi que la manière dont elles le seront. C'est pourquoi sa configuration demande la priorité minimale à partir de laquelle **`debconf`** posera une question. Quand il s'interdit d'interroger l'humain, ce programme utilise automatiquement la valeur par défaut définie par le mainteneur du paquet. Il faut encore choisir une interface pour l'affichage des questions (frontal, ou *frontend* en anglais).

Parmi la liste des interfaces possibles, *noninteractive* (non interactive) est très particulière : la choisir désactive toute interaction avec l'utilisateur. Si un paquet désire malgré tout lui communiquer une note d'information, celle-ci sera automatiquement transformée en courrier électronique.

Pour reconfigurer **`debconf`**, on utilise l'outil **`dpkg-reconfigure`** inclus dans le paquet **`debconf`** ; la commande est **`dpkg-reconfigure debconf`**. Il est aussi possible de changer temporairement les choix de configuration effectués à l'aide de variables d'environnement (`DEBIAN_FRONTEND` permet ainsi de changer d'interface, comme expliqué dans la page de manuel `debconf(7)`).

Gestion des interactions en ligne de commande

Finalement, les interactions en ligne de commande des scripts de configuration exécutés par **`dpkg`** sont les plus difficiles à éliminer. Il n'existe en effet aucune solution standard, et aucune réponse n'est meilleure qu'une autre.

La solution généralement employée est de supprimer l'entrée standard (en y redirigeant le contenu de `/dev/null`, par exemple avec la syntaxe ***`commande`*** **`</dev/null`**), ou d'y brancher un flux continu de retours à la ligne. Cette méthode n'est pas fiable à 100 % mais elle permet en général d'accepter les choix par défaut, puisque la plupart des scripts interprètent l'absence de réponse explicite comme une validation de la valeur proposée par défaut.

La combinaison miracle

Si l'on met bout à bout les éléments de configuration exposés dans les sections précédentes, il est possible de rédiger un petit script capable d'effectuer une mise à jour automatique assez fiable.

EXEMPLE **Script pour mise à jour non interactive**

```
export DEBIAN_FRONTEND=noninteractive
yes '' | apt-get -y -o Dpkg::Options::="--force-confdef" -o Dpkg::Options
  ::="--force-confold" dist-upgrade
```

ASTUCE **Mise à jour semi-automatique**

Si vous souhaitez mettre à jour votre système régulièrement mais jugez pénible d'attendre la fin du téléchargement de tous les paquets, il existe une solution : le téléchargement automatique des paquets avec la séquence de commandes **`apt-get update; apt-get -y -d dist-upgrade; apt-get autoclean`** (qu'on pourra placer dans une crontab ; toutes les explications nécessaires se trouvent dans le chapitre 9). L'option **`-d`** ou **`--download-only`** indique à APT qu'il doit se contenter de télécharger les paquets impliqués par l'opération indiquée. Il suffit alors à l'utilisateur d'exécuter la même commande sans cette option pour qu'elle s'exécute immédiatement : en effet, APT disposera déjà des fichiers nécessaires dans sa mémoire *cache*. La commande **`apt-get autoclean`** permet de faire le ménage dans le stock de paquets .deb pour que sa taille n'augmente pas trop (seule la dernière version de chaque paquet est alors conservée).



7



Résolution de problèmes et sources d'information

Pour un administrateur, le plus important est d'être capable de faire face à toute situation, connue ou inconnue. Je vous propose dans ce chapitre un ensemble de méthodes qui vous permettront — je l'espère — d'isoler la cause des problèmes que vous ne manquerez pas de rencontrer, pour mieux les résoudre ensuite.

SOMMAIRE

- Les sources de documentation
  - Les pages de manuel
  - Documentation au format *info*
  - La documentation spécifique
  - Les sites web
  - Les *HOWTO*
- Procédures type
  - Configuration d'un logiciel
  - Surveiller l'activité des démons
  - Demander de l'aide sur une liste de diffusion
  - Signaler un bogue en cas de problème incompréhensible

MOTS-CLEFS

- Documentation
- Résolution de problèmes
- Fichiers logs
- README.Debian
- Manuel
- info

Les sources de documentation

Avant de pouvoir comprendre ce qui se passe réellement en cas de problème, il faut connaître le rôle théorique de chaque programme impliqué. Pour cela, rien de tel que de consulter leur documentation ; mais les sources en étant multiples et dispersées, il convient de les connaître toutes.

Les pages de manuel

Les pages de manuel, relativement austères de prime abord, regroupent pourtant une foule d'informations indispensables. Présentons rapidement la commande qui permet de les consulter. Il s'agit de **`man` *`page-manuel`*** où le nom de la page de manuel est le plus souvent celui de la commande à découvrir. Pour se renseigner sur les options possibles de la commande **`cp`**, on tapera donc la commande **`man cp`** à l'invite de l'interpréteur de commandes (dans une console ou dans un émulateur de terminal par exemple).

B.A.-BA **Interpréteur de commandes**

Un interpréteur de commande — souvent désigné par shell en anglais — est un programme qui exécute les commandes saisies par l'utilisateur ou stockées dans un script. En mode interactif, il affiche une invite (finissant généralement par $ pour un utilisateur normal, ou par # pour l'administrateur) indiquant qu'il est prêt à lire une nouvelle commande.
L'interpréteur de commandes le plus usité est probablement **`bash`** (*Bourne Again SHell*) mais d'autres existent : **`csh`**, **`tcsh`**, **`zsh`**, etc.
La plupart des shells offrent en outre des fonctionnalités d'assistance à la saisie, comme la complétion des noms de commandes ou de fichiers (qu'on active généralement par des tabulations successives), ou encore la gestion d'un historique des commandes déjà exécutées.

ASTUCE **Pages de manuel en français**

Certaines traductions sont regroupées dans le paquet *manpages-fr*. Installez-le pour bénéficier de beaucoup plus de pages traduites ! Il s'agit essentiellement de pages de manuel qui ne peuvent pas être associées à des paquets spécifiques. Les autres sont en effet fournies directement au sein des paquets concernés.
Comme toujours, il faut garder un œil critique sur les traductions : elles peuvent contenir des erreurs ou ne plus être à jour par rapport à la version originale en anglais. En cas de doute, n'hésitez pas à comparer avec la version anglaise, que vous pouvez consulter en exécutant **`LC_ALL=C man` *`page-manuel`***.

Les pages de manuel ne documentent pas uniquement les programmes accessibles en ligne de commande, mais aussi les fichiers de configuration, les appels système, les fonctions de la bibliothèque C, etc. Des collisions de noms surviennent donc. Ainsi, la commande **`read`** de l'interpréteur de commandes s'appelle comme l'appel système `read`. C'est pourquoi les pages de manuel sont classées par section :

1. commandes exécutables depuis l'interpréteur ;
2. appels système (fonctions fournies par le noyau) ;
3. fonctions de bibliothèques (fournies par les bibliothèques système) ;
4. périphériques (sous Unix, ce sont des fichiers spéciaux, habituellement placés sous /dev/) ;
5. fichiers de configuration (formats et conventions) ;
6. jeux ;
7. ensemble de macros et de standards ;
8. commandes d'administration système ;
9. routines du noyau.

Il est possible de préciser la section de la page de manuel recherchée : pour consulter la documentation de l'appel système `read`, on tapera donc **`man 2 read`**. En l'absence d'une section explicite, c'est la première section abritant une page du nom demandé qui sera utilisée. Ainsi, **`man shadow`** renvoie `shadow(5)` parce qu'il n'y a pas de pages de manuel *shadow* dans les sections 1 à 4.

ASTUCE **whatis**

Si vous ne voulez pas consulter la page de manuel complète mais obtenir uniquement une description courte de la commande pour confirmer qu'il s'agit bien de celle que vous cherchez, tapez simplement **`whatis commande`**.

```
$ whatis scp
scp (1)     - secure copy (remote file copy program)
```

Cette description courte — présente dans toutes les pages de manuel — se retrouve dans la section *NAME* (ou *NOM*) qui les débute toutes.

Évidemment, si vous ne connaissez pas les noms des commandes, le manuel ne vous sera pas d'un grand secours. C'est l'objet de la commande **`apropos`**, qui permet de mener une recherche dans les descriptions courtes des pages de manuel (toutes arborent un tel résumé de leurs fonctions en une ligne). **`apropos`** renvoie donc une liste des pages de manuel qui mentionnent le ou les mots-clés demandés. En choisissant bien ceux-ci, on trouvera le nom de la commande recherchée.

EXEMPLE **Retrouver `cp` avec `apropos`**

```
$ apropos "copy file"
cp (1)               - copy files and directories
cpio (1)             - copy files to and from archives
hcopy (1)            - copy files from or to an HFS volume
install (1)          - copy files and set attributes
```

ASTUCE **Naviguer de proche en proche**

Beaucoup de pages de manuel disposent d'un paragraphe *SEE ALSO* ou VOIR AUSSI, généralement à la fin. Il renvoie vers d'autres pages de manuel portant sur des notions ou commandes proches de celle en cours d'examen, ou vers des documentations externes. Il est ainsi possible de retrouver la documentation pertinente même quand le premier choix n'était pas optimal.

La commande **`man`** n'est plus le seul moyen de consulter les pages de manuel, car le programme **`konqueror`** de KDE offre désormais cette possibilité. On trouve encore une interface web, fournie par le paquet **`man2html`** : les pages de manuel sont alors consultables à l'aide d'un navigateur. Sur l'ordinateur d'installation du paquet, utilisez cette URL :

▸ http://localhost/cgi-bin/man/man2html

Cet utilitaire a donc besoin d'un serveur web. C'est pourquoi vous choisirez d'installer ce paquet sur l'un de vos serveurs : tous les utilisateurs du réseau

local bénéficieront du service (y compris les postes non Linux) et vous éviterez de devoir mettre en place un serveur HTTP sur chaque poste. Par ailleurs, si votre serveur est accessible depuis l'extérieur, il peut être souhaitable de restreindre l'accès à ce service aux seuls utilisateurs du réseau local.

CHARTE DEBIAN **Pages de manuel obligatoires**

Debian impose à chaque programme d'être accompagné d'une page de manuel. Si l'auteur amont ne la fournit pas, le développeur Debian rédige en général une page minimaliste qui renverra au moins le lecteur à l'emplacement de la documentation originale.

Documentation au format *info*

Le projet GNU a rédigé les manuels de la plupart de ses programmes au format *info* ; c'est pourquoi de nombreuses pages de manuel renvoient vers la documentation *info* correspondante. Ce format offre quelques avantages mais le programme qui permet de consulter ces documentations est également un peu plus complexe.

Il s'appelle évidemment **`info`**, et on l'invoque en lui passant le nom du « nœud » à consulter. En effet, la documentation *info* a une structure hiérarchique et **`info`** invoqué sans paramètres affichera la liste des « nœuds » disponibles au premier niveau. Habituellement, un nœud porte le nom de la commande correspondante.

Les commandes de navigation dans la documentation ne sont pas très intuitives. La meilleure méthode pour se familiariser avec le programme est sans doute de l'invoquer puis de taper **`h`** (pour *help*, ou demande d'aide) et de suivre les instructions pour apprendre par la pratique. Alternativement, vous pouvez aussi employer un navigateur graphique, beaucoup plus convivial. À nouveau, **`konqueror`** convient ; le paquet **`info2www`** fournit également une interface web.

▸ http://localhost/cgi-bin/info2www

La documentation spécifique

Chaque paquet intègre sa documentation spécifique : même les logiciels les moins bien documentés disposent en général au moins d'un fichier README (lisez-moi) contenant quelques informations intéressantes et/ou importantes. Cette documentation est installée dans le répertoire /usr/share/doc/<*paquet*> (où *paquet* représente le nom du paquet). Si elle est très volumineuse, elle peut ne pas être intégrée au paquet du programme mais constituer son propre paquet, alors intitulé *`paquet`*`-doc`. Le paquet du programme recommande en général le paquet de documentation pour le mettre en exergue.

Dans le répertoire /usr/share/doc/<*paquet*> se trouvent également quelques fichiers fournis par Debian et complétant la documentation du point de vue des particularités ou améliorations du paquet par rapport à une installation traditionnelle du logiciel. Le fichier README.Debian signale ainsi toutes les adaptations effectuées pour être en conformité avec la charte Debian. Le fichier

changelog.Debian.gz permet quant à lui de suivre les modifications apportées au paquet au fil du temps : il est très utile pour essayer de comprendre ce qui a changé entre deux versions installées et qui n'ont apparemment pas le même comportement. Enfin, on trouve parfois un fichier NEWS.Debian.gz documentant les changements majeurs du programme qui peuvent concerner directement l'administrateur.

Les sites web

Dans la majorité des cas, un logiciel libre dispose d'un site web pour le diffuser et fédérer la communauté de ses développeurs et utilisateurs. Ces sites regorgent souvent d'informations pertinentes sous différentes formes : les documentations officielles, des foires aux questions (FAQ), les archives des listes de diffusion relatives au logiciel, etc. Fréquemment, un problème rencontré aura déjà fait l'objet de nombreuses questions ; la FAQ ou les archives de l'une des listes de diffusion en abriteront alors la solution. Une bonne maîtrise d'un moteur de recherche s'avérera précieuse pour trouver rapidement les pages pertinentes (en restreignant éventuellement la recherche au domaine ou sous-domaine Internet dédié au logiciel). Si le moteur renvoie trop de pages ou si les réponses ne correspondent pas à ce qui est attendu, l'ajout du mot clé **`debian`** permet de restreindre les réponses en ciblant les informations concernant les utilisateurs de ce système.

Si vous ne connaissez pas l'adresse du site web du logiciel, il y a différents moyens de l'obtenir. Vérifiez d'abord la description du paquet : elle contient fréquemment un pointeur sur le site web officiel du logiciel. Si aucune URL n'y est indiquée, il convient alors d'examiner /usr/share/doc/<*paquet*>/copyright. Le mainteneur Debian y mentionne en effet l'endroit où il a récupéré les codes sources du programme, et il est probable qu'il s'agisse justement du site web en question. Si à ce stade votre recherche est toujours infructueuse, il faut consulter un annuaire de logiciels libres tel que Freshmeat.net ou encore directement effectuer une recherche sur un moteur comme Google.

▸ http://freshmeat.net

ASTUCE **De l'erreur à la solution**

Si le logiciel renvoie un message d'erreur très spécifique, saisissez-le dans un moteur de recherche (entre apostrophes doubles « " » pour ne pas rechercher les mots individuellement, mais l'expression complète) : dans la majorité des cas, les premiers liens renvoyés contiendront la réponse que vous cherchez.

Les *HOWTO*

Un *HOWTO* (comment faire) est une documentation décrivant concrètement, étape par étape, comment atteindre un but prédéfini. Les buts couverts sont relativement variés mais souvent techniques : mettre en place l'*IP Masquerading* (masquage d'IP), configurer le *RAID logiciel*, installer un serveur Samba, etc. Ces documents essaient souvent de couvrir l'ensemble des problématiques susceptibles de se produire dans la mise en œuvre d'une technologie donnée.

Les *HOWTO* sont gérés par le *Linux Documentation Project* (projet de documentation Linux, LDP), dont le site web abrite donc l'ensemble de ces documents :

▸ http://www.tldp.org

DÉCOUVERTE **Documentation en français**

Il arrive fréquemment qu'une documentation traduite en français soit disponible dans un paquet séparé portant le nom du paquet correspondant suivi de `-fr` (code ISO officiel de la langue française).
Ainsi, le paquet *apt-howto-fr* contient la traduction française du *HOWTO* dédié à *APT*. De même, les paquets *quick-reference-fr* (référence rapide) et *debian-reference-fr* (référence Debian) sont les versions françaises des guides de référence Debian (initialement rédigés en anglais par Osamu Aoki).

Pour les consulter localement, il suffit d'installer les paquets *doc-linux-html* et *doc-linux-fr-html*. Les versions HTML seront alors disponibles dans le répertoire /usr/share/doc/HOWTO.

Restez critique en lisant ces documents. Ils sont fréquemment vieux de plusieurs années ; leurs informations seront donc parfois obsolètes. Ce phénomène est encore plus fréquent pour leurs traductions, puisque les mises à jour de ces dernières ne sont ni systématiques ni instantanées après la publication d'une nouvelle version du document original — ou des joies de travailler dans un environnement bénévole et sans contrainte...

Procédures type

Cette section vise à présenter quelques conseils génériques portant sur certaines opérations qu'un administrateur est souvent amené à effectuer. Ces procédures ne couvriront évidemment pas tous les cas de figure de manière exhaustive, mais pourront servir de points de départ pour les cas les plus ardus.

Configuration d'un logiciel

Face à un paquet inconnu que l'on souhaite configurer, il faut procéder par étapes. En premier lieu, il convient de lire ce que le responsable du paquet peut avoir d'intéressant à signaler. La lecture de /usr/share/doc/<*paquet*>/README.Debian permet en effet de découvrir les aménagements spécifiques prévus pour faciliter l'emploi du logiciel concerné. C'est parfois indispensable pour comprendre des différences avec le comportement original du logiciel, tel qu'il est décrit dans des documentations généralistes comme les *HOWTO*. Parfois, ce fichier détaille aussi les erreurs les plus courantes afin de vous éviter de perdre du temps sur des problèmes classiques.

Ensuite, il faut consulter la documentation officielle du logiciel — référez-vous à la section précédente pour identifier les différentes sources de documentation. La commande **`dpkg -L` *`paquet`*** donne la liste des fichiers inclus dans le paquet ; on pourra ainsi repérer rapidement les documentations disponibles (ainsi que les fichiers de configuration, situés dans /etc/). **`dpkg -s` *`paquet`*** produit les en-têtes du paquet et donne les éventuels paquets recommandés ou suggérés : on pourra y trouver de la documentation ou un utilitaire facilitant la configuration du logiciel.

Enfin, les fichiers de configuration sont souvent auto-documentés par de nombreux commentaires explicatifs détaillant les différentes valeurs possibles de chaque paramètre de configuration — à tel point qu'il suffit parfois de choisir la ligne à activer parmi celles proposées. Dans certains cas, des exemples de fichiers de configuration sont fournis dans le répertoire /usr/share/doc/<*paquet*>/examples/. Ils pourront alors servir de base à votre propre fichier de configuration.

CHARTE DEBIAN **Emplacement des exemples**

Tout exemple doit être installé dans le répertoire /usr/share/doc/<*paquet*>/examples. Il peut s'agir d'un fichier de configuration, d'un code source de programme (exemple d'emploi d'une bibliothèque), ou d'un script de conversion de données que l'administrateur pourrait employer dans certains cas (comme pour initialiser une base de données). Si l'exemple est spécifique à une architecture, il convient de l'installer dans /usr/lib/<*paquet*>/examples et de créer un lien pointant sur lui dans le répertoire /usr/share/doc/<*paquet*>/examples

Surveiller l'activité des démons

Un démon complique un peu la compréhension des situations, puisqu'il n'interagit pas directement avec l'administrateur. Pour vérifier son fonctionnement, il est donc nécessaire de le tester, par exemple avec une requête HTTP pour le démon Apache (un serveur web).

Pour permettre ces vérifications, chaque démon garde généralement des traces de tout ce qu'il a fait ainsi que des erreurs qu'il a rencontrées — on les appelle logs (journaux de bord du système). Les logs sont stockés dans /var/log/ ou l'un de ses sous-répertoires. Pour connaître le nom précis du fichier de log de chaque démon, on se référera à la documentation. Attention, un seul test ne suffit pas toujours s'il ne couvre pas la totalité des cas d'utilisation possibles : certains problèmes ne se manifestent que dans certaines circonstances particulières.

OUTIL **Le démon `syslogd`**

`syslogd` est particulier : il collecte les traces (messages internes au système) qui lui sont envoyées par les autres programmes. Chaque trace est associée à un sous-système (courrier électronique, noyau, authentification, etc.) et à une priorité, deux informations que **`syslogd`** consulte pour décider de son traitement : la trace peut être consignée dans divers fichiers de logs et/ou envoyée sur une console d'administration. Le détail des traitements effectués est défini par le fichier de configuration /etc/syslog.conf (documenté dans la page de manuel éponyme).
Certaines fonctions C, spécialisées dans l'envoi de traces, facilitent l'emploi du démon **`syslogd`**. Certains démons gèrent toutefois eux-mêmes leurs fichiers de logs (c'est par exemple le cas de **`samba`**, le serveur implémentant les partages Windows sur Linux).

B.A.-BA **Démon**

Un démon (*daemon*) est un programme non invoqué explicitement par l'utilisateur et qui reste en arrière-plan, attendant la réalisation d'une condition avant d'effectuer une tâche. Beaucoup de logiciels serveurs sont des démons — terme qui explique la lettre « d » souvent présente à la fin de leur nom (**`talkd`**, **`telnetd`**, **`httpd`**, etc.).

Toute démarche préventive commence par une consultation régulière des logs des serveurs les plus importants. Vous pourrez ainsi diagnostiquer les problèmes avant même qu'ils ne soient signalés par des utilisateurs mécontents. En effet, ceux-ci attendent parfois qu'un problème se répète plusieurs jours pour en faire part. On pourra faire appel à un utilitaire spécifique pour analyser le contenu des fichiers de logs les plus volumineux. On trouve de tels utilitaires pour les serveurs web (citons **`analog`**, **`awstats`**, **`webalizer`** pour Apache), pour les serveurs FTP, pour les serveurs *proxy/cache*, pour les pare-feu, pour les serveurs de courrier électronique, pour les serveurs DNS, et même pour les serveurs d'impression. Certains de ces utilitaires fonctionnent de manière modulaire et permettent d'analyser plusieurs types de fichiers de logs. C'est le cas de **`lire`** ou

encore de **modlogan**. D'autres outils, comme **logcheck** (logiciel abordé dans le chapitre 9), permettent de scruter ces fichiers à la recherche d'alertes à traiter.

Demander de l'aide sur une liste de diffusion

Si vos diverses recherches ne vous ont pas permis de venir à bout d'un problème, il est possible de se faire aider par d'autres personnes, peut-être plus expérimentées. C'est tout l'intérêt de la liste de diffusion debian-user-french@lists.debian.org. Comme toute communauté, elle a ses règles qu'il convient de respecter. La première est de vérifier que le problème qui vous concerne n'apparaît pas dans sa foire aux questions (voir lien en bas de paragraphe). Il faut également le rechercher dans les archives récentes de la liste avant de poser sa question.

- http://wiki.debian.net/?DebianFrench
- http://lists.debian.org/debian-user-french/

Ces deux conditions remplies, vous pouvez envisager de décrire votre problème sur la liste de diffusion. Incluez le maximum d'informations pertinentes : les différents essais effectués, les documentations consultées, comment vous avez diagnostiqué le problème, les paquets concernés ou susceptibles d'être en cause. Consultez le système de suivi de bogues de Debian (BTS, ou « *Bug Tracking System* ») à la recherche d'un problème similaire et mentionnez le résultat de cette recherche en fournissant les liens vers les bogues trouvés. Rappelons que toute exploration du BTS débute à la page ci-dessous :

- http://www.debian.org/Bugs/index.fr.html

Plus vous aurez été courtois et précis, plus grandes seront vos chances d'obtenir une réponse — ou du moins des éléments de réponse. Si vous recevez des informations intéressantes par courrier privé, pensez à faire une synthèse publique des réponses reçues afin que tout le monde puisse en profiter et que les archives de la liste, dûment explorées par divers moteurs de recherche, donnent directement la réponse à ceux qui se poseront la même question que vous.

ASTUCE **Lire une liste sur le Web**

Pour des listes de diffusion à haut volume comme debian-user-french@lists.debian.org (*duf* pour les intimes), il peut être intéressant de les parcourir comme un forum de discussion (*newsgroup*). Gmane.org permet justement la consultation des listes Debian sous ce format. La liste francophone précitée est ainsi disponible à l'adresse suivante :

- http://dir.gmane.org/gmane.linux.debian.user.french

B.A.-BA **La Netiquette s'applique**

D'une manière générale, pour toutes les correspondances électroniques sur des listes de diffusion, il convient de respecter les règles de la Netiquette. On regroupe sous ce terme un ensemble de règles de bon sens, allant de la politesse aux erreurs à ne pas commettre.

- http://web.ccr.jussieu.fr/ccr/Netiquette.html

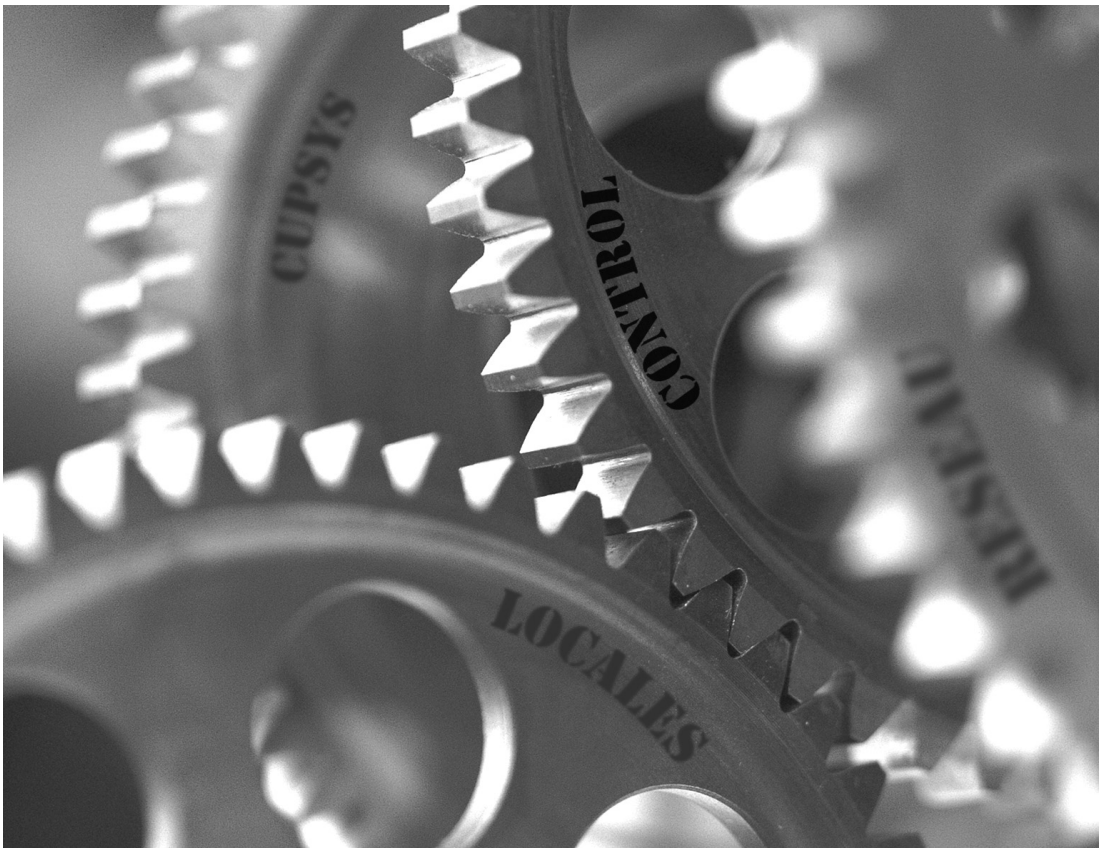
Signaler un bogue en cas de problème incompréhensible

Si tous vos efforts pour résoudre un problème restent vains, il est possible que celui-ci ne relève pas de votre responsabilité et qu'il s'agisse en fait d'un bogue dans le programme récalcitrant. Dans ce cas, la procédure à adopter est de le signaler à Debian ou directement aux auteurs du logiciel. Pour cela, il convient d'isoler le mieux possible le problème et de créer une situation de test minimale qui permette de le reproduire. Si vous savez quel programme est la cause apparente du problème, il est possible de retrouver le paquet concerné à l'aide de la commande **dpkg -S *fichier_en_cause***. La consultation du système de suivi de bogues (`http://bugs.debian.org/`*paquet*) permettra de vérifier que ce bogue n'a pas encore été répertorié. On pourra alors envoyer son propre

rapport de bogue à l'aide de la commande **`reportbug`** — en incluant le maximum d'informations, en particulier la description complète du cas minimal de test qui permet de reproduire le bogue.

Les éléments du présent chapitre constituent un moyen de résoudre, efficacement, les embarras que les chapitres suivants pourraient susciter. Faites-y donc appel aussi souvent que nécessaire !

8



Configuration de base : réseau, comptes, impression...

Un ordinateur nouvellement installé par **`debian-installer`** se veut aussi fonctionnel que possible, mais de nombreux services restent à paramétrer. Par ailleurs, il est bon de savoir comment changer certains éléments de configuration définis lors de l'installation initiale.

SOMMAIRE

- Francisation du système
- Configuration du réseau
- Attribution et résolution des noms
- Base de données des utilisateurs et des groupes
- Création de comptes
- Environnement des interpréteurs de commandes
- Configuration de l'impression
- Configuration du chargeur d'amorçage
- Autres configurations : synchronisation, logs, partages...
- Compilation d'un noyau
- Installation d'un noyau

MOTS-CLEFS

- Configuration
- Francisation
- Locales
- Réseau
- Résolution de noms
- Utilisateurs
- Groupes
- Comptes
- Interpréteur de commandes
- Impression
- Chargeur de démarrage
- Compilation de noyau

Ce chapitre passe en revue tout qui relève de ce que l'on peut appeler la « configuration de base » : réseau, langue et « locales », utilisateurs et groupes, impression, points de montage, etc.

Francisation du système

Il est probable que l'ordinateur fonctionne déjà en français si l'installation a été menée dans cette langue. Mais il est bon de savoir ce que l'installateur a fait à cette fin pour pouvoir le changer plus tard si le besoin s'en faisait sentir.

OUTIL **La commande `locale` pour afficher la configuration courante**

La commande **`locale`** permet d'afficher un résumé de la configuration courante des différents paramétrages de la locale (format des dates, des nombres, etc.), présenté sous la forme d'un ensemble de variables d'environnement standards et dédiées à la modification dynamique de ces réglages.

Définir la langue par défaut

Le paquet *locales* rassemble les éléments nécessaires au bon fonctionnement des « localisations » des différentes applications. Lors de son installation, ce paquet pose quelques questions afin de choisir les langues prises en charge. Il est à tout moment possible de revenir sur ces choix en exécutant **`dpkg-reconfigure locales`**.

On demande d'abord de choisir toutes les « locales » à prendre en charge. La sélection de toutes les locales françaises (c'est-à-dire celles débutant par « `fr_FR` ») est un choix raisonnable. N'hésitez pas à sélectionner d'autres locales si

CULTURE **Jeux de caractères**

À chaque locale sont normalement associés un « jeu de caractères » (ensemble des caractères possibles) et un « encodage » (manière de représenter les caractères pour l'ordinateur) de prédilection.

L'encodage *ISO-8859-1* (ou « Latin 1 ») avait cours en France. Mais, pour des raisons historiques, il n'incluait pas certains caractères (e dans l'o, y tréma majuscule, symbole euro). C'est pourquoi *ISO-8859-15* (ou « Latin 9 », voire « Latin 0 ») a vu le jour. Entre autres, il remplace respectivement l'ancien symbole monétaire international (un rond à quatre branches), « $1/2$ », « $1/4$ » et « $3/4$ » par le symbole de l'euro, « œ », « Œ » et « Ÿ ». Ces deux jeux sont limités à 256 caractères, puisqu'un seul octet (de 8 bits) représente chacun d'entre eux. D'autres langues utilisaient d'autres jeux de caractères, ou d'autres encodages, de la famille « Latin » ou non.

Pour travailler avec des langues étrangères, il fallait donc régulièrement jongler avec différents encodages et jeux de caractères. De surcroît, la rédaction de documents multilingues posait parfois des problèmes quasi insurmontables. Unicode (super catalogue de presque tous les systèmes d'écriture des langues du monde) fut créé pour contourner ce problème. Son encodage particulier UTF-8 conserve tous les 128 symboles ASCII (codés sur 7 bits) mais traite différemment les autres caractères. Ces derniers sont précédés par une séquence de bits d'« échappement » plus ou moins longue. Cela permet de représenter tous les caractères Unicode sur un ou plusieurs octets, selon le besoin.

Petit à petit, l'ensemble des applications migre vers un usage généralisé d'UTF-8. Cela s'effectue d'autant plus facilement que cet encodage est la norme pour les documents XML.

la machine héberge des utilisateurs étrangers. Cette liste des locales connues du système est stockée dans le fichier /etc/locale.gen. Il est possible d'intervenir sur ce fichier à la main, mais il faut penser à exécuter **`locale-gen`** après chaque modification ; cela génère les fichiers nécessaires au bon fonctionnement des locales éventuellement ajoutées, tout en supprimant les fichiers obsolètes.

La seconde question, intitulée « Jeu de paramètres régionaux par défaut », requiert une locale par défaut. Elle doit sa formulation au fait que la locale ne gouverne pas seulement la langue des textes mais aussi le format de présentation des nombres, des dates et des heures, des sommes monétaires ainsi que le mode de comparaison alphabétique (afin de tenir compte des caractères accentués). Le choix recommandé en France est « `fr_FR@euro` ». Les Belges francophones préféreront « `fr_BE@euro` », les Luxembourgeois « `fr_LU@euro` », les Suisses « `fr_CH` » et les Canadiens « `fr_CA` ». Le fichier /etc/environment est alors modifié pour renseigner la locale par défaut dans la variable d'environnement `LANG`.

EN COULISSES **Qui emploie `/etc/environment` ?**

Le fichier /etc/environment sert aux programmes **`login`**, **`gdm`**, ou encore **`ssh`** pour créer leurs variables d'environnement.
Ces applications n'effectuent pas cela directement, mais via un module PAM (pam_env.so). PAM (*Pluggable Authentication Module*, ou module d'authentification connectable) est une bibliothèque modulaire centralisant les mécanismes d'authentification, d'initialisation de sessions et de gestion des mots de passe. Voir page 214 pour un exemple de configuration de PAM.

Configurer le clavier en mode console

Le paquet *console-data* contient les différentes dispositions de clavier accessibles au paquet *console-tools* pour configurer la console — c'est-à-dire l'ordinateur quand il est en mode texte plein écran comme au démarrage, par opposition au mode graphique. La commande **`dpkg-reconfigure console-data`** permet de revenir à tout moment sur la disposition de clavier choisie pour le mode console.

À la première question, et sauf cas très particulier, optez pour « Choisir un codage clavier pour votre architecture ». Répondez ensuite en fonction de votre clavier réel. En règle générale, il faudra choisir un clavier « azerty » puis une disposition de touches « french » et enfin la variante « With Euro (Latin 9) ».

Configurer le clavier en mode graphique

Ce réglage relève de la configuration du serveur XFree86 (voir le chapitre 12). Vous répondrez « xfree86 » à « Jeu de règles XKB » et probablement « pc105 » à « Type de clavier ». La « disposition » d'un clavier azerty français est « fr ». Vous pourrez laisser vide la « variante » du clavier.

POUR ALLER PLUS LOIN **Options du clavier**

Quelques options du clavier sous XFree86 pouvant faciliter la saisie dans certaines circonstances, je déconseille de toutes les rejeter. Ainsi, `altwin:left_meta_win,compose:rwin` transforme la touche Windows de gauche en modificateur Meta, ce qui permet de l'employer pour des raccourcis clavier. La touche Windows de droite devient alors la touche Compose, et permet de saisir des caractères spéciaux en combinant des caractères simples. Taper successivement **`Compose ' e`** produira ainsi un e accent aigu (« é »). Toutes ces combinaisons sont décrites dans le fichier /usr/X11R6/lib/X11/locale/iso8859-15/Compose.
La configuration n'est cependant pas figée : XFree86 permet à tout instant de modifier ou d'adapter la configuration du clavier — qu'il s'agisse de tout le clavier ou de quelques options affectant le comportement des touches particulières. C'est pourquoi GNOME et KDE, entre autres, peuvent fournir un panneau de configuration Clavier dans leurs préférences.

Configuration du réseau

Le réseau étant automatiquement réglé lors de l'installation initiale, le fichier /etc/network/interfaces contient déjà une configuration valide. Une ligne débutant par `auto` donne la liste des interfaces à configurer automatiquement au démarrage. Souvent, on y trouve `eth0`, qui correspond à la première carte Ethernet.

Interface Ethernet

Si l'ordinateur dispose d'une carte réseau Ethernet, il faut configurer le réseau qui y est associé en optant pour l'une de deux méthodes. La configuration dynamique par DHCP, la plus simple, nécessite la présence d'un serveur DHCP sur le réseau local. Elle peut indiquer un nom d'hôte souhaité, ce qui correspond au paramètre facultatif `hostname` dans l'exemple ci-dessous. Le serveur DHCP renvoie alors les paramètres de configuration du réseau qui conviennent.

EXEMPLE **Configuration par DHCP**

```
auto eth0
iface eth0 inet dhcp
  hostname arrakis
```

Une configuration « statique » doit mentionner de manière fixe les paramètres du réseau. Cela inclut au minimum l'adresse IP et le masque de sous-réseau, parfois

B.A.-BA **Rappels réseau essentiels (Ethernet, adresse IP, sous-réseau, broadcast…)**

La majorité des réseaux locaux actuels sont des réseaux Ethernet qui fonctionnent par trames, c'est-à-dire que les données y circulent de manière non continue, par petits blocs. Le débit varie de 10 Mbit/s pour les cartes Ethernet les plus anciennes, à 1000 Mbit/s pour la plus récente génération (100 Mbit/s étant le débit le plus fréquent à l'heure actuelle). Les câbles correspondants sont connus sous le nom de 10baseT, 100baseT ou 1000baseT, dits en « paire torsadée » (*twisted pair*), dont chaque extrémité est munie d'un connecteur RJ45.

Une adresse IP est un numéro employé pour identifier une interface réseau d'un ordinateur sur le réseau local ou sur l'Internet. Ce numéro se code sur 32 bits et se représente habituellement comme 4 nombres séparés par des points (ex : `192.168.0.1`), chaque nombre pouvant varier de 0 à 255 (représentant ainsi 8 bits de données).

Un masque de sous-réseau (*netmask*) définit par son codage en binaire quelle portion d'une adresse IP correspond au réseau — le reste y spécifiant l'identifiant de la machine. Dans l'exemple de configuration statique donné ici, le masque de sous-réseau `255.255.255.0` (24 « 1 » suivis de 8 « 0 » en représentation binaire) indique que les 24 premiers bits de l'adresse IP correspondent à l'adresse réseau, les 8 derniers relevant alors du numéro de machine.

L'adresse de réseau est une adresse IP dont la partie décrivant le numéro de machine est à zéro. On décrit souvent la plage d'adresses IP d'un réseau complet par la syntaxe *a.b.c.d/e* où *a.b.c.d* est l'adresse réseau et *e* le nombre de bits affectés à la partie réseau dans une adresse IP. Le réseau de l'exemple s'écrirait ainsi : `192.168.0.0/24`.

Un routeur est une machine reliant plusieurs réseaux entre eux. Tout le trafic y parvenant est réorienté sur le bon réseau. Pour cela, le routeur analyse les adresses IP destinataires. Le routeur est souvent qualifié de passerelle ; il s'agit alors habituellement d'une machine qui permet de sortir d'un réseau local (vers un réseau étendu comme l'Internet).

L'adresse de *broadcast*, spéciale, permet de joindre tous les postes du réseau. Presque jamais « routée », elle ne fonctionne donc que sur le réseau considéré. Concrètement, cela signifie qu'un paquet de données adressé au *broadcast* ne franchit jamais un routeur.

les adresses de réseau et de *broadcast*. Un éventuel routeur vers l'extérieur sera précisé en tant que passerelle.

EXEMPLE **Configuration statique**

```
auto eth0
iface eth0 inet static
  address 192.168.0.3
  netmask 255.255.255.0
  broadcast 192.168.0.255
  network 192.168.0.0
  gateway 192.168.0.1
```

Interface PPP

Une connexion point à point (PPP) établit une connexion intermittente, le plus souvent grâce à un modem. Selon qu'il est téléphonique (sur le réseau téléphonique commuté RTC) ou ADSL, les technologies correspondantes et la configuration nécessaire diffèrent fortement.

Connexion par modem téléphonique

Une connexion par modem téléphonique requiert un compte chez un fournisseur d'accès, comprenant numéro de téléphone, identifiant, mot de passe et, parfois, protocole d'authentification employé. On la configurera à l'aide de l'utilitaire **`pppconfig`** du paquet Debian éponyme. Par défaut, il utilise la connexion *provider* (fournisseur d'accès). En cas de doute sur le protocole d'authentification, choisissez *PAP* : il est proposé par la majorité des fournisseurs d'accès.

Après configuration, il est possible de se connecter par la commande **`pon`** (à laquelle on fournira le nom de la connexion si la valeur par défaut — *provider* — ne convient pas). On coupera la connexion par la commande **`poff`**.

OUTIL **Connexion à la demande avec `diald`**

`diald` est un service de connexion à la demande, établissant automatiquement une connexion dès que nécessaire en détectant qu'un paquet IP doit sortir et l'interrompant après une période d'inactivité.

Connexion par modem ADSL

Le terme générique de « modem ADSL » recouvre des périphériques aux fonctionnements très différents. Les modems les plus simples à employer avec Linux sont ceux qui disposent d'une interface Ethernet.

ASTUCE **Démarrer `ppp` via `init`**

Les connexions PPP par ADSL sont par définition intermittentes. Comme elles ne sont pas facturées à la durée, la tentation est grande de les garder toujours ouvertes ; un moyen simple est de les faire démarrer par le processus **`init`**. Il suffit pour cela d'ajouter la ligne ci-dessous au fichier /etc/inittab ; toute interruption provoquera alors immédiatement l'invocation d'une nouvelle connexion par **`init`**.

```
adsl:2345:respawn:/usr/sbin/pppd call dsl-provider
```

La plupart des connexions ADSL subissent une déconnexion quotidienne, mais cette méthode permet de réduire la durée de la coupure.

MATÉRIEL **Alcatel Speedtouch**

Le modem USB Alcatel *Speedtouch* est assez populaire et fonctionne d'une manière particulière. Il dispose donc d'un paquet Debian spécifique : *speedtouch*, employant le protocole PPPOA (*Point-to-Point Protocol Over ATM*, ou protocole point à point sur ATM).

Modem fonctionnant avec PPPOE

La majorité des modems fonctionnent avec le protocole PPPOE (*Point-to-Point Protocol Over Ethernet*, ou protocole point à point sur Ethernet). L'utilitaire **pppoeconf** (du paquet éponyme) configurera la connexion. Pour cela, il modifiera le fichier /etc/ppp/peers/dsl-provider avec les paramètres fournis et enregistrera les informations d'authentification dans les fichiers /etc/ppp/pap-secrets et /etc/ppp/chap-secrets. Il est recommandé d'accepter toutes les modifications qu'il propose.

Cette configuration mise en place, on pourra démarrer la connexion ADSL par la commande **pon dsl-provider** et la stopper avec **poff dsl-provider**.

Modem fonctionnant avec PPTP

Le protocole PPTP (*Point-to-Point Tunneling Protocol*, ou protocole point à point par tunnel) est une invention de Microsoft. Déployé aux débuts de l'ADSL, il a rapidement été remplacé par PPPOE. Si ce protocole vous est imposé, voyez au chapitre 10 la section relative aux réseaux privés virtuels, détaillant PPTP.

Modem fonctionnant avec DHCP

Ce mode de fonctionnement ne fait plus du tout intervenir PPP. Le modem est connecté à l'ordinateur par un câble Ethernet (croisé) et fait office de serveur DHCP. Il suffit alors à l'ordinateur de configurer une connexion réseau par DHCP ; le modem s'inscrit automatiquement comme passerelle par défaut et prend en charge le travail de routage (c'est-à-dire qu'il gère le trafic réseau entre l'ordinateur et l'Internet).

Ce mode est notamment employé par la Freebox, le modem ADSL fourni par le fournisseur d'accès Free.

B.A.-BA **Câble croisé pour une connexion Ethernet directe**

Lorsqu'on relie un ordinateur à un réseau local, on branche habituellement un câble (droit ou décroisé) entre la carte réseau et un répétiteur ou un commutateur. Un tel câble ne fonctionne cependant pas pour relier deux ordinateurs directement (i.e. sans répétiteur/commutateur intermédiaire) ; un câble croisé est alors nécessaire.

B.A.-BA **/proc et /sys, systèmes de fichiers virtuels**

Les arborescences /proc et /sys sont gérées par des systèmes de fichiers « virtuels ». Il s'agit en fait d'un moyen pratique de récupérer des informations depuis le noyau (en lisant des fichiers virtuels) et de lui en communiquer (en écrivant dans des fichiers virtuels).
/sys est tout particulièrement prévu pour donner accès à des objets internes du noyau, en particulier ceux qui représentent les différents périphériques du système. Le noyau peut ainsi partager de nombreuses informations : l'état de chaque périphérique (endormi ou pas), s'agit-il d'un périphérique amovible, etc. Signalons que /sys n'existe que depuis les noyaux 2.6.

Attribution et résolution des noms

Affubler de noms les numéros IP vise à en faciliter la mémorisation par l'homme. En réalité, une adresse IP identifie une interface réseau — un périphérique associé à une carte réseau ou assimilé ; chaque machine peut donc en compter plusieurs, et par conséquent recevoir plusieurs noms dans le système responsable de leur attribution : le DNS.

Chaque machine est cependant identifiée par un nom principal (ou « canonique »), stocké dans le fichier /etc/hostname et communiqué au noyau Linux par les scripts d'initialisation à travers la commande **hostname**. On peut en prendre connaissance dans le fichier virtuel /proc/sys/kernel/hostname.

Étonnamment, le nom de domaine n'est pas géré de la même manière, mais provient du nom complet de la machine, obtenu par une résolution de noms. On

pourra le modifier dans le fichier /etc/hosts ; il suffit d'y placer un nom complet de machine au début de la liste des noms associés à l'adresse de la machine comme dans l'exemple ci-dessous :

```
127.0.0.1     localhost
192.168.0.1   arrakis.falcot.com arrakis
```

Résolution de noms

Le mécanisme de résolution de noms de Linux, modulaire, peut s'appuyer sur différentes sources d'informations déclarées dans le fichier /etc/nsswitch.conf. L'entrée qui concerne la résolution des noms d'hôtes est `hosts`. Par défaut, elle contient `files dns`, ce qui signifie que le système consulte en priorité le fichier /etc/hosts puis interroge les serveurs DNS. Des serveurs NIS/NIS+ ou LDAP forment d'autres sources possibles.

Configuration des serveur DNS

Le DNS (*Domain Name Service*, ou service de noms) est un service distribué et hiérarchique associant des noms à des adresses IP et vice versa. Concrètement, il permet de savoir que `www.eyrolles.com` est en réalité l'adresse IP `213.244.11.247`.

Pour accéder aux informations du DNS, il faut disposer d'un serveur DNS relayant les requêtes. Falcot SA a les siens, mais un particulier fait normalement appel aux serveurs DNS de son fournisseur d'accès à l'Internet.

Les serveurs DNS à employer sont donnés dans le fichier /etc/resolv.conf à raison d'un par ligne, le terme `nameserver` y précédant l'adresse IP, comme dans l'exemple suivant.

```
nameserver 212.27.32.176
nameserver 212.27.32.177
```

Fichier `/etc/hosts`

En l'absence d'un serveur de noms sur le réseau local, il est tout de même possible d'établir une petite table de correspondance entre adresses IP et noms de machines dans le fichier /etc/hosts, et habituellement réservée aux postes du réseau local. La syntaxe de ce fichier est très simple : chaque ligne significative précise une adresse IP suivie de la liste de tous les noms qui y sont associés (le premier étant « complètement qualifié », c'est-à-dire incluant le nom de domaine).

Ce fichier est disponible même en cas de panne réseau ou quand les serveurs DNS sont injoignables, mais ne sera vraiment utile que dupliqué sur toutes les machines du réseau. Au moindre changement dans les correspondances, il faudra

donc le mettre à jour partout. C'est pourquoi /etc/hosts ne renferme généralement que les entrées les plus importantes (et notamment celle de sa propre machine).

Pour un petit réseau non connecté à l'Internet, ce fichier suffira, mais à partir de cinq machines il est recommandé d'installer un serveur DNS en bonne et due forme.

ASTUCE **Court-circuiter le DNS**

Étant donné que le fichier /etc/hosts est consulté avant d'interroger le DNS, il est possible d'y mettre des informations différentes de celles habituellement renvoyées par celui-ci, et donc de le court-circuiter.
Cela permet, en cas de changements DNS pas encore propagés, de tester l'accès à un site web avec le nom prévu même si celui-ci n'est pas encore associé à la bonne adresse IP.
Autre emploi original, il est possible de rediriger le trafic destiné à un hôte donné vers la machine locale afin qu'aucune communication avec cet hôte ne soit possible. Les noms de serveurs dédiés à l'envoi de bannières publicitaires pourraient faire l'objet d'une telle mesure, ce qui rendrait la navigation plus fluide et moins distrayante puisque leurs annonces ne pourraient plus être chargées.

Base de données des utilisateurs et des groupes

La liste des utilisateurs est habituellement stockée dans le fichier /etc/passwd, alors que le fichier /etc/shadow stocke les mots de passe chiffrés. Tous deux sont de simples fichiers texte, au format relativement simple, consultables et modifiables avec un éditeur de texte. Chaque utilisateur y est décrit sur une ligne par plusieurs champs séparés par des deux-points (« : »).

B.A.-BA **Groupe Unix**

Un groupe Unix est une entité regroupant plusieurs utilisateurs afin qu'ils puissent facilement se partager des fichiers à l'aide du système de droits intégré (en jouissant justement des mêmes droits).

ATTENTION **Édition des fichiers système**

Les fichiers système mentionnés dans ce chapitre sont au format texte simple et sont donc éditables avec un éditeur de texte. Étant donnée leur importance, il est toutefois recommandé de prendre des précautions supplémentaires garantissant qu'un fichier ne soit pas modifié par plusieurs personnes à la fois (ce qui pourrait causer une corruption).
Pour cela, il suffit d'employer la commande **`vipw`** pour éditer /etc/passwd, ou **`vigr`** pour /etc/group. Ces dernières posent un verrou sur le fichier concerné avant d'exécuter un éditeur de texte. L'option **`-s`** de ces commandes permet d'éditer le fichier *shadow* correspondant.

Liste des utilisateurs : `/etc/passwd`

Voici la liste des champs du fichier /etc/passwd :

- identifiant (ou *login*), par exemple `rhertzog` ;
- mot de passe : il s'agit d'un mot de passe chiffré par la fonction à sens unique **`crypt`** ou **`md5`**. La valeur spéciale « x » indique que le mot de passe chiffré est stocké dans /etc/shadow ;

- `uid` : numéro unique identifiant l'utilisateur ;
- `gid` : numéro unique du groupe principal de l'utilisateur (Debian crée par défaut un groupe spécifique à chacun) ;
- `GECOS` : champ de renseignements qui contient habituellement le nom complet de l'utilisateur ;
- répertoire de connexion, attribué à l'utilisateur pour qu'il y stocke ses fichiers personnels (la variable d'environnement `$HOME` y pointe habituellement) ;
- programme à exécuter après la connexion. Il s'agit généralement d'un interpréteur de commandes (shell), donnant libre cours à l'utilisateur. Si l'on précise **`/bin/false`** (programme qui ne fait rien et rend la main immédiatement), l'utilisateur ne pourra pas se connecter.

B.A.-BA **Crypt, une fonction à sens unique**

`crypt` est une fonction à sens unique qui transforme une chaîne A en une autre chaîne B de telle sorte qu'à partir de B, il ne soit pas possible dans le cas général de retrouver A. La seule manière d'identifier A est de tester toutes les valeurs possibles, en vérifiant pour chacune si sa transformation par la fonction produit B ou non. Elle utilise jusqu'à 8 caractères en entrée (chaîne A) et génère une chaîne de 13 caractères ASCII imprimables (chaîne B).

Le fichier des mots de passe chiffrés et cachés : `/etc/shadow`

Le fichier /etc/shadow contient les champs suivants :

- identifiant (ou *login*) ;
- mot de passe chiffré ;
- plusieurs champs de gestion de l'expiration du mot de passe.

SÉCURITÉ **Sûreté du fichier `/etc/shadow`**

/etc/shadow, contrairement à son alter ego /etc/passwd, est inaccessible en lecture aux utilisateurs. Tout mot de passe chiffré stocké dans /etc/passwd est lisible par tous ; un indélicat peut alors entreprendre de le « casser » par une méthode de force brute, consistant simplement à chiffrer successivement tous les mots de passe simples pour tenter de le découvrir. Cette attaque, dite « du dictionnaire », qui dévoile les mots de passe mal choisis, n'est plus possible avec le fichier /etc/shadow.

DOCUMENTATION **Formats des fichiers `/etc/passwd`, `/etc/shadow` et `/etc/group`**

Ces formats sont documentés dans les pages de manuel suivantes : `passwd(5)`, `shadow(5)`, et `group(5)`.

Modifier un compte ou mot de passe existant

Quelques commandes permettent de modifier la plupart des informations stockées dans ces bases de données. Chaque utilisateur peut ainsi changer de mot de passe, sans doute le champ le plus variable, grâce à la commande **`passwd`**. **`chfn`** (*CHange Full Name*), réservée au super-utilisateur root, intervient sur le champ `GECOS`. **`chsh`** (*CHange SHell*) permet de changer de « shell de login », ou interpréteur de commandes de connexion, mais le choix des utilisateurs sera limité à la liste donnée dans /etc/shells — alors que l'administrateur pourra saisir le nom de programme de son choix.

Enfin, la commande **`chage`** (*CHange AGE*) donnera à l'administrateur la possibilité de modifier les conditions d'expiration du mot de passe (l'option **`-l`** ***`utilisateur`*** rappelant la configuration actuelle). On pourra d'ailleurs forcer l'expiration d'un mot de passe grâce à la commande **`passwd -e`** ***`utilisateur`***, qui obligera l'utilisateur à changer son mot de passe à la prochaine connexion.

POUR ALLER PLUS LOIN **Base de données système et NSS**

Au lieu d'employer les fichiers habituels pour gérer les listes des utilisateurs et des groupes, on peut recourir à d'autres types de base de données — comme LDAP ou **db** — en employant un module NSS (*Name Service Switch*, ou multiplexeur de service de noms) adéquat. Les listes des modules employés se trouvent dans le fichier /etc/nsswitch.conf sous les entrées `passwd`, `shadow` et `group`. Voir page 213 pour un exemple concret d'emploi du module NSS pour LDAP.

Bloquer un compte

On peut se trouver dans l'obligation de « bloquer le compte » d'un utilisateur, par mesure disciplinaire ou dans le cadre d'une enquête. Il s'agit en fait de l'empêcher de se connecter à nouveau, sans pour autant détruire son compte et ses fichiers. Cela s'effectue simplement par la commande **`passwd -l`*`utilisateur`*** (*lock*, ou bloquer). La remise en service s'effectue de même, avec l'option **`-u`** (*unlock*, ou débloquer).

Liste des groupes : `/etc/group`

La liste des groupes est stockée dans le fichier /etc/group, simple base de données textuelle au format comparable à celui de /etc/passwd, qui utilise les champs suivants :

- identifiant (le nom du groupe) ;
- mot de passe (facultatif) : il ne sert qu'à intégrer un groupe dont on n'est pas habituellement membre (avec la commande **`newgrp`** ou **`sg`** — voir encadré) ;
- `gid` : numéro unique identifiant le groupe ;
- liste des membres : liste des identifiants d'utilisateurs membres du groupe, séparée par des virgules.

B.A.-BA **Travailler avec plusieurs groupes**

Chaque utilisateur peut donc faire partie de plusieurs groupes ; l'un d'entre eux est son « groupe principal » ; le groupe principal par défaut est mis en place lors de la connexion. Par défaut, chaque fichier qu'il crée lui appartient, ainsi qu'au groupe principal. Cela n'est pas toujours souhaitable : c'est par exemple le cas lors d'un travail dans un répertoire partagé grâce à un groupe différent de son groupe principal. Dans ce cas, l'utilisateur a intérêt à changer temporairement de groupe principal grâce aux commandes **`newgrp`** — qui démarre un nouveau shell — ou **`sg`** — qui se contente d'exécuter une commande. Ces commandes permettent aussi de rejoindre un groupe dont on ne fait pas partie si le groupe est protégé par un mot de passe connu.
La commande **`id`** permet de vérifier à tout instant son identifiant personnel (variable `uid`), le groupe principal actuel (variable `gid`), et la liste des groupes dont on fait partie (variable `groupes`).

ASTUCE **`getent`**

La commande **`getent`** (*get entries*) consulte les bases de données du système de manière classique, en employant les appels système adéquats, donc les modules NSS configurés dans le fichier /etc/nsswitch.conf. Elle prend un ou deux arguments : le nom de la base de données à consulter et une éventuelle clé de recherche. Ainsi, la commande **`getent passwd rhertzog`** renvoie les informations de la base de données des utilisateurs concernant l'utilisateur `rhertzog`.

Les commandes **`groupadd`** et **`groupdel`** permettent respectivement de créer et de supprimer un groupe. La commande **`groupmod`** modifie les informations d'un groupe (son `gid` ou son identifiant). La commande **`passwd -g`** ***`groupe`*** modifiera le mot de passe d'un groupe, tandis que la commande **`passwd -r -g`** ***`groupe`*** le supprimera.

Création de comptes

L'une des premières actions de l'administrateur est de créer les comptes de ses utilisateurs, ce qui s'effectue très simplement avec la commande **`adduser`**. Celle-ci prend simplement en argument l'identifiant utilisateur à créer.

adduser pose quelques questions avant de créer le compte à proprement parler, mais son déroulement offre peu de surprises. Le fichier de configuration /etc/adduser.conf offre toutefois quelques paramétrages intéressants. On pourra ainsi prévoir automatiquement un quota à chaque nouvel utilisateur en dupliquant celui d'un utilisateur « modèle ». On pourra aussi modifier l'emplacement du compte utilisateur, ce qui ne présente que rarement de l'utilité — c'est le cas si les utilisateurs sont si nombreux qu'il est souhaitable de répartir leurs comptes sur plusieurs disques. On pourra encore choisir un autre interpréteur de commandes par défaut.

B.A.-BA **Quota**

Le terme « quota » désigne une limitation des ressources de la machine qu'un utilisateur peut employer. Il s'agit souvent d'espace disque.

La création du compte fabrique le répertoire personnel et y recopie le contenu du répertoire modèle /etc/skel, afin de fournir quelques fichiers standards.

Dans certains cas, il sera utile d'ajouter un utilisateur dans un groupe, en particulier pour lui conférer des droits supplémentaires. Par exemple, un utilisateur intégré au groupe *audio* pourra accéder aux périphériques son (voir encadré « Droits d'accès à un périphérique » page 119). Pour ce faire, on procède avec la commande **adduser *utilisateur groupe***.

B.A.-BA **Droits d'accès à un périphérique**

Chaque périphérique matériel est représenté sous Unix par un fichier dit « spécial », habituellement stocké dans l'arborescence /dev/ (*DEVices*). On distingue deux types de fichiers spéciaux selon la nature du périphérique : des fichiers en « mode caractère » et des fichiers en « mode bloc », chaque mode ne permettant qu'un nombre limité d'opérations. Alors que le mode caractère limite les interactions aux opérations de lecture et d'écriture, le mode bloc permet aussi de se déplacer dans le flux de données disponibles. Enfin, chaque fichier spécial est associé à deux nombres (dits « majeur » et « mineur ») qui identifient de manière unique le périphérique auprès du noyau. Un tel fichier, créé par la commande **mknod**, n'a donc qu'un nom symbolique plus pratique pour l'utilisateur humain.
Les droits d'accès à un fichier spécial décrivent directement les droits d'accès au périphérique. Ainsi, un fichier comme /dev/mixer — représentant le mixer audio — n'est accessible en lecture/écriture qu'à `root` et aux membres du groupe `audio`. Seuls ces utilisateurs pourront donc exploiter le mixer audio.

Environnement des interpréteurs de commandes

Les interpréteurs de commandes (ou shells), premier contact de l'utilisateur avec l'ordinateur, doivent être assez conviviaux. La plupart utilisent des scripts d'initialisation permettant de configurer leur comportement (complétion automatique, texte d'invite, etc.).

B.A.-BA **Shell de connexion et shell (non) interactif**

Pour simplifier, un shell de connexion est invoqué lors d'une connexion — sur la console, via **telnet** ou **ssh**, ou à travers la commande explicite **bash --login**. Un shell interactif est celui qui prend place dans un terminal de type **xterm** ; un shell non interactif est celui qui permet d'exécuter un script.

bash, l'interpréteur de commandes standard, emploie les scripts d'initialisation /etc/bash.bashrc (pour les shells « interactifs ») et /etc/profile (pour les shells « de connexion »).

DÉCOUVERTE **Autres shells, autres scripts**

Chaque interpréteur de commande a une syntaxe spécifique et ses propres fichiers de configuration. Ainsi, **zsh** emploie /etc/zshrc et /etc/zshenv ; **csh** utilise /etc/csh.cshrc, /etc/csh.login et /etc/csh.logout… les pages de manuel de ces programmes documentent les fichiers employés.

B.A.-BA **Complétion automatique**

De nombreux interpréteurs de commandes en sont capables : il s'agit pour eux de compléter automatiquement un nom de commande ou d'argument (fichier ou répertoire) saisi partiellement. Pour cela, l'utilisateur enfoncera la touche de tabulation ; il peut ainsi travailler plus vite et avec moins de risques d'erreur.
Cette fonctionnalité est très riche : il est possible de personnaliser son comportement en fonction de chaque commande. Ainsi le premier argument suivant **apt-get** sera proposé en fonction de la syntaxe de cette commande, même s'il ne correspond à aucun fichier (en l'occurrence, les choix possibles sont `install`, `remove`, `upgrade`, etc.).

Pour **bash**, il est intéressant d'activer la « complétion automatique » dans le fichier /etc/bash.bashrc (il suffit pour cela d'y décommenter quelques lignes).

B.A.-BA **Le tilde, raccourci vers** `HOME`

Le tilde est fréquemment employé pour désigner le répertoire pointé par la variable d'environnement `HOME` (à savoir le répertoire de connexion de l'utilisateur, par exemple /home/rhertzog). Les interpréteurs de commandes font la substitution automatiquement : ~/hello.txt devient /home/rhertzog/hello.txt.

En plus de ces scripts communs à tous, chaque utilisateur peut se créer des fichiers ~/.bashrc et ~/.bash_profile pour personnaliser son shell. Les ajouts les plus courants sont la mise en place d'alias, mots automatiquement remplacés avant exécution de la commande, ce qui accélère la saisie. On pourra ainsi créer un alias `la` pour la commande **`ls -la | less`**, et se contenter de saisir **`la`** pour inspecter en détail le contenu d'un répertoire.

B.A.-BA **Variables d'environnement**

Les variables d'environnement permettent de stocker des paramètres globaux à destination du shell ou des divers programmes appelés. Elles sont contextuelles (chaque processus a son propre ensemble de variables d'environnement) mais héritables. Cette dernière caractéristique offre la possibilité à un shell de connexion de déclarer des variables qui se retrouveront dans tous les programmes exécutés par son intermédiaire.

Un élément important de configuration des shells est la mise en place de variables d'environnement par défaut. Si l'on néglige les variables spécifiques à un interpréteur de commande, il est préférable de mettre celles-ci en place dans le fichier /etc/environment, utilisé par les différents programmes susceptibles d'initier une session shell. Parmi les variables susceptibles d'y être définies, citons `ORGANIZATION` qui contient habituellement le nom de l'entreprise ou organisation et `HTTP_PROXY` qui indique l'existence et l'emplacement d'un proxy (ou mandataire) HTTP.

ASTUCE **Tous les shells configurés à l'identique**

Les utilisateurs souhaitent souvent configurer de la même manière shells de connexion et interactifs. Pour cela, ils choisissent d'interpréter (ou « sourcer ») le contenu de ~/.bashrc depuis le fichier ~/.bash_profile. Il est possible de faire de même avec les fichiers communs à tous les utilisateurs (en appelant /etc/bash.bashrc depuis /etc/profile).

Configuration de l'impression

Cette étape a longtemps causé bien des soucis, désormais en passe d'être résolus grâce à l'apparition de *cupsys*, serveur d'impression libre connaissant le protocole IPP (*Internet Printing Protocol*, ou protocole d'impression sur Internet).

Ce logiciel est réparti en plusieurs paquets Debian : *cupsys* est le serveur central ; *cupsys-bsd* est une couche de compatibilité offrant les commandes du système d'impression BSD traditionnel (démon **lpd**, commandes **lpr**, **lpq**, etc.) ; *cupsys-client* renferme un ensemble de programmes pour interagir avec le serveur (bloquer ou débloquer une imprimante, consulter ou annuler les impressions en cours, etc.). Enfin, *cupsys-driver-gimpprint* contient une collection supplémentaire de pilotes d'imprimantes pour **cupsys**.

Après installation de ces différents paquets, **cupsys** s'administre très facilement grâce à son interface web accessible à l'adresse locale `http://localhost:631`. On pourra y ajouter des imprimantes (y compris réseau), les supprimer et les administrer. On peut encore administrer *cupsys* avec l'interface graphique **gnome-cups-manager** (du paquet Debian éponyme).

COMMUNAUTÉ **CUPS**

CUPS (*Common Unix Printing System*, ou système d'impression commun sous Unix), est une marque déposée de la société *Easy Software Products*, à l'origine de **cupsys**.

▸ http://www.easysw.com

ATTENTION **Obsolescence de /etc/printcap**

cupsys n'utilise plus le fichier /etc/printcap, désormais obsolète. Les programmes qui se fieraient à son contenu pour connaître la liste des imprimantes disponibles feraient donc erreur. Pour éviter ce désagrément, on supprimera ce fichier pour en faire un lien symbolique vers /var/run/cups/printcap, fichier maintenu par *cupsys* pour assurer la compatibilité.

Configuration du chargeur d'amorçage

Il est probablement déjà fonctionnel, mais il est toujours bon de savoir configurer et installer un chargeur d'amorçage au cas où celui-ci disparaîtrait du *Master Boot Record* (enregistrement d'amorçage maître). Cela peut se produire suite à l'installation d'un autre système d'exploitation, tel que Windows. Ces connaissances vous permettront également d'en modifier la configuration si l'actuelle ne vous convient pas.

B.A.-BA **Master Boot Record**

Le *Master Boot Record* (MBR, ou enregistrement d'amorçage maître) est la zone des 512 premiers octets du premier disque dur, chargée par le BIOS pour donner la main à un programme capable de démarrer le système d'exploitation voulu. En général, un chargeur d'amorçage s'installe donc sur le MBR en écrasant son contenu antérieur.

Identifier ses disques

La configuration du chargeur d'amorçage doit identifier les différents disques et leurs partitions. Linux emploie pour cela un système de fichiers spéciaux (dits en mode « bloc »), stockés dans le répertoire /dev/. Ainsi, /dev/hda est le disque maître du premier contrôleur IDE, et /dev/hdb son disque esclave. /dev/hdc et /dev/hdd sont respectivement les disques maître et esclave du deuxième contrôleur IDE, et ainsi de suite pour les autres. /dev/sda correspond au premier disque dur SCSI, /dev/sdb au deuxième, etc.

Chaque partition est représentée par un numéro d'ordre au sein du disque où elle réside : /dev/sda1 est donc la première partition du premier disque SCSI et /dev/hdb3 la troisième partition du disque dur esclave sur le premier contrôleur IDE.

L'architecture PC (ou « i386 ») est limitée à quatre partitions « primaires » par disque. Pour outrepasser cette limitation, l'une d'entre elles sera créée comme

CULTURE **udev et /dev**

Le répertoire /dev abrite traditionnellement des fichiers dits « spéciaux », destinés à représenter les périphériques du système (voir encadré « Droits d'accès à un périphérique » page 119). Cette structure statique ne permettait pas d'établir dynamiquement les numéros « majeurs » et « mineurs » de ces fichiers, ce qui contraignait les développeurs du noyau à limiter leur nombre puisque l'attribution a priori des identifiants interdisait d'en ajouter après établissement des conventions. De plus, le noyau nommait arbitrairement les périphériques, ce qui ne convenait guère à ceux que l'on connecte à chaud car le nom du fichier spécial correspondant variait.

En employant **udev** (*Userspace*), un système de fichiers stocké en RAM et géré par le programme **udevd** dissimule le contenu de /dev. Ce programme collabore avec **hotplug** (voir page 157) pour être informé de l'apparition (à chaud) des périphériques puis crée dynamiquement les fichiers spéciaux correspondants dans /dev. Le contenu de /dev est donc perdu à chaque redémarrage mais **udev** le recréera systématiquement.

Ce mécanisme permet de choisir dynamiquement le nom du fichier : on pourra ainsi garder le même nom pour un périphérique donné quel que soit le connecteur employé. Par ailleurs, /dev ne contient plus que les fichiers utiles sur le moment. Auparavant, certains modules noyau se chargeaient automatiquement lorsqu'on tentait d'accéder au périphérique correspondant ; désormais le fichier spécial du périphérique n'existe plus avant d'avoir chargé le module… ce qui n'est pas très grave puisque la plupart des modules sont chargés au démarrage grâce à la détection automatique du matériel. Mais pour des périphériques non détectables (comme le bon vieux lecteur de disquettes ou la souris PS/2), cela ne fonctionne pas. Pensez donc à ajouter les modules `floppy`, `psmouse` et `mousedev` dans /etc/modules afin de forcer leur chargement au démarrage.

une partition « étendue », et pourra alors contenir des partitions « secondaires ». Ces dernières portent toujours un numéro supérieur ou égal à 5. La première partition secondaire pourra donc être /dev/hda5, suivie de /dev/hda6, etc.

Il n'est pas toujours facile de mémoriser quel disque est branché sur le second contrôleur IDE ou en troisième position dans la chaîne SCSI. Pour retrouver les bonnes correspondances, on pourra consulter les messages affichés par le noyau lors de son démarrage : celui-ci décrit en effet les périphériques détectés. La commande **dmesg** rappelle cette séquence :

```
hda: SAMSUNG SP0802N, ATA DISK drive
hdc: SAMSUNG DVD-ROM SD-616E, ATAPI CD/DVD-ROM drive
hdd: Memup 5232IA, ATAPI CD/DVD-ROM drive
[...]
hdc: ATAPI 48X DVD-ROM CD-MR drive, 512kB Cache, UDMA(33)
hdd: ATAPI 52X CD-ROM CD-R/RW CD-MRW drive, 2048kB Cache, UDMA(33)
```

Les exemples de fichiers de configuration donnés dans les sections suivantes reposent tous sur le même cas : un seul disque IDE maître, dont la première partition est dédiée à un ancien Windows et la seconde contient Debian GNU/Linux.

Configuration de LILO

LILO (*LInux LOader*, ou chargeur de Linux) est le plus ancien chargeur d'amorçage, solide mais rustique. Il code en dur dans le MBR l'adresse du noyau à démarrer, c'est pourquoi chaque mise à jour de celui-ci (ou du fichier de configuration de LILO) doit être suivie de la commande **lilo**. L'oublier produira un système incapable de démarrer si l'ancien noyau a été supprimé ou remplacé.

LILO a pour fichier de configuration /etc/lilo.conf, dont un exemple simple est donné ci-dessous.

EXEMPLE **Fichier de configuration de LILO**

```
# Le disque sur lequel LILO doit s'installer.
# En indiquant le disque et non pas une partition,
# on ordonne à LILO de s'installer sur le MBR.
boot=/dev/hda
# la partition qui contient Debian
root=/dev/hda2
# l'élément à charger par défaut
default=Linux

# Noyau le plus récent
image=/vmlinuz
  label=Linux
  initrd=/initrd.img
  read-only

# Ancien noyau (si le noyau nouvellement installé ne démarre pas)
image=/vmlinuz.old
  label=LinuxOLD
  initrd=/initrd.img.old
  read-only
  optional

# Seulement pour un double amorçage Linux/Windows
other=/dev/hda1
  label=Windows
```

Configuration de GRUB

GRUB (*GRand Unified Bootloader, ou grand chargeur d'amorçage unifié)* est plus récent. Il n'est pas nécessaire de l'invoquer après chaque mise à jour du noyau puisqu'il sait lire les systèmes de fichiers et retrouver tout seul la position du noyau sur le disque. Pour l'installer dans le MBR du premier disque IDE, on saisira simplement **`grub-install /dev/hda`**.

GRUB a pour fichier de configuration /boot/grub/menu.lst, dont un exemple simple est donné ci-dessous.

EXEMPLE **Fichier de configuration de GRUB**

```
# Démarre automatiquement après 30 secondes
timeout 30
# Démarre la première entrée par défaut
default 0
# Si celle-ci échoue, alors essaie la seconde
fallback 1

# Dernier noyau installé
title GNU/Linux
root (hd0,1)
kernel /vmlinuz root=/dev/hda2
initrd /initrd.img

# Ancien noyau (si le récent ne démarre pas)
title GNU/Linux OLD
root (hd0,1)
kernel /vmlinuz.old root=/dev/hda2
```

ATTENTION **Noms des disques pour GRUB**

GRUB fait appel au BIOS pour identifier les disques durs. `(hd0)` correspond au premier disque ainsi détecté, `(hd1)` au deuxième, etc. Dans la majorité des cas, cet ordre correspond exactement à l'ordre habituel des disques sous Linux, mais des problèmes peuvent survenir lorsque l'on associe disques SCSI et disques IDE. GRUB stocke les correspondances qu'il détecte dans le fichier /boot/grub/device.map. Si vous y trouvez des erreurs (parce que vous savez que votre BIOS détecte les disques dans un autre ordre), corrigez-les manuellement et exécutez à nouveau **`grub-install`**.
La première partition du premier disque est notée `(hd0,0)`, la seconde `(hd0,1)`, etc.

```
initrd /initrd.img.old

# Seulement pour un double amorçage Linux/Windows
title Microsoft Windows
rootnoverify (hd0,0)
makeactive
chainloader +1
```

Cas des Macintosh : configuration de Yaboot

Yaboot est le chargeur de démarrage employé par les Macintosh récents. Ils n'amorcent pas comme les PC, mais recourent à une partition de démarrage dite de *bootstrap*, à partir de laquelle le BIOS (ou *OpenFirmware*) exécute le chargeur, et sur laquelle le programme **ybin** installe **yaboot** et son fichier de configuration. On n'exécutera à nouveau cette commande qu'en cas de modification du fichier /etc/yaboot.conf (il est en effet dupliqué sur la partition de *bootstrap*, et **yaboot** sait retrouver la position des noyaux sur les disques).

Avant d'exécuter **ybin** il faut disposer d'un fichier /etc/yaboot.conf valide. L'exemple ci-dessous pourrait constituer un fichier minimal.

EXEMPLE **Fichier de configuration de Yaboot**

```
# La partition de bootstrap
boot=/dev/hda2
# Le disque
device=hd:
# La partition Linux
partition=3
root=/dev/hda3
# Démarre après 3 sec. d'inactivité
timeout=30

install=/usr/lib/yaboot/yaboot
magicboot=/usr/lib/yaboot/ofboot
enablecdboot

# Dernier noyau installé
image=/vmlinux
        label=linux
        initrd=/initrd.img
        read-only

# Ancien noyau
image=/vmlinux.old
        label=old
        initrd=/initrd.img.old
        read-only

# Uniquement pour un double amorçage Linux/Mac OS X
macosx=/dev/hda5

# bsd=/dev/hdaX et macos=/dev/hdaX
# sont également possibles
```

Attention : cet exemple fait appel au mécanisme **initrd**, réservé aux noyaux de la série 2.6.x en ce qui concerne l'architecture *powerpc*. Si vous employez des noyaux 2.4.x, supprimez donc les lignes `initrd=...`.

Autres configurations : synchronisation, logs, partages...

Cette section regroupe de nombreux éléments qu'il est bon de connaître pour maîtriser tous les aspects de la configuration du système GNU/Linux. Ils sont cependant traités brièvement et renvoient souvent à la documentation de référence.

Fuseau horaire

Le fuseau horaire, configuré lors de l'installation initiale, peut être modifié par l'intermédiaire de l'outil **`base-config`**. Sa configuration est stockée dans le fichier /etc/timezone ; par ailleurs le lien symbolique /etc/localtime est mis à jour pour pointer vers le bon fichier du répertoire /usr/share/zoneinfo, qui contient notamment les dates des changements d'heure pour les pays appliquant une heure d'été.

Pour changer temporairement de fuseau horaire, il est possible de mettre en place un fuseau horaire ayant la priorité sur les réglages du système avec la variable d'environnement TZ.

```
$ date
sam aoû 28 15:49:09 CEST 2004
$ TZ="Pacific/Honolulu" date
sam aoû 28 03:50:07 HST 2004
```

B.A.-BA **Le lien symbolique**

Un lien symbolique est un pointeur vers un autre fichier. Quand on y accède, c'est le fichier ainsi pointé qui est ouvert. Sa suppression n'entraîne pas la suppression du fichier pointé. De même, il ne dispose pas de droits propres, ce sont ceux de la cible qui comptent. Enfin, il peut pointer sur n'importe quel type de fichier : répertoires, fichiers spéciaux (*sockets*, tubes, périphériques, etc.), autres liens symboliques...
La commande **`ln -s cible nom-lien`** crée un lien symbolique *nom-lien* pointant sur *cible*.

Rotation des fichiers de logs

Les fichiers de logs prenant rapidement du volume, il est nécessaire de les archiver. On emploie en général une archive « tournante » : le fichier de log est régulièrement archivé, et seules ses *X* dernières archives sont conservées. **`logrotate`**, le programme chargé de ces rotations, suit les directives données du fichier /etc/logrotate.conf et de tous ceux du répertoire /etc/logrotate.d. L'administrateur peut modifier ces fichiers s'il souhaite adapter la politique de rotation des logs définie par Debian. La page de manuel `logrotate(1)` décrit toutes les options autorisées dans ces fichiers de configuration. Il peut être intéressant d'augmenter le nombre de fichiers conservés dans la rotation des logs, ou de déplacer les fichiers de logs dans un répertoire spécifique dédié à l'archivage au lieu de les supprimer. On peut encore les envoyer par courrier électronique pour les archiver ailleurs.

Le programme **`logrotate`** est exécuté quotidiennement par l'ordonnanceur **`cron`**.

B.A.-BA **NTP**

NTP (*Network Time Protocol*, ou protocole d'heure en réseau), permet à une machine de se synchroniser sur une autre en prenant en compte de manière relativement précise les délais induits par le transfert de l'information sur le réseau et les autres décalages possibles.
Bien qu'il existe de nombreux serveurs NTP sur l'Internet, les plus connus peuvent être surchargés. C'est pourquoi il est recommandé d'employer le serveur NTP *pool.ntp.org* — c'est en réalité une collection de machines qui ont accepté de jouer le rôle de serveur NTP public.
Si vous administrez un réseau important, il est toutefois recommandé de mettre en place votre propre serveur NTP, qui se synchronisera avec les serveurs publics. Dans ce cas, toutes les autres machines de votre réseau pourront utiliser le serveur NTP interne au lieu d'augmenter la charge sur les serveurs publics.

Synchronisation horaire

La synchronisation horaire, qui peut paraître superflue sur un ordinateur, prend toute son importance dans le cadre d'un réseau. Les utilisateurs n'ayant pas le droit de modifier la date et l'heure, il est important que ces informations soient exactes pour ne pas les gêner. Par ailleurs, le fait d'avoir tous les ordinateurs synchronisés permet de mieux croiser les informations obtenues à partir de logs issus de machines différentes. Ainsi, en cas d'attaque, il est plus simple de reconstituer la séquence chronologique des actions des indélicats sur les différentes machines compromises.

Pour les stations de travail

Les stations de travail étant redémarrées régulièrement, il suffit de les synchroniser par NTP au démarrage. Pour cela, il est possible d'y installer le paquet Debian *ntpdate*. On changera au besoin le serveur NTP employé en modifiant le fichier /etc/default/ntpdate.

Pour les serveurs

Les serveurs ne redémarrent que très rarement, et il est très important que leur heure système soit juste. Pour conserver une heure correcte en permanence, on installera un serveur NTP local, service proposé par le paquet *ntp-simple*. Dans sa configuration par défaut, le serveur se synchronisera sur *pool.ntp.org* et fournira l'heure à qui la lui demande sur le réseau local. On le configurera à travers le fichier /etc/ntp.conf ; l'élément le plus intéressant à changer est le serveur NTP de référence. Si le réseau compte beaucoup de serveurs, il peut être intéressant de n'avoir qu'un seul serveur qui se synchronise sur les serveurs publics, les autres se synchronisant sur lui.

POUR ALLER PLUS LOIN **Module GPS et autres sources de temps**

Si la synchronisation horaire est cruciale dans votre réseau, il est possible d'équiper un serveur d'un module GPS (qui demandera l'heure aux satellites GPS) ou DCF-77 (qui la captera sur l'horloge atomique de Francfort). Il faut alors recourir au paquet *ntp-refclock*, capable de piloter ces modules. Dans ces cas, la configuration du serveur NTP est un peu plus compliquée, et la consultation de sa documentation un préalable absolument nécessaire.

Partage des droits d'administration

Bien souvent, plusieurs administrateurs s'occupent du réseau. Partager le mot de passe de l'utilisateur « root » n'est pas très élégant et ouvre la porte à des abus du fait de l'anonymat de ce compte partagé. La solution à ce problème est le programme **sudo**, qui permet à certains utilisateurs d'exécuter certaines commandes avec des droits particuliers. Dans son emploi le plus courant **sudo**

permet à un utilisateur de confiance d'exécuter n'importe quelle commande en tant que root.

Quand il s'installe, le paquet *sudo* ne donne aucun droit à personne. Pour en déléguer certains, l'administrateur doit faire appel à la commande **visudo**, qui permet de modifier le fichier de configuration /etc/sudoers (dans l'éditeur de texte **vi**, ou tout éditeur mentionné dans la variable d'environnement EDITOR). L'ajout d'une ligne *utilisateur* ALL=(ALL) ALL permettra à l'utilisateur concerné d'exécuter n'importe quelle commande en tant que root.

Des configurations plus sophistiquées permettront de n'autoriser que quelques commandes particulières à certains utilisateurs. Tous les détails de ces différentes possibilités sont donnés dans la page de manuel sudoers(5).

Liste des points de montage

Le fichier /etc/fstab donne la liste de tous les montages possibles (effectués automatiquement au démarrage ou à exécuter manuellement pour les périphériques amovibles). Chaque point de montage y est détaillé sur une ligne par plusieurs champs séparés par des blancs, et qui sont :

- périphérique à monter : il peut s'agir d'une partition locale (disque dur, cédérom) ou d'un système de fichiers distant (tel que NFS) ;
- point de montage : c'est l'endroit de l'arborescence où ce système de fichiers sera rendu accessible ;
- type : ce champ définit le système de fichiers employé sur le périphérique. ext3, vfat, ntfs, reiserfs, xfs en sont quelques exemples.

  La liste complète des systèmes de fichiers reconnus est disponible dans la page de manuel mount(1). La valeur spéciale swap sert aux partitions d'échange ; la valeur spéciale auto demande au programme **mount** de détecter automatiquement le système de fichiers (ce qui est surtout utile pour les lecteurs de disquettes, car chacune peut abriter un système de fichiers différent) ;
- options : elles sont nombreuses, dépendent du système de fichiers, et sont documentées dans la page de manuel de **mount**. Les plus courantes sont
  - rw ou ro feront respectivement monter le système de fichiers en lecture/écriture ou en lecture seule.
  - noauto désactive le montage automatique au démarrage.
  - user autorise tous les utilisateurs à monter ce système de fichiers (opération d'ordinaire réservée à root).
  - defaults correspond à l'ensemble d'options (rw, suid, dev, exec, auto, nouser et async), qu'on pourra désactiver individuellement après defaults.
- sauvegarde : ce champ est presque toujours à 0. Lorsqu'il vaut 1, il indique à l'utilitaire **dump** que la partition contient des données à sauvegarder ;

B.A.-BA Montage et démontage

Le « montage » est l'action d'intégrer le contenu d'un périphérique (souvent un disque) à l'arborescence générale du système. Le système de fichiers racine est toujours monté par le noyau. Lors de l'initialisation de l'ordinateur, d'autres périphériques y sont souvent intégrés à l'aide de la commande **mount**.
Les périphériques amovibles, non montés automatiquement, devront être montés manuellement par l'utilisateur. Il faudra également que celui-ci puisse les démonter (ou retirer de l'arborescence) ; c'est d'ailleurs un préalable nécessaire à l'éjection de certains d'entre eux, comme les cédéroms. Les utilisateurs normaux ne sont normalement pas habilités à employer les commandes **mount** et **umount**. L'administrateur peut toutefois autoriser ces opérations (indépendamment pour chaque point de montage) en positionnant l'option user dans le fichier /etc/fstab.

B.A.-BA NFS, un système de fichiers réseau

NFS — *Network File System* — est un système de fichiers réseau ; sous Linux il permet d'accéder de manière transparente à des fichiers distants en les intégrant dans l'arborescence du système.

- ordre de vérification : ce dernier champ indique si l'intégrité du système de fichiers doit être vérifiée au démarrage et dans quel ordre cette vérification doit avoir lieu. S'il est à `0`, aucune vérification n'est faite. Le système de fichiers racine doit avoir la valeur `1`, les autres systèmes de fichiers permanents du système recevront la valeur `2`.

EXEMPLE **Exemple de fichier `/etc/fstab`**

```
# /etc/fstab: static file system information.
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
proc            /proc           proc    defaults        0       0
/dev/hda3       none            swap    sw              0       0
/dev/hda4       /               ext3    defaults,errors=remount-ro 0
        1
/dev/hdc        /media/cdrom    iso9660 ro,user,noauto  0       0
/dev/fd0        /media/floppy   auto    rw,user,noauto  0       0
arrakis:/partage /partage       nfs     defaults        0       0
```

La dernière entrée de cet exemple correspond à un système de fichiers en réseau (NFS) : le répertoire /partage du serveur *arrakis* est monté sur le répertoire /partage local. Le format du fichier /etc/fstab est documenté dans la page de manuel `fstab(5)`.

POUR ALLER PLUS LOIN **Auto-monteur**

Le paquet *am-utils* fournit l'auto-monteur **`amd`**, capable de monter les périphériques amovibles à la demande lorsqu'un utilisateur tentera d'accéder à leur point de montage habituel. Il les démontera automatiquement quand plus aucun processus n'y accédera. D'autres auto-monteurs existent, comme par exemple **`automount`** du paquet *autofs*.

`locate` et `updatedb`

La commande **`locate`** retrouve l'emplacement d'un fichier dont on connaît une partie du nom. Elle renvoie un résultat quasi instantanément car elle consulte une base de données particulière qui stocke l'emplacement de tous les fichiers du système ; celle-ci est mise à jour quotidiennement par la commande **`updatedb`** (exécutée par l'intermédiaire du script /etc/cron.daily/find).

Tout le monde pouvant utiliser **`locate`**, il faut veiller à ce que l'existence de fichiers volontairement cachés ne puisse être révélée. C'est pourquoi la commande **`updatedb`** est exécutée avec les droits restreints de l'utilisateur *nobody* (classique sur les systèmes Unix pour ce genre d'emploi). Par ailleurs, il est possible à l'administrateur de lui indiquer dans le fichier /etc/updatedb.conf des répertoires à ne pas prendre en compte (simplement en les listant dans la variable PRUNEPATHS).

Le paquet *slocate* va encore plus loin en remplaçant la commande **`locate`** par une version plus sécurisée qui ne renvoie que des noms de fichiers auxquels votre utilisateur a accès.

Compilation d'un noyau

Les noyaux fournis par Debian intègrent le plus grand nombre possible de fonctionnalités ainsi qu'un maximum de pilotes, afin de couvrir le plus grand spectre de configurations matérielles existantes. C'est pourquoi certains utilisateurs préfèrent recompiler le noyau pour n'y inclure que le strict nécessaire — autant pour optimiser la consommation de mémoire et les performances que pour limiter les failles de sécurité (le code compilé portant alors sur une fraction plus faible du code existant).

La recompilation du noyau est aussi nécessaire si l'on souhaite employer certaines fonctionnalités non intégrées dans sa version standard mais disponibles sous forme de correctifs, ou patches.

Introduction et prérequis

Debian gère le noyau sous forme de paquet. La manière traditionnelle de le compiler et de l'installer n'impliquant pas cela, des outils spécifiques ont été développés. Ils permettent de générer facilement un paquet Debian à partir des sources du noyau Linux ainsi que d'éventuels patches. Les noyaux restant sous le contrôle du système de paquetage, ils peuvent être rapidement supprimés ou déployés sur plusieurs machines. De plus, les scripts associés à ces paquets permettent également une meilleure interaction avec le chargeur de démarrage.

Pour compiler un noyau Linux à la « manière Debian », il faudra employer les outils présents dans le paquet *kernel-package*. Par ailleurs, la configuration du noyau nécessitera le paquet *libncurses5-dev*. Enfin, le paquet *fakeroot* permettra de créer le paquet Debian sans utiliser les droits de l'administrateur.

Récupérer les sources

Comme tout ce qui peut être utile sur un système Debian, les sources du noyau Linux sont disponibles en paquet. Pour les récupérer, il faudra donc installer un paquet *kernel-source-version*. Une requête **`apt-cache search ^kernel-source`** permet d'obtenir la liste des différentes versions du noyau empaquetées par Debian. Les dernières versions en date sont vraisemblablement disponibles dans la distribution *unstable* : on peut les y récupérer sans grands risques (surtout si votre APT est configuré conformément aux instructions de la section « Travailler avec plusieurs distributions » page 89). Il est à noter que les codes sources contenus dans ces paquets ne correspondent pas exactement à ceux que publient Linus Torvalds et les développeurs noyau : Debian applique en effet un certain nombre de patches — comme toutes les distributions. Ces modifications incluent des correctifs (dont certains concernent des failles de sécurité) qui sont en attente d'intégration dans une version ultérieure du noyau ainsi que quelques fonctionnalités spécifiques à Debian (comme *cramfs*, un système de fichiers spécifique pour l'image initrd).

CULTURE **Emplacement des sources du noyau**
Traditionnellement, les sources du noyau Linux ont toujours été placées sous /usr/src/linux, nécessitant donc les droits root pour la compilation. Comme vous le savez, il faut pourtant éviter de travailler inutilement avec les droits de l'administrateur. Il existe bien un groupe `src` qui permet à ses membres de travailler dans ce répertoire, mais on évitera malgré tout de recourir à /usr/src. En conservant les sources du noyau dans un répertoire personnel, vous optez pour la sécurité à tout point de vue : pas de fichiers inconnus du système de paquetage dans /usr/, et pas de risque d'induire en erreur les programmes qui scrutent /usr/src/linux pour obtenir des informations sur le noyau employé.

Dans la suite de cette section, c'est le noyau 2.6.10 qui sera systématiquement retenu. Vous pourrez bien entendu adapter ces exemples à la version particulière du noyau qui vous intéresse.

Ainsi donc, le paquet *kernel-source-2.6.10* a été installé. Il contient le fichier /usr/src/kernel-source-2.6.10.tar.bz2, une archive compressée des sources du noyau. Il faut décompacter ces fichiers dans un nouveau répertoire (et non pas directement dans /usr/src, car il n'y a pas besoin de droits particuliers pour compiler un noyau Linux) : ~/kernel/ conviendra.

```
$ mkdir ~/kernel; cd ~/kernel
$ tar jxf /usr/src/kernel-source-2.6.10.tar.bz2
```

Configuration du noyau

La prochaine étape consiste à configurer le noyau conformément à ses besoins. Le mode opératoire dépend des objectifs.

Si l'on recompile une version plus récente du noyau (éventuellement dotée d'un patch supplémentaire), le plus probable est qu'on veuille rester aussi près que possible de la configuration standard proposée par Debian. Dans ce cas, et au lieu de tout reconfigurer depuis zéro, il est bon de copier le fichier /boot/config-<*version*> (la version est celle du noyau employé actuellement — **`uname -r`** vous la révélera au besoin) en .config dans le répertoire des sources du noyau :

```
$ cp /boot/config-2.6.8-2-686 ~/kernel/kernel-source-2.6.10/.config
```

Si vous ne souhaitez pas changer la configuration, vous pouvez en rester là et sauter directement à la section suivante. Dans le cas contraire, ou si vous avez décidé de tout reconfigurer depuis zéro, il faudra prendre le temps de configurer votre noyau. Pour cela, il propose différentes interfaces, qu'on invoque depuis le répertoire des sources par la commande **`make`** suivie d'un argument.

`make menuconfig` compile et exécute une interface évoluée en mode texte (c'est ici que le paquet *libncurses5-dev* est requis) qui propose de naviguer dans une structure hiérarchique présentant les options proposées. Une pression sur la touche **`Espace`** change la valeur de l'option sélectionnée et **`Entrée`** valide le bouton sélectionné en bas de l'écran : **`Select`** permet de rentrer dans le sous-menu sélectionné, **`Exit`** remonte d'un cran dans la hiérarchie, et **`Help`** produit des informations plus détaillées sur le rôle de l'option sélectionnée. Les flèches permettent de se positionner dans la liste des options et des boutons. Pour quitter le configurateur, il faut sélectionner **`Exit`** depuis le menu principal. Le programme propose alors de sauvegarder les changements : acceptez si vous êtes satisfait de vos choix.

Les autres interfaces ont un fonctionnement similaire, mais inscrit dans des interfaces graphiques plus modernes : **`make xconfig`** emploie la boîte à outils Qt

(ou Tk pour les noyaux 2.4) et **`make gconfig`** recourt à GTK+. La première a besoin de *libqt3-mt-dev* tandis que la seconde requiert *libglib2.0-dev*, *libglade2-dev* et *libgtk2.0-dev*.

`make-kpkg`, commande présentée au paragraphe suivant, exécutera automatiquement **`make oldconfig`** pour s'assurer de la présence d'une configuration noyau. Cette méthode de configuration se contente de réutiliser les choix mémorisés dans le fichier .config. En l'absence de ce dernier, elle se comporte comme **`make config`**, une interface textuelle posant une à une des centaines de questions... Si le fichier .config existe déjà mais ne mentionne pas toutes les options existantes, alors cette méthode ne posera que les questions associées aux nouvelles options, pour lesquelles le fichier ne précise rien.

ASTUCE **`make-kpkg --config`**

Vous pouvez indiquer à **`make-kpkg`** d'employer une autre méthode de configuration que **`make oldconfig`** en lui indiquant la cible à invoquer (`menuconfig`, `xconfig` ou `gconfig`) avec l'option **`--config`**.

Compilation et génération du paquet

ATTENTION **Nettoyer avant de recommencer**

Si vous avez déjà effectué une première compilation dans le répertoire et souhaitez recommencer depuis des sources vierges, il faut exécuter **`fakeroot make-kpkg clean`**. De plus, cela vous permettra de générer un paquet avec un nouveau nom (paramètre **`--append-to-version`** différent).

À ce stade, Debian emploie la commande **`make-kpkg`** pour compiler le noyau puis générer le paquet Debian correspondant. Tout comme **`make`**, **`make-kpkg`** accepte en paramètre le nom d'une cible à exécuter : **`kernel-image`** génère un paquet du noyau compilé, **`kernel-doc`** un paquet contenant la documentation incluse avec le noyau, **`kernel-headers`** un paquet des fichiers d'en-têtes du noyau (fichiers .h du répertoire include — utiles à la compilation de certains modules externes), et **`kernel-source`** un paquet contenant les sources du noyau.

`make-kpkg` accepte plusieurs paramètres : **`--append-to-version`** ***`suffixe`*** ajoute *suffixe* à la fin du nom du noyau et donc du paquet généré. **`--revision`** ***`revision`*** définit le numéro de version du paquet généré. Debian emploie certains suffixes pour identifier les noyaux standards compilés spécifiquement pour certains processeurs (`-386`, `-686`, `-686-smp`, `-k6`, `-k7`, `-k7-smp`). Il est conseillé de ne pas utiliser ces suffixes afin de distinguer facilement les paquets officiels (forgés par le projet Debian) des autres.

Le programme **`make-kpkg`** tentera de créer le paquet Debian avec les droits de root, mais nous souhaitons éviter cela : c'est pourquoi nous avons recouru au programme **`fakeroot`**.

ASTUCE **Entêtes d'un paquet noyau**

`make-kpkg` utilise les informations contenues dans le fichier /etc/kernel-pkg.conf pour générer les en-têtes du paquet Debian du noyau. Si vous souhaitez diffuser le paquet, il est donc souhaitable de modifier ce fichier afin d'y placer des informations correctes.

```
$ fakeroot make-kpkg --append-to-version -falcot --revision 1 kernel-
  image
[...]
$ ls ../*.deb
../kernel-image-2.6.10-falcot_1_i386.deb
```

Comme vous le constatez, le paquet créé ici s'appelle kernel-image-2.6.10-falcot_1_i386.deb.

Compilation de modules externes

Certains modules sont gérés en dehors noyau Linux officiel. Pour les employer, il faut les compiler de concert avec le noyau correspondant. Debian fournit les sources d'un certain nombre de modules externes : *pcmcia-source* (gestion du PCMCIA plus récent), *alsa-source* (pour certaines cartes sonores), *qc-usb-source* pour le pilote de certaines *webcams* USB (Logitech QuickCam Express), etc. Notez que les deux premiers paquets ne sont utiles que dans le cadre de noyaux 2.4 puisque le noyau 2.6 intègre les plus récentes versions de tous ces modules.

Il est difficile de dresser la liste des modules externes disponibles sous forme de sources dans Debian, mais la commande **`apt-cache search source$`** permet de restreindre le champ de la recherche. De toute façon, cette liste n'apporte rien puisqu'il n'y a pas de raison particulière de compiler des modules externes sauf quand on sait qu'on en a besoin — auquel cas la documentation du périphérique vous renseignera.

Prenons l'exemple du paquet *qc-usb-source* : après installation, une archive .tar.gz des sources des modules se trouve dans /usr/src/. Il faut décompacter ces sources dans notre répertoire :

```
$ cd ~/kernel/
$ tar zxf /usr/src/qc-usb-modules.tar.gz
$ ls modules/
qc-usb-source
```

ATTENTION **Bien conserver les paramètres**
Lors de l'invocation de **`make-kpkg modules-image`**, il est important de reprendre le même paramètre **`--append-to-version`** qu'à l'invocation précédente (probablement **`make-kpkg kernel-image`**), puisque son contenu influe sur le nom du répertoire où les modules sont installés, et ce dernier doit correspondre à la version du noyau.
De plus, il faut toujours appeler **`make-kpkg`** depuis le répertoire des sources du noyau — même si l'objectif est de compiler des modules externes placés dans d'autres répertoires.

Les sources des modules sont à présent disponibles dans le répertoire ~/kernel/modules/qc-usb-source. Pour compiler ces modules et en créer un paquet Debian, il faut appeler **`make-kpkg`** avec le paramètre **`modules-image`** en lui indiquant via la variable d'environnement `MODULE_LOC` où trouver les modules (en l'absence de cette variable, il emploie /usr/src/modules — qui ne nous convient pas). Par défaut, il essaie alors de créer les paquets de tous les modules externes que vous aurez décompactés. L'option **`--added-modules`** permet d'indiquer explicitement les modules externes à compiler. Pour en citer plusieurs, séparez-les par une virgule.

```
$ export MODULE_LOC=~/kernel/modules
$ cd ~/kernel/kernel-source-2.6.10
$ fakeroot make-kpkg --append-to-version -falcot modules-image
[...]
Module /home/rhertzog/kernel/modules/qc-usb-source processed fine
```

```
$ ls ../*.deb
../kernel-image-2.6.10-falcot_1_i386.deb
../qc-usb-modules-2.6.10-falcot_0.6.2-2+1_i386.deb
```

Emploi d'un patch sur le noyau

Certaines fonctionnalités ne sont pas intégrées au noyau standard faute de stabilité ou d'accord des mainteneurs du noyau. Dans ce cas, il arrive qu'elles soient diffusés sous la forme de correctif (ou patch), que chacun est libre d'appliquer sur les sources du noyau.

Debian diffuse certains de ces patches par le biais des paquets *kernel-patch-** (exemple : *kernel-patch-debianlogo* qui remplace le sympathique pingouin Tux du démarrage, mascotte de Linux, par un logo Debian). Ces paquets installent des fichiers dans /usr/src/kernel-patches/.

Pour employer un ou plusieurs des patches installés, il suffit de fournir à **make-kpkg** une option **--added-patches** détaillant les patches à appliquer avant d'exécuter la compilation. Avant de se lancer dans une nouvelle compilation, il est recommandé de nettoyer les sources avec la commande **make-kpkg clean**.

```
$ cd ~/kernel/kernel-source-2.6.10
$ fakeroot make-kpkg clean
$ fakeroot make-kpkg --append-to-version -mppe --revision 1 --added-
  patches mppe kernel-image
$ ls ../*.deb
../kernel-image-2.6.10-falcot_1_i386.deb
../kernel-image-2.6.10-mppe_1_i386.deb
../qc-usb-modules-2.6.10-falcot_0.6.2-2+1_i386.deb
```

Attention, un patch ne fonctionnant pas forcément avec toutes les versions des noyaux, il est possible que **make-kpkg** échoue à l'appliquer sur les sources du noyau. Un message vous en informera alors : dans ce cas, référez-vous à la documentation disponible dans le paquet Debian du patch (dans le répertoire /usr/share/doc/kernel-patch-*). Il est probable que le mainteneur indique pour quelles versions du noyau il a été prévu.

Installation d'un noyau

Caractéristiques d'un paquet Debian du noyau

Un paquet Debian de noyau installe l'image du noyau (vmlinuz-<*version*>), sa configuration (config-<*version*>) et sa table de symboles (System.map-<*version*>) dans /boot/. La table de symboles permet aux développeurs de comprendre le sens d'un message d'erreur du noyau (en son absence, les « *oops* » — messages générés suite à un déréférencement de pointeur invalide — n'indiqueraient que

POUR ALLER PLUS LOIN **Configurations particulières**

Cette section présente le comportement par défaut d'un paquet Debian de noyau Linux, mais tout est paramétrable par le biais du fichier /etc/kernel-img.conf. Consultez la page de manuel associée pour en apprendre plus : `kernel-img.conf(5)`

des adresses n'ayant aucune signification pour eux). Les modules sont installés dans le répertoire /lib/modules/<*version*>/.

Les scripts de configuration du paquet génèrent automatiquement une image *initrd* (*init ram disk*) — cette dernière est un mini-système préparé en mémoire (*ram disk*) par le chargeur de démarrage et démarré par le noyau Linux dans le seul but de charger les modules nécessaires pour accéder au périphérique contenant le système Debian complet (par exemple le pilote pour les disques IDE). Un avertissement est automatiquement affiché, prévenant que l'emploi d'un *initrd* requiert un ajustement de la configuration du chargeur de démarrage : une fois la configuration de ce dernier validée, vous pouvez désactiver cet avertissement en plaçant `warn_initrd = No` dans /etc/kernel-img.conf. Enfin, les scripts de post-installation mettent à jour les liens symboliques /vmlinuz, /vmlinuz.old, /initrd.img et /initrd.img.old pour qu'ils pointent respectivement sur les deux derniers noyaux installés ainsi que leurs images *initrd* associées.

La configuration standard des chargeurs de démarrage s'appuie sur ces liens symboliques pour employer automatiquement le dernier noyau installé, tout en laissant la possibilité de démarrer sur le noyau précédent si le dernier installé ne fonctionne pas. Si **`lilo`** est installé, l'installation d'un nouveau noyau vous proposera de l'exécuter pour mettre à jour le code d'amorçage et lui faire prendre en compte le nouveau noyau. Si **`grub`** est votre chargeur de démarrage, refusez sa proposition d'exécuter **`lilo`** et pensez à désinstaller ce dernier dans la foulée pour ne plus être gêné.

Installation avec dpkg

L'emploi fréquent d'**`apt-get`** a tendance à faire oublier l'existence de **`dpkg`**. Le moyen le plus simple d'installer un noyau compilé soi-même reste pourtant la commande **`dpkg -i paquet.deb`**. **`paquet.deb`** représente évidemment le nom d'un paquet *kernel-image*, comme par exemple kernel-image-2.6.10-falcot_1_i386.deb.

La configuration de base obtenue peut aussi bien devenir un serveur qu'un poste de bureautique et elle est reproductible en masse de façon semi-automatisée. Une machine en disposant n'est toutefois pas encore adaptée à un usage donné, c'est pourquoi l'administrateur doit à présent compléter la préparation. Pour cela il commencera par mettre en place les couches logicielles basses appelées « services Unix ».



9



Services Unix

Ce chapitre parcourt un ensemble de services fondamentaux, souvent communs à beaucoup d'Unix. Tout administrateur se doit de les connaître.

SOMMAIRE

- Démarrage du système
- Connexion à distance
- Gestion des droits
- Interfaces d'administration
- Les événements système de **`syslog`**
- Le super-serveur **`inetd`**
- Planification synchrone : **`cron`** et **`atd`**
- Planification asynchrone : **`anacron`**
- Les quotas
- Supervision
- Sauvegarde
- Branchements « à chaud » : **`hotplug`**
- Gestion de l'énergie
- Cartes pour portables : PCMCIA

MOTS-CLEFS

- Démarrage du système
- Scripts d'initialisation
- SSH, Telnet
- Droits et permissions
- Supervision
- Inetd
- Cron
- Sauvegarde
- Hotplug
- APM, ACPI
- PCMCIA

Démarrage du système

Lorsque l'ordinateur démarre, les nombreux messages défilant sur la console révèlent de nombreuses initialisations et configurations automatiques. Parfois, il est souhaitable de modifier légèrement le déroulement de cette étape, ce qui implique de bien la comprendre. C'est l'objet de cette section.

En tout premier lieu, le BIOS prend le contrôle de l'ordinateur, détecte les disques, charge le *Master Boot Record* (enregistrement d'amorçage maître), et l'exécute. Le chargeur d'amorçage prend alors le relais, trouve le noyau sur le disque, le charge et l'exécute. Enfin, le noyau s'initialise, monte la partition contenant la racine de l'arborescence, et peut enfin démarrer le premier programme : **`init`**.

Celui-ci exécute tout un ensemble de processus en suivant les indications du fichier /etc/inittab. Le premier programme exécuté (correspondant à l'étape *sysinit*) est **`/etc/init.d/rcS`**, script qui exécute tous les programmes du répertoire /etc/rcS.d.

Parmi ceux-ci, on trouve successivement :

- la configuration du clavier de la console ;
- le chargement des pilotes : modules noyaux spécifiés dans le fichier /etc/modules, puis les modules indiqués par le programme **`discover`** en fonction du matériel qu'il détecte ;
- la vérification de l'intégrité des systèmes de fichiers ;
- le montage des partitions locales ;
- la configuration du réseau ;
- le montage des systèmes de fichiers distants (NFS).

B.A.-BA **Le processus, une invocation de programme**

Un processus est la représentation en mémoire d'un programme qui s'exécute. Il regroupe toutes les informations nécessaires au bon déroulement du logiciel (code machine, état des registres du processeur, mémoire allouée, descripteurs de fichiers ouverts, etc.). Un même programme peut faire l'objet de plusieurs processus, y compris sous le même identifiant utilisateur.

B.A.-BA **Modules du noyau et options**

Les modules du noyau disposent eux aussi d'options qu'on peut paramétrer dans un fichier. Les noyaux 2.4 utilisent pour cela /etc/modules.conf, généré par le programme **`update-modules`** à partir des informations du répertoire /etc/modutils/. Les noyaux 2.6 préfèrent /etc/modprobe.conf, incluant /lib/modules/modprobe.conf (généré par **`update-modules`** à partir des informations du répertoire /etc/modprobe.d/).
Ces changements sont liés aux évolutions du programme **`modprobe`** — le programme permettant de charger un module noyau avec ses dépendances (les modules peuvent en effet faire appel à d'autres modules). Pour un noyau 2.6, le programme **`modprobe`** correspondant provient ainsi du paquet *module-init-tools*, et non plus de *modutils*. Ces deux paquets coexistent très bien et les bons programmes sont employés automatiquement en fonction de la version du noyau.

SÉCURITÉ **Gare à la substitution d'`init` par un shell**

Le premier processus démarré est par convention le programme **`init`**. Toutefois, il est possible de passer au noyau une option **`init`** indiquant un autre programme.
Toute personne capable d'accéder à l'ordinateur pourra appuyer sur le bouton Reset et ainsi le redémarrer, puis, via l'invite du chargeur d'amorçage, passer au noyau l'option **`init=/bin/sh`** pour obtenir un accès root sans connaître le mot de passe de l'administrateur.
Pour éviter cela, on peut protéger le chargeur d'amorçage par un mot de passe. Pensez alors à protéger aussi l'accès au BIOS (un mécanisme de protection par mot de passe est presque toujours disponible), sans quoi un indélicat pourra toujours démarrer sur une disquette contenant son propre système Linux, qu'il utilisera pour accéder aux disques durs de l'ordinateur.
Sachez enfin que la plupart des BIOS disposent de passe-partout génériques. Prévus à l'origine pour dépanner les distraits qui oublient les leurs, ces mots de passe sont désormais publics et diffusés sur l'Internet (vérifiez vous-même en cherchant *BIOS generic passwords* sur un moteur de recherche). Toutes ces protections ralentiront donc l'accès non autorisé à la machine, sans pouvoir l'empêcher totalement. C'est pourquoi il est vain de chercher à protéger un ordinateur si l'attaquant peut y accéder physiquement.

Après cette phase, **`init`** reprend la main et démarre les programmes associés au niveau d'exécution (*runlevel*) normal, soit par défaut le niveau 2. Il exécute

`/etc/init.d/rc 2`, script qui démarre tous les services donnés du répertoire /etc/rc2.d/ débutant par la lettre « S ». Le nombre qui suit sert à classer alphanumériquement les services pour les démarrer dans le bon ordre (certains peuvent en effet dépendre d'autres services). D'une manière générale, les services de base (comme **`syslogd`** ou **`portmap`**) sont démarrés en premier, suivis par les services standards, pour conclure généralement par le service qui démarre l'interface graphique (par exemple **`gdm`**).

`init` distingue plusieurs niveaux d'exécution car il peut basculer de l'un à l'autre par la commande **`telinit`** ***`nouveau-niveau`***. Dès son invocation, **`init`** exécute à nouveau **`/etc/init.d/rc`** avec le nouveau niveau d'exécution désiré, script qui démarre à son tour les services manquants et arrête ceux qui ne sont plus souhaités. Pour cela, il se réfère au contenu du répertoire /etc/rc<*X*>.d (où *X* représente le nouveau niveau d'exécution). Les scripts débutant par « S » (comme *Start*) sont des services à démarrer, ceux débutant par « K » (comme *Kill*) sont des services à stopper. Le script évite de redémarrer tout service déjà actif dans le niveau d'exécution précédent.

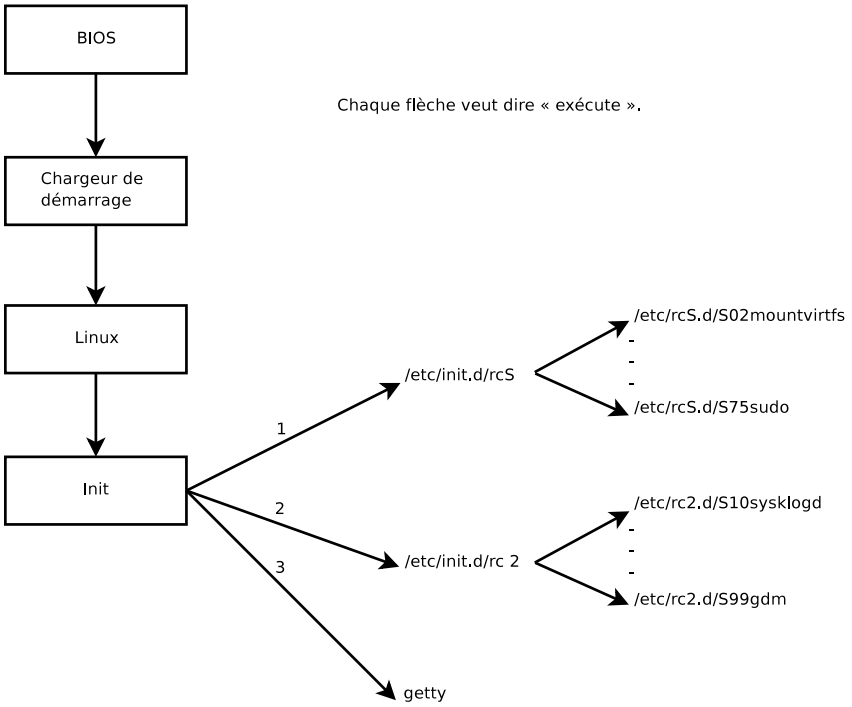


Figure 9–1 Étapes du démarrage d'un ordinateur sous Linux

Tous les scripts contenus dans les différents répertoires /etc/rc<*X*>.d ne sont que des liens symboliques, créés à l'installation du paquet concerné par le programme **`update-rc.d`**, et menant vers les scripts réels, stockés sous /etc/init.d. Pour adapter à sa guise les services à démarrer ou à stopper à chaque niveau d'exécution, l'administrateur exécutera à nouveau le programme **`update-rc.d`**

en lui fournissant les paramètres adéquats. La page de manuel `update-rc.d(1)` en détaille la syntaxe précise.

CHARTE DEBIAN **Redémarrage des services**

Les scripts de configuration des paquets Debian redémarrent parfois certains services pour assurer leur disponibilité ou leur faire prendre en compte certaines nouvelles options. La commande de redémarrage d'un service **`/etc/init.d/`*`service`* `restart`** ne prend pas en compte le niveau d'exécution, suppose (à tort) que le service est actuellement employé, et peut donc le redémarrer à mauvais escient. Debian a donc introduit le programme **`invoke-rc.d`**, auquel les scripts de configuration doivent recourir pour appeler les scripts d'initialisation des services. Il n'exécutera que les commandes nécessaires (un service stoppé ne peut pas être redémarré, ni arrêté à nouveau, etc.). Attention, contrairement à l'usage, le suffixe .d est ici employé sur un nom de programme et non pas sur un répertoire.

Enfin, **`init`** démarre les programmes de contrôle des différentes consoles virtuelles (**`getty`**). Ils affichent une invite, attendent un nom d'utilisateur, puis exécutent **`login`** ***`utilisateur`*** pour démarrer une session.

Connexion à distance

Il est essentiel pour un administrateur de pouvoir se connecter à distance sur un ordinateur. Les serveurs, confinés dans leur propre salle, disposent en effet rarement d'un clavier et d'un écran connectés en permanence — mais sont reliés au réseau.

Connexion à distance : `telnet`

Le protocole *telnet*, doyen de la connexion à distance, est le pire du point de vue de la sécurité. Les mots de passe y circulant en clair — c'est-à-dire sans être chiffrés — tout indélicat situé sur le réseau peut les intercepter. Vous prendrez donc soin de désinstaller ce service obsolète :

```
# apt-get remove telnetd
```

Il en existe cependant une adaptation corrigeant ses défauts rédhibitoires ; elle emploie SSL (*Secure Socket Layer*) pour authentifier le partenaire et chiffrer les communications. Les paquets *telnetd-ssl* et *telnet-ssl* en fournissent respectivement le serveur et le client.

CULTURE **SSH comparé à RSH**

Les outils SSH reprennent en les sécurisant les programmes de la classique famille RSH (*Remote Shell*, ou shell à distance) — **`rsh`**, **`rlogin`**, et **`rcp`**. Ces derniers sont toujours disponibles dans les paquets *rsh-server* et *rsh-client* mais ils sont désormais fortement déconseillés.

Connexion à distance sécurisée : SSH

Le protocole *SSH* (*Secured Shell*, ou shell sécurisé), contrairement à *telnet*, a été conçu dans une optique de sécurité et de fiabilité. Les connexions ainsi mises en place sont sûres : le partenaire est authentifié et tous les échanges sont chiffrés.

SSH offre encore deux services de transfert de fichiers. **scp** est un utilitaire en ligne de commande qui s'emploie comme **rcp** (ou **cp**) ; tout chemin sur une autre machine sera préfixé du nom de celle-ci suivie du caractère deux-points.

```
$ scp fichier machine:/home/rhertzog/
```

sftp est un programme interactif très similaire à **ftp**. Ainsi une même session **sftp** peut transférer plusieurs fichiers, et il est possible d'y manipuler les fichiers distants (supprimer, changer leur nom ou leurs droits, etc.).

Debian emploie OpenSSH, version libre de SSH maintenue par le projet **OpenBSD** (un système d'exploitation libre basé sur un noyau BSD) et *fork* du logiciel SSH originel développé par la société finlandaise *SSH Communications Security Corp*. Celle-ci, qui en avait débuté le développement sous la forme d'un logiciel libre, avait en effet décidé de le poursuivre sous une licence propriétaire. Le projet OpenBSD créa donc OpenSSH pour maintenir une version libre de SSH.

Pour installer OpenSSH, il suffit de saisir **apt-get install ssh**. Pour que la machine concernée accepte les connexions SSH, il faudra y accepter l'exécution de **sshd** (la question est posée au cours de l'installation).

B.A.-BA **Fork**

Le terme *fork* (fourche, ou projet dérivé), dans le cadre d'un logiciel, désigne un nouveau projet, concurrent de l'original dont il s'inspire et qu'il a entièrement copié au début. Ces deux logiciels identiques divergent rapidement sur le plan du développement. C'est souvent un désaccord dans l'équipe qui est à l'origine d'un *fork*.
Cette possibilité provient directement du caractère libre d'un logiciel ; un *fork* est sain lorsqu'il permet la poursuite du développement sous forme de logiciel libre (en cas de changement de licence par exemple). Un *fork* issu d'un désaccord technique est souvent un gâchis de ressources humaines ; on lui préférera la résolution du différend. Il n'est d'ailleurs pas rare d'assister à la fusion des branches d'un *fork* quand elles font ce constat amer.

Utiliser des applications X11 à distance

Le protocole SSH permet de faire suivre (*forward*) les données graphiques (dites « X11 », du nom du système graphique le plus répandu sous Unix) : le serveur leur met alors en place un canal de données spécifique. Concrètement, une application graphique exécutée à distance peut s'afficher sur le serveur XFree86 de l'écran local, et toute la session (manipulation comme affichage) sera sécurisée. Cette fonctionnalité étant par défaut désactivée pour des raisons de sécurité, on l'activera en précisant `X11Forwarding yes` dans le fichier de configuration /etc/ssh/sshd_config. L'utilisateur pourra ensuite en profiter en spécifiant l'option **-X** de **ssh**.

Créer des tunnels chiffrés avec le *port forwarding*

Ses options **-R** et **-L** permettent à **ssh** de créer des « tunnels chiffrés » entre deux machines, déportant de manière sécurisée un port TCP local sur une machine distante ou vice versa.

ssh -L 8000:serveur:25 intermediaire crée une *socket* locale sur le port 8000. Toute connexion établie sur ce port fera débuter par **ssh** une connexion de l'ordinateur `intermediaire` vers le port 25 de `serveur`, à laquelle elle la reliera.

ssh -R 8000:serveur:25 intermediaire crée une *socket* écoutant le port 8000 sur la machine `intermediaire`. Toute connexion établie sur ce port fera débuter par **ssh** une connexion depuis la machine locale vers le port 25 de `serveur`, à laquelle elle la reliera.

Accéder à distance à des bureaux graphiques

VNC (*Virtual Network Computing*, ou informatique en réseau virtuel) permet d'accéder à distance à des bureaux graphiques.

Cet outil sert principalement en assistance technique : l'administrateur peut constater les erreurs de l'utilisateur et lui montrer la bonne manipulation sans devoir se déplacer à ses côtés. Il suffit pour cela à l'utilisateur d'exécuter **x11vnc** (du paquet Debian éponyme), commande pour laquelle on peut même prévoir une icône spécifique. Il suffit alors à l'administrateur d'utiliser un client VNC de son choix (par exemple **xvncviewer**, du paquet Debian éponyme) pour se connecter sur son bureau graphique et examiner ce qui s'y passe, voire intervenir.

Il sert encore aux utilisateurs nomades, ou responsables d'entreprise, ayant des besoins ponctuels de connexion depuis chez eux, qui retrouvent ainsi à distance un bureau similaire à celui qu'ils ont au travail. La configuration d'un tel service est plus compliquée : il faut d'abord installer le paquet *vncserver*, puis suivre les indications du fichier /usr/share/doc/vncserver/README.inetd. On résout ainsi le problème de l'authentification, puisque seuls les utilisateurs disposant de comptes locaux passeront le cap de la connexion via **gdm** (ou les équivalents **kdm**, **xdm**, etc.).

vncserver crée un serveur X destiné exclusivement à une connexion à distance. **x11vnc** exploite quant à lui un serveur X existant (avec un utilisateur local) et l'exporte temporairement, afin que quelqu'un d'autre puisse s'y connecter.

B.A.-BA **Gestionnaire d'écran**

gdm, **kdm** et **xdm** sont des gestionnaires d'écran (*Display Manager*). Ils prennent le contrôle de l'interface graphique peu après son initialisation afin de proposer à l'utilisateur un écran d'identification. Une fois ce dernier authentifié, il démarre les programmes requis pour démarrer une session de travail graphique.

Gestion des droits

Linux est résolument multi-utilisateur ; il est donc nécessaire de prévoir un système de permissions contrôlant les opérations que chacun peut faire sur les fichiers et répertoires, recouvrant toutes les ressources du système (sur un système Unix, tout périphérique est représenté par un fichier ou répertoire). Ce principe est commun à tous les Unix mais un rappel est toujours utile, d'autant qu'il existe quelques usages avancés méconnus et relativement intéressants.

Chaque fichier ou répertoire dispose de permissions spécifiques pour trois catégories d'utilisateurs :

- son propriétaire (symbolisé par `u` comme *user*) ;
- son groupe propriétaire (symbolisé par `g` comme *group*) — représentant tous les utilisateurs membres du groupe ;
- les autres (symbolisés par `o` comme *other*).

Trois types de droits peuvent s'y combiner :

- lecture (symbolisé par `r` comme *read*) ;
- écriture (ou modification, symbolisé par `w` comme *write*) ;
- exécution (symbolisé par `x`comme *eXecute*).

Dans le cas d'un fichier, ces droits sont faciles à interpréter : l'accès en lecture permet d'en consulter le contenu (et notamment de le copier), l'accès en écriture de le modifier, et l'accès en exécution permet de tenter de l'exécuter (ce qui ne fonctionnera que s'il s'agit d'un programme).

SÉCURITÉ **Exécutables `setuid` et `setgid`**

Deux droits particuliers concernent les fichiers exécutables : le bit `setuid` et le bit `setgid` (symbolisés par la lettre « s »). Ils permettent à n'importe quel utilisateur d'exécuter le programme en question avec respectivement les droits de son propriétaire ou de son groupe propriétaire. Ce mécanisme donne accès à des fonctionnalités requérant des droits plus élevés que ceux dont on dispose habituellement.
Un programme `setuid` root s'exécutant systématiquement sous l'identité du super-utilisateur, il est très important d'en contrôler la fiabilité. En effet, un utilisateur capable de le détourner pour lui faire appeler une commande de son choix pourrait alors endosser l'identité de root et avoir tous les droits sur le système.

Un répertoire est traité différemment. L'accès en lecture donne le droit de consulter la liste de ses entrées, l'accès en écriture celui d'y créer ou supprimer des fichiers, et l'accès en exécution de le traverser (et notamment d'en faire le répertoire courant avec la commande **`cd`**). Pouvoir traverser un répertoire sans le lire donne le droit d'accéder à celles de ses entrées dont on connaît le nom, mais pas de les trouver si on ignore leur existence.

SÉCURITÉ **Répertoire `setgid` et sticky bit**

Le bit `setgid` s'applique également aux répertoires. Toutes les entrées qu'on y créera recevront alors pour groupe propriétaire celui du répertoire (si leur auteur en fait partie !), au lieu de prendre comme c'est l'habitude le groupe principal de leur créateur. Cela évitera à celui-ci de changer de groupe principal (par la commande **`newgrp`**) lors d'un travail dans une arborescence partagée entre plusieurs utilisateurs d'un même groupe dédié.
Le bit *sticky* (symbolisé par la lettre « t ») est un droit réservé aux répertoires. Il est notamment employé pour les répertoires temporaires (comme /tmp) puisqu'il limite la suppression d'un fichier à son propriétaire et à celui de son répertoire parent. En son absence, tout le monde pourrait supprimer les fichiers d'autrui sous /tmp, puisque ce répertoire est ouvert en écriture à tous.

Trois commandes manipulent les permissions associées à un fichier :

- **`chown` *`utilisateur fichier`*** affecte un nouveau propriétaire à un fichier ;
- **`chgrp` *`groupe fichier`*** opère sur son groupe propriétaire ;
- **`chmod` *`droits fichier`*** intervient sur ses droits.

Il existe deux manières de présenter les droits ; parmi elles, la représentation symbolique, sans doute la plus simple à comprendre et mémoriser, met en jeu les lettres symboles déjà citées. Pour chaque catégorie d'utilisateurs (`u`/`g`/`o`), on peut définir les droits (`=`), en ajouter (`+`), ou en retrancher (`-`). Ainsi, la formule `u=rwx,g+rw,o-r` donne au propriétaire les droits de lecture, d'écriture, et d'exécution ; ajoute au groupe propriétaire les droits de lecture et d'écriture ;

et supprime le droit de lecture aux autres utilisateurs. Les droits non concernés par les opérations d'ajout ou de retranchement restent inchangés.

La représentation numérique octale associe chaque droit à une valeur : 4 pour la lecture, 2 l'écriture, et 1 pour l'exécution. On associe à chaque combinaison de droits la somme de ces chiffres, valeurs qu'on attribue ensuite aux différentes catégories d'utilisateurs en les mettant bout à bout dans l'ordre habituel (propriétaire, groupe, autres).

La commande **`chmod 765 fichier`** mettra donc en place les droits suivants : lecture, écriture et exécution au propriétaire (car 7 = 4 + 2 + 1) ; lecture et écriture au groupe (car 6 = 4 + 2) ; lecture et exécution aux autres (car 5 = 4 + 1).

Pour représenter le cas échéant les droits spéciaux, on pourra préfixer à ce nombre un quatrième chiffre selon le même principe, sachant que les bits `setuid`, `setgid` et *sticky* valent respectivement 4, 2 et 1. **`chmod 4765`** associera donc le bit `setuid` aux droits décrits précédemment.

POUR ALLER PLUS LOIN **`umask`**

Lorsqu'une application crée un fichier, elle lui donne des permissions indicatives, sachant que le système retire automatiquement certains droits, donnés par la commande **`umask`**. Saisissez **`umask`** dans un shell ; vous observerez un masque tel que `0022`. Ce n'est qu'une représentation octale des droits à retirer systématiquement (en l'occurrence, les droits en écriture pour le groupe et les autres utilisateurs).
Si on lui passe une nouvelle valeur octale, la commande **`umask`** permet également de changer de masque.

ASTUCE **Application récursive**

Il arrive que l'on doive changer les permissions de toute une arborescence. Toutes les commandes décrites disposent donc d'une option **`-R`**, effectuant l'opération demandée de manière récursive.
La distinction entre répertoires et fichiers pose souvent problème lors des opérations récursives. C'est la raison de l'introduction de la lettre « X » dans la représentation symbolique des droits. Elle représente un droit d'exécution qui ne concerne que les répertoires (mais pas les fichiers ne disposant pas encore de ce droit). Ainsi, **`chmod -R a+X répertoire`** n'ajoutera les droits d'exécution pour toutes les catégories d'utilisateurs (`a`) qu'à tous les sous-répertoires et aux fichiers déjà exécutables pour au moins une catégorie d'utilisateurs.

ASTUCE **Changer l'utilisateur et le groupe**

On souhaite souvent changer le groupe d'un fichier en même temps qu'on change celui-ci de propriétaire. La commande **`chown`** propose donc une syntaxe spéciale pour cela : **`chown utilisateur:groupe`**

Interfaces d'administration

Recourir à une interface graphique d'administration est intéressant dans différentes circonstances. Un administrateur ne connaît pas nécessairement tous les détails de configuration de tous ses services, et n'a pas forcément le temps de se documenter à leur sujet. Une interface graphique d'administration accélérera donc le déploiement d'un nouveau service. Par ailleurs, elle pourra simplifier la mise en place des réglages des services les plus pénibles à configurer.

Une telle interface n'est qu'une aide, pas une fin en soi. Dans tous les cas, l'administrateur devra maîtriser son comportement pour comprendre et contourner tout problème éventuel.

Aucune interface n'étant parfaite, on est par ailleurs tenté de recourir à plusieurs solutions. C'est à éviter dans la mesure du possible, car les différents outils sont parfois incompatibles de par leurs hypothèses de travail. Même si tous visent une grande souplesse et tentent d'adopter comme unique référence le fichier de configuration, ils ne sont pas toujours capables d'intégrer des modifications externes.

Administrer sur interface web : `webmin`

C'est sans doute l'une des interfaces d'administration les plus abouties. Il s'agit d'un système modulaire fonctionnant dans un navigateur web et qui dispose de nombreux modules couvrant une vaste palette de domaines et d'outils. Par ailleurs, il est internationalisé et relativement bien traduit en français.

> SÉCURITÉ **Mot de passe root**
>
> À la première connexion, l'identification s'effectue avec l'identifiant « root » et son mot de passe habituel. Il est cependant recommandé de changer dès que possible le mot de passe employé pour *webmin* ; ainsi, une compromission de celui-ci n'impliquera pas le mot de passe de root.
> Attention ! *webmin* étant fonctionnellement très riche, un utilisateur malveillant y accédant pourra vraisemblablement compromettre la sécurité de tout le système. D'une manière générale, les interfaces de ce type sont déconseillées sur les systèmes importants, aux contraintes de sécurité élevées (pare-feu, serveurs sensibles, etc.).

Le paquet *webmin* contient l'infrastructure de base (notamment un mini-serveur web dédié). On trouvera quelques modules traitant de la configuration de base (créer des utilisateurs et des groupes, gérer les fichiers crontab, les scripts d'initialisation, consulter les logs, etc.) dans le paquet *webmin-core*.

Il ne reste plus qu'à installer les modules supplémentaires souhaités. Consultez la longue liste des modules disponibles en exécutant **`apt-cache search webmin-`**. Voici une liste de quelques modules intéressants :

- *webmin-bind* : configuration du serveur DNS (service de noms) ;
- *webmin-postfix* : configuration du serveur SMTP (courrier électronique) ;
- *webmin-inetd* : configuration du super-serveur ;
- *webmin-quota* : gestion des quotas utilisateur ;
- *webmin-dhcpd* : configuration du serveur DHCP ;
- *webmin-proftpd* : configuration du serveur FTP ;
- *webmin-samba* : configuration du serveur Samba ;
- *webmin-software* : installation ou suppression de logiciels à partir des paquets Debian et mise à jour du système.

L'interface d'administration est accessible depuis un navigateur web à l'adresse `https://localhost:10000`. Attention ! tous les modules ne sont pas directement exploitables ; il faut parfois les configurer en précisant les emplacements du fichier de configuration concerné et de quelques exécutables. Souvent, le système vous y invite poliment lorsqu'il n'arrive pas à faire fonctionner le module demandé.

> ALTERNATIVE **Linuxconf**
>
> L'interface d'administration Linuxconf est également disponible en paquet Debian. Moins populaire que *webmin*, elle souffre d'une orientation plus marquée vers les distributions basées sur le format de paquet RPM.

> À SUIVRE **gnome-system-tools**
>
> Le projet GNOME fournit lui aussi une interface graphique d'administration, loin d'être stable même si elle est déjà disponible dans Debian au sein du paquet *gnome-system-tools*.

Configuration des paquets : `debconf`

De nombreux paquets s'auto-configurent après avoir demandé quelques éléments durant l'installation, questions posées à travers l'outil Debconf. On peut reconfigurer ces paquets en exécutant **`dpkg-reconfigure paquet`**.

Dans la plupart des cas, ces réglages sont très simples : seules quelques variables importantes du fichier de configuration sont modifiées. Ces variables sont parfois regroupées entre deux lignes « démarcatrices » de sorte qu'une reconfiguration du paquet limite sa portée sur la zone qu'elles délimitent. Dans d'autres cas, une reconfiguration ne changera rien si le script détecte une modification manuelle du fichier de configuration, l'objectif étant bien évidemment de préserver ces interventions humaines (le script se considère alors incapable d'assurer que ses propres modifications ne perturberont pas l'existant).

CHARTE DEBIAN **Préserver les modifications**

La charte Debian demandant expressément de tout faire pour préserver au maximum les changements manuels apportés aux fichiers de configuration, de plus en plus de scripts modifiant ces derniers prennent des précautions. Le principe général est simple : le script n'effectue des modifications que s'il connaît l'état du fichier de configuration, vérification effectuée par comparaison de la somme de contrôle du fichier avec celle du dernier fichier produit automatiquement. Si elles correspondent, le script s'autorise à modifier le fichier de configuration. Dans le cas contraire, il considère qu'on y est intervenu et demande quelle action il doit effectuer (installer le nouveau fichier, conserver l'ancien, ou tenter d'intégrer les nouvelles modifications au fichier existant). Ce principe de précaution fut longtemps propre à Debian, mais les autres distributions l'embrassent peu à peu.
Le programme **ucf** (du paquet Debian éponyme) offre des facilités pour gérer cela.

Les événements système de `syslog`

Principe et fonctionnement

COMMUNAUTÉ **Martin Schulze**

Martin Schulze est le mainteneur amont de **syslogd** et **klogd**. C'est un développeur Debian de la première heure, qui a de nombreuses responsabilités au sein du projet. Outre ses fonctions de mainteneur de paquets, il est membre des équipes *debian-admin* et *sécurité*, a longtemps participé au processus d'acceptation des nouveaux mainteneurs, et assure la fonction de rédacteur en chef du journal électronique hebdomadaire *Debian Weekly News*. Il gère également les mises à jour périodiques de la version stable de Debian.

Deux démons (**syslogd** et **klogd**) ont pour charge de collecter les messages de service provenant des applications et du noyau puis de les répartir dans des fichiers de logs (habituellement stockés dans le répertoire /var/log). Ils obéissent au fichier de configuration /etc/syslog.conf.

Chaque message de log est associé à un sous-système applicatif (nommé *facility* dans la documentation) :

- `auth` et `authpriv` : concernent l'authentification ;
- `cron` : provient du serveur **cron** ou **atd** ;
- `daemon` : concerne un démon sans classification particulière (serveur DNS, **ntp**, etc.) ;
- `ftp` : concerne le serveur FTP
- `kern` : message provenant du noyau ;
- `lpr` : provient du sous-système d'impression ;
- `mail` : provient de la messagerie électronique ;
- `news` : message du sous-système Usenet (notamment du serveur NNTP — *Network News Transfer Protocol*, ou protocole de transfert des nouvelles sur le réseau — gérant les forums de discussion) ;
- `syslog` : message du serveur **syslogd** lui-même ;

- `user` : messages utilisateur (générique) ;
- `uucp` : messages du sous-système UUCP (*Unix to Unix Copy Program*, ou programme de copie d'Unix à Unix, un vieux protocole employé pour faire circuler entre autres des messages électroniques) ;
- `local0` à `local7` : réservés pour les utilisations locales.

À chaque message est associée une priorité. En voici la liste par ordre décroissant :

- `emerg` : au secours. Le système est probablement inutilisable ;
- `alert` : vite, il y a péril en la demeure, des actions doivent être entreprises immédiatement ;
- `crit` : les conditions sont critiques ;
- `err` : erreur ;
- `warn` : avertissement (erreur potentielle) ;
- `notice` : condition normale mais message significatif ;
- `info` : message informatif ;
- `debug` : message de déboguage.

Le fichier de configuration

La syntaxe complexe du fichier /etc/syslog.conf est détaillée dans la page de manuel `syslog.conf(5)`. Le principe global est d'écrire des paires « sélecteur » et « action ». Le sélecteur définit l'ensemble des messages concernés, et l'action décrit comment le traiter.

Syntaxe du sélecteur

Le sélecteur est une liste (ayant pour séparateur le point-virgule) de couples « *facilité.priorité* » (exemple : `auth.notice;mail.info`). L'astérisque peut y représenter toutes les facilités ou toutes les priorités (exemples : `*.alert` ou `mail.*`). On peut regrouper plusieurs facilités en les séparant par une virgule (exemple : `auth,mail.info`). La priorité indiquée recouvre aussi les messages de priorité supérieure ou égale : `auth.alert` désigne donc les messages du sous-système `auth` de priorités `alert` ou `emerg`. Préfixée par un point d'exclamation, elle désignera au contraire les priorités strictement inférieures : `auth.!notice` désignera donc les messages issus de `auth` et de priorité `info` ou `debug`. Préfixée par un signe égal, elle correspondra exactement à la seule priorité indiquée (`auth.=notice` ne concernera donc que les messages de `auth` de priorité `notice`).

Au sein du sélecteur, chaque élément de la liste surcharge les éléments précédents. Il est donc possible de restreindre un ensemble ou d'en exclure certains éléments. À titre d'exemple, `kern.info;kern.!err` définit les messages du noyau de priorité comprise entre `info` et `warn`. La priorité `none` désigne l'ensemble vide (aucune des priorités), et peut servir pour exclure une facilité d'un ensemble

de messages. Ainsi, `*.crit;kern.none` désigne tous les messages de priorité supérieure ou égale à `crit` ne provenant pas du noyau.

Syntaxe des actions

Les différentes actions possibles sont :

- ajouter le message à un fichier (exemple : /var/log/messages) ;
- envoyer le message à un serveur **`syslog`** distant (exemple : `@log.falcot.com`);
- envoyer le message dans un tube nommé préexistant (exemple : `|/dev/xconsole`);
- envoyer le message à un ou plusieurs utilisateurs s'ils sont connectés (exemple : `root,rhertzog`);
- envoyer le message à tous les utilisateurs connectés (exemple : `*`);
- écrire le message sur une console texte (exemple : `/dev/tty8`).

B.A.-BA **Le tube nommé, un tube persistant**

Un tube nommé est un type particulier de fichier fonctionnant comme un tube traditionnel (*pipe*), mais par l'intermédiaire d'un fichier. Ce mécanisme a l'avantage de pouvoir mettre en relation deux processus n'ayant aucun rapport de parenté. Toute écriture dans un tube nommé est bloquante jusqu'à ce qu'un autre processus tente d'y lire des données. Ce dernier lira alors les données écrites par l'autre partie.
Un tel fichier se crée avec la commande **`mkfifo`**.

SÉCURITÉ **Déporter les logs**

C'est une bonne idée que d'enregistrer les logs les plus importants sur une machine séparée (voire dédiée), car cela empêchera un éventuel intrus de supprimer les traces de son passage (sauf à compromettre également cet autre serveur). Par ailleurs, en cas de problème majeur (tel qu'un plantage noyau), disposer de logs sur une autre machine augmente les chances de retrouver le déroulement des événements.

Le super-serveur `inetd`

Inetd (communément appelé « super-serveur Internet ») est en réalité un serveur de serveurs, employé pour invoquer à la demande les serveurs rarement employés qui ne fonctionnent donc pas en permanence.

Le fichier /etc/inetd.conf donne la liste de ces serveurs et de leurs ports habituels, qu'**`inetd`** écoute tous ; dès qu'il détecte une connexion sur l'un d'entre eux, il exécute le programme du serveur correspondant.

CHARTE DEBIAN **Enregistrer un service dans `inetd.conf`**

Les paquets souhaiteraient parfois enregistrer un nouveau serveur dans le fichier /etc/inetd.conf, mais la charte Debian interdit à tout paquet de modifier un fichier de configuration qui ne relève pas de lui. C'est pourquoi le paquet *inetd* fournit le script **`update-inetd`**, que les autres paquets peuvent employer pour demander au super-serveur de prendre en compte un nouveau serveur.

Chaque ligne significative du fichier /etc/inetd.conf décrit un service par sept champs (séparés par des blancs) :

- Le nom du service (qui définit implicitement le numéro de port TCP ou UDP standard par la correspondance du fichier /etc/services).

- Le type de *socket* : `stream` pour une connexion TCP. `dgram` pour des datagrammes UDP ;
- Le protocole : `tcp` ou `udp`.
- Les options : deux valeurs sont possibles : `wait` ou `nowait`. Ce dernier convient aux connexions TCP, facilement multiplexables. Pour les programmes répondant sur UDP, il ne faut retenir `nowait` que si le serveur est capable de gérer plusieurs connexions en parallèle. On pourra suffixer ce champ d'un point suivi du nombre maximum de connexions autorisées par minute (la limite par défaut étant de 40).
- L'identifiant utilisateur exécutant le serveur.
- Le chemin complet du programme serveur à exécuter.
- Les arguments : il s'agit de la liste complète des arguments du programme, y compris son propre nom (`argv[0]` en C).

L'exemple ci-dessous illustre les cas les plus courants.

EXEMPLE **Extrait de `/etc/inetd.conf`**

```
talk   dgram  udp wait    nobody.tty /usr/sbin/in.talkd in.talkd
finger stream tcp nowait  nobody     /usr/sbin/tcpd     /usr/sbin/in.
  fingerd
ident  stream tcp nowait  nobody     /usr/sbin/identd   identd -i
```

Le programme **`tcpd`** est souvent employé dans le fichier /etc/inetd.conf. Il permet de restreindre les connexions entrantes en appliquant des règles de contrôle, documentées dans la page de manuel `hosts_access(5)` et qui se configurent dans les fichiers /etc/hosts.allow et /etc/hosts.deny.

COMMUNAUTÉ **Wietse Venema**

Wietse Venema, dont les compétences en matière de sécurité en font un programmeur réputé, est l'auteur du programme **`tcpd`**. C'est également l'auteur principal de Postfix, serveur de messagerie électronique (SMTP — *Simple Mail Transfer Protocol*, ou protocole simple de courrier électronique) modulaire conçu pour être plus sûr et plus fiable que **`sendmail`**, au long historique de failles de sécurité.

Planification synchrone : `cron` et `atd`

`cron` est le démon en charge d'exécuter des commandes planifiées et récurrentes (chaque jour, chaque semaine, etc.) ; **`atd`** est celui qui s'occupe des commandes à exécuter une seule fois, à un instant précis et futur.

Dans un système Unix, de nombreuses tâches sont régulièrement planifiées :

- la rotation des logs ;
- la mise à jour de la base de données du programme **`locate`** ;
- les sauvegardes ;
- des scripts d'entretien (comme le nettoyage des fichiers temporaires).

Par défaut, tous les utilisateurs peuvent planifier l'exécution de tâches. C'est pourquoi chacun dispose de sa propre *crontab*, où il peut consigner les commandes à planifier. Il peut la modifier en exécutant **`crontab -e`** (ses informations sont stockées dans le fichier /var/spool/cron/crontabs/<*utilisateur*>).

SÉCURITÉ **Restreindre `cron` ou `atd`**

On peut restreindre l'accès à **`cron`** en créant le fichier d'autorisation explicite /etc/cron.allow, où l'on consignera les seuls utilisateurs autorisés à planifier des commandes. Tous les autres seront automatiquement dépourvus de cette fonctionnalité. Inversement, pour n'en priver qu'un ou deux trouble-fête, on écrira leur nom dans le fichier d'interdiction explicite /etc/cron.deny. Le même mécanisme encadre **`atd`**, avec les fichiers /etc/at.allow et /etc/at.deny.

L'utilisateur root dispose de sa *crontab* personnelle, mais peut également employer le fichier /etc/crontab ou déposer des crontab supplémentaires dans le répertoire /etc/cron.d. Ces deux dernières solutions ont l'avantage de pouvoir préciser l'utilisateur sous l'identité duquel exécuter la commande.

Le paquet *cron* propose par défaut des commandes planifiées qui exécutent :

- une fois par heure les programmes du répertoire /etc/cron.hourly ;
- une fois par jour les programmes du répertoire /etc/cron.daily ;
- une fois par semaine les programmes du répertoire /etc/cron.weekly ;
- une fois par mois les programmes du répertoire /etc/cron.monthly.

De nombreux paquets Debian profitent de ce service pour déposer dans ces répertoires des scripts de maintenance nécessaires au fonctionnement optimal de leur service.

Format d'un fichier `crontab`

Chaque ligne significative d'une crontab décrit une commande planifiée grâce aux six champs suivants :

- la condition sur les minutes (nombres compris de 0 à 59) ;
- la condition sur les heures (de 0 à 23) ;
- la condition sur le jour du mois (de 1 à 31) ;
- la condition sur le mois (de 1 à 12) ;
- la condition sur le jour de la semaine (de 0 à 7, 0 et 7 correspondant au dimanche ; il est également possible d'employer les trois premières lettre du nom du jour en anglais comme `Sun`, `Mon`, etc.) ;
- la commande à exécuter (quand toutes les conditions précédentes sont remplies).

Chaque condition peut s'exprimer sous la forme d'une énumération de valeurs possibles (séparées par des virgules). La syntaxe `a-b` décrit l'intervalle de toutes les valeurs comprises entre `a` et `b`. La syntaxe `a-b/c` décrit un intervalle avec

ASTUCE **Raccourcis textuels pour `cron`**

Des abréviations, qui remplacent les cinq premiers champs d'une entrée de crontab, décrivent les planifications les plus classiques. Les voici :

- `@yearly` : une fois par an (le premier janvier à 0h00) ;
- `@monthly` : une fois par mois (le premier du mois à 0h00) ;
- `@weekly` : une fois par semaine (le dimanche à 0h00) ;
- `@daily` : une fois par jour (à 0h00) ;
- `@hourly` : une fois par heure (au début de chaque heure).

un incrément de c (exemple : `0-10/2` correspond à `0,2,4,6,8,10`). Le joker `*` représente toutes les valeurs possibles.

EXEMPLE **Exemple de `crontab`**

```
#Format
#min heu jou moi jsem commande

# Télécharge les données tous les soirs à 19:25
 25  19  *   *   *    $HOME/bin/get.pl

# Le matin à 8h00, en semaine (lundi à vendredi)
 00  08  *   *   1-5  $HOME/bin/fait_quelquechose

# Redémarre le proxy IRC après chaque reboot
@reboot /usr/bin/dircproxy
```

ASTUCE **Exécuter une commande au démarrage**

Pour exécuter une commande une seule fois, juste après le démarrage de l'ordinateur, on peut recourir à la macro `@reboot` (un simple redémarrage de **`cron`** ne déclenche pas une commande planifiée avec `@reboot`). Cette macro remplace elle aussi les cinq premiers champs d'une entrée dans la crontab.

Emploi de la commande `at`

La commande **`at`** prévoit l'exécution d'une commande à un moment ultérieur. Elle prend en paramètre l'horaire et la date prévus et sur son entrée standard, la commande à exécuter alors. Cette dernière sera exécutée comme si elle avait été saisie dans un interpréteur de commandes. **`at`** conserve d'ailleurs l'environnement courant afin de pouvoir travailler exactement dans les mêmes conditions que celles de la planification. L'horaire est indiqué en suivant les conventions habituelles : `16:12` représente 16h12. La date peut être précisée au format `JJ.MM.AA` (`27.07.04` représentant ainsi 27 juillet 2004). En son absence, la commande sera exécutée dès que l'horloge atteindra l'heure signalée (le jour même ou le lendemain). On peut encore écrire explicitement *today* (aujourd'hui) ou *tomorrow* (demain).

EXEMPLE **Exemple d'emploi de la commande `at`**

```
$ cat <<FIN | at 16:12 27.07.05
> echo "Joyeux anniversaire" | mail hertzog@debian.org
> FIN
warning: commands will be executed using /bin/sh
job 2 at 2005-07-27 16:12
```

Une autre syntaxe permet d'exprimer une durée d'attente : **`at now +` *`nombre période`***. *période* peut valoir `minutes`, `hours` (heures), `days` (jours) ou `weeks` (semaines). *nombre* indique simplement le nombre de ces unités qui doivent s'écouler avant exécution de la commande.

On peut toujours annuler une commande planifiée avec la commande **`atrm` *`numéro-de-tâche`***. Le numéro de tâche est indiqué par la commande **`at`** lors de la planification mais on pourra le retrouver grâce à la commande **`atq`**, qui donne la liste des commandes actuellement planifiées.

Planification asynchrone : anacron

anacron est le démon qui complète **cron** pour les ordinateurs non allumés en permanence. Les tâches régulières étant habituellement planifiées au milieu de la nuit, elles ne seront jamais exécutées si la machine est éteinte à ce moment-là, **anacron** tente de les exécuter en prenant en compte les périodes où l'ordinateur ne fonctionnait pas.

Attention, **anacron** fera fréquemment exécuter cette activité en retard quelques minutes après le démarrage de la machine, ce qui peut en perturber la réactivité. C'est pourquoi les tâches du fichier /etc/anacrontab sont démarrées sous la commande **nice**, qui réduit leur priorité d'exécution et limitera donc l'impression de lenteur.

L'installation d'**anacron** désactive l'exécution par **cron** des scripts des fichiers /etc/cron.hourly, /etc/cron.daily, /etc/cron.weekly, et /etc/cron.monthly. On évite ainsi qu'ils soient pris en compte trop souvent (par **anacron** et par **cron**). Mais **cron** reste actif et se chargera encore d'exécuter les autres commandes planifiées (notamment par les utilisateurs).

Les quotas

Le système des quotas permet de limiter l'espace disque alloué à un utilisateur ou un groupe d'utilisateurs. Pour le mettre en place, il faut disposer d'un noyau activant sa prise en charge (option de compilation `CONFIG_QUOTA`) — ce qui est le cas des noyaux Debian. Les logiciels de gestion des quotas se trouvent dans le paquet Debian *quota*.

Pour les activer sur un système de fichiers, il faut indiquer dans le fichier /etc/fstab `usrquota` et `grpquota`, respectivement pour des quotas utilisateurs ou de groupes. Redémarrer l'ordinateur permet ensuite de mettre à jour les quotas en l'absence d'activité disque (condition nécessaire à une bonne comptabilité de l'espace disque déjà consommé).

La commande **edquota *utilisateur*** (ou **edquota -g *groupe***) permet de changer les limites tout en consultant la consommation actuelle.

> POUR ALLER PLUS LOIN **Définir les quotas par script**
>
> Le programme **setquota** peut être employé dans un script pour modifier automatiquement de nombreux quotas. Sa page de manuel détaille la syntaxe précise à employer.

Le système de quotas permet de définir quatre limites :

- deux limites (*soft* et *hard*, respectivement douce et dure) concernent le nombre de blocs consommés. Un bloc — petite zone d'espace disque — contient habituellement jusqu'à 1024 octets du même fichier. Les blocs non saturés induisent donc des pertes d'espace disque. Un quota de 100 blocs, qui permet

théoriquement de stocker 102 400 octets, sera pourtant saturé par 100 fichiers de 500 octets, ne représentant que 50 000 octets au total.

- deux limites (*soft* et *hard*) concernent le nombre d'*inodes* employés. Chaque fichier consomme au moins un *inode* pour stocker les informations le concernant (droits, propriétaires, date de dernier accès, etc.). Il s'agit donc d'une limite sur le nombre de fichiers de l'utilisateur.

Une limite *soft* peut être franchie temporairement ; l'utilisateur sera simplement averti de son dépassement de quota par le programme **`warnquota`**, habituellement invoqué par **`cron`**. Une limite *hard* ne peut jamais être franchie : le système refusera toute opération provoquant un dépassement du quota dur.

On peut définir, par la commande **`edquota -t`**, une « période de grâce » maximale autorisée pour un dépassement de limite *soft*. Ce délai écoulé, la limite *soft* se comportera comme une limite *hard* et l'utilisateur devra donc repasser sous elle pour pouvoir à nouveau écrire quoi que ce soit sur le disque.

POUR ALLER PLUS LOIN **Systématiser un quota pour les nouveaux utilisateurs**

Pour instaurer un quota systématique chez les nouveaux utilisateurs, il faut le configurer sur un utilisateur « modèle » (avec **`edquota`** ou **`setquota`**) et indiquer son nom dans la variable QUOTAUSER du fichier /etc/adduser.conf. Ce paramétrage sera alors automatiquement repris pour chaque nouvel utilisateur créé avec la commande **`adduser`**.

Supervision

La supervision consiste à surveiller l'activité de l'ordinateur pour s'assurer que tout se passe bien et qu'il est bien utilisé comme prévu. Les logs permettent de comprendre le passé, mais on peut aussi voir ce qui se passe en temps réel.

Surveillance des logs avec `logcheck`

Le programme **`logcheck`** scrute les fichiers de logs toutes les heures et envoie par courrier électronique à « root » les plus inhabituels pour aider à détecter tout nouveau problème.

La liste des fichiers scrutés se trouve dans le fichier /etc/logcheck/logcheck.logfiles ; les choix par défaut conviendront si le fichier /etc/syslog.conf n'a pas été complètement remodelé.

`logcheck` peut fonctionner en 3 modes plus ou moins détaillés : *paranoid* (paranoïaque), *server* (serveur), et *workstation* (station de travail). Le premier étant le plus verbeux, on le réservera aux serveurs spécialisés (comme les pare-feu). Le deuxième mode, choisi par défaut, est recommandé pour les serveurs. Le dernier, prévu pour les stations de travail, élimine encore plus de messages.

Dans tous les cas, il faudra probablement paramétrer **logcheck** pour exclure des messages supplémentaires (selon les services installés) sous peine d'être envahi chaque heure par une flopée de messages inintéressants. Leur mécanisme de sélection étant relativement complexe, il faut lire à tête reposée le document /usr/share/doc/logcheck-database/README.logcheck-database.gz pour bien le comprendre.

Plusieurs types de règles sont appliquées :

- celles qui qualifient un message comme résultant d'une tentative d'attaque (/etc/logcheck/cracking.d/) ;
- celles qui annulent cette qualification (/etc/logcheck/cracking.ignore.d/) ;
- celles qui qualifient un message comme une alerte de sécurité (/etc/logcheck/violations.d/) ;
- celles qui annulent cette qualification (/etc/logcheck/violations.ignore.d) ;
- et enfin celles qui s'appliquent à tous les messages restants (les *System Events*, ou événements système).

ATTENTION **Ignorer un message**

Tout message marqué comme une tentative d'attaque ou une alerte de sécurité (suite par exemple à une règle du fichier /etc/logcheck/violations.d/monfichier) ne pourra être ignoré que par une règle des fichiers /etc/logcheck/violations.ignore.d/monfichier ou /etc/logcheck/violations.ignore.d/monfichier-<*extension*>.

Un événement système sera systématiquement signalé, sauf si une règle de l'un des répertoires /etc/logcheck/ignore.d.*/ dicte de l'ignorer. Évidemment, seuls les répertoires correspondant à des niveaux de verbosité supérieurs ou égaux au niveau sélectionné sont pris en compte.

Surveillance de l'activité

En temps réel

ASTUCE **Vos logs en fond d'écran**

Certains administrateurs aiment voir les messages de logs défiler en temps réel. Ils pourront les intégrer dans le fond d'écran de leur bureau graphique avec la commande **root-tail** (du paquet Debian éponyme). Le programme **xconsole** (du paquet *xbase-clients*) les fera défiler dans une petite fenêtre ; les messages sont directement issus de **syslogd** par l'intermédiaire du tube nommé /dev/xconsole.

top est un utilitaire interactif qui affiche la liste des processus en cours d'exécution. Par défaut, son critère de tri est l'utilisation actuelle du processeur (touche **p**) mais on peut opter pour la mémoire occupée (touche **M**), le temps processeur consommé (touche **T**) ou le numéro de processus ou PID (touche **N**). La touche **k** permet d'indiquer un numéro de processus à tuer.

Si le processeur semble être surchargé, il est ainsi possible d'observer quels processus se battent pour son contrôle ou consomment toute la mémoire disponible. **top** est un outil de base très souple, et sa page de manuel explique comment en personnaliser l'affichage pour l'adapter aux besoins et aux habitudes de chacun.

gnome-system-monitor et **qps**, outils graphiques similaires à **top**, en proposent les principales fonctionnalités.

Historique

La charge du processeur, le trafic réseau ou l'espace disque disponible sont des informations qui varient en permanence. Il est souvent intéressant de garder une trace de leur évolution pour mieux cerner l'usage qui est fait de l'ordinateur.

C'est tout l'intérêt d'un outil comme **`cacti`**, programme complet de suivi d'historique d'informations système, capable d'utiliser SNMP (*Simple Network Management Protocol*, ou protocole simple de gestion du réseau) pour surveiller d'autres éléments du réseau. Après installation du paquet *cacti*, on pourra le configurer à travers une interface web, accessible à l'adresse `http://localhost/cacti` (à condition d'utiliser le serveur web **`apache`** et de l'avoir obligé à relire ses fichiers de configuration par la commande **`/etc/init.d/apache reload`**). Le compte `admin` (de mot de passe `admin`) est configuré par défaut, mais le système demande d'en changer le mot de passe dès la première connexion.

Sa puissance complique malheureusement l'installation de **`cacti`**, et il faudra lire sa documentation pour savoir comment configurer de nouvelles données à surveiller. On la trouve à partir du fichier /usr/share/doc/cacti/html/index.html.

ALTERNATIVE **`mrtg`**

`mrtg` (du paquet Debian éponyme) est un outil plus ancien et plus rustique capable d'agréger des données historiques et d'en faire des graphiques. Il dispose d'un certain nombre de scripts de récupération des données les plus couramment surveillées : charge, trafic réseau, impacts (*hits*) web, etc.
Les paquets *mrtg-contrib* et *mrtgutils* contiennent des scripts d'exemples, prêts à l'emploi.

Sauvegarde

L'une des responsabilités principales de tout administrateur, la sauvegarde reste un sujet complexe dont les outils puissants sont en général difficiles à maîtriser.

De nombreux logiciels existent : citons **`amanda`**, un système client/serveur doté de nombreuses options, mais plutôt difficile à configurer. Des dizaines d'autres paquets Debian sont dédiés à des solutions de sauvegarde, comme vous le montrera la commande **`apt-cache search backup`**.

Plutôt que de détailler le fonctionnement de certains d'entre eux, je vais exposer la réflexion menée par les administrateurs de Falcot SA pour définir leur stratégie de sauvegarde.

Chez Falcot SA, les sauvegardes répondent à deux besoins : récupérer des fichiers supprimés par erreur et remettre en route rapidement tout ordinateur (serveur ou bureautique) dont le disque dur subit une panne.

Les sauvegardes sur bandes ayant été jugées trop lentes et trop coûteuses, les données seront sauvegardées sur les disques durs d'un serveur dédié, où l'emploi

du RAID logiciel les protégera d'une défaillance du disque. Les ordinateurs bureautiques ne sont pas sauvegardés individuellement, mais les utilisateurs sont informés que leur compte personnel, situé sur le serveur de fichiers de leur département, sera sauvegardé. La commande **rsync** (du paquet éponyme) sauvegarde quotidiennement ces différents serveurs.

L'espace disque disponible interdit la mise en place d'une sauvegarde complète quotidienne. C'est pourquoi la synchronisation par **rsync** est précédée d'une duplication du contenu de la dernière sauvegarde par des liens durs (*hard links*), qui évitent de consommer trop d'espace disque. Le processus **rsync** ne remplacera ensuite que les fichiers modifiés depuis la dernière sauvegarde. Ce mécanisme permet de conserver un grand nombre de sauvegardes sur un volume réduit. Toutes les sauvegardes étant accessibles en même temps, on pourra effectuer rapidement des comparaisons entre deux dates données.

EN PRATIQUE **Sauvegarde avec rsync**

Celle de Falcot SA est documentée sur le site web indiqué ci-dessous.

▸ http://www.mikerubel.org/computers/rsync_snapshots/

Les ordinateurs de bureau, non sauvegardés, seront faciles à régénérer à partir des cédéroms fabriqués par le programme **mondo**. Amorçables, ces cédéroms permettent de réinstaller complètement le système de la machine.

ALTERNATIVE **systemimager**

Le logiciel *systemimager* permet lui aussi de restaurer rapidement des ordinateurs complets, à partir d'une image stockée par exemple sur un serveur.

Les administrateurs de Falcot SA sont conscients des limites de leur politique de sauvegarde. Ne pouvant pas protéger le serveur de sauvegarde aussi bien qu'une bande dans un coffre ignifugé, ils l'ont installé dans une pièce séparée de sorte qu'un sinistre tel qu'un incendie se déclarant dans la salle des serveurs ne détruise pas aussi les sauvegardes. Par ailleurs, ils réalisent une sauvegarde incrémentale sur dévédérom une fois par semaine — seuls les fichiers modifiés depuis la dernière sauvegarde sont concernés.

POUR ALLER PLUS LOIN **Sauvegarde SQL, LDAP**

De nombreux services (comme les bases de données SQL ou LDAP) ne peuvent pas être sauvegardés simplement en copiant leurs fichiers (sauf s'ils sont correctement interrompus durant la sauvegarde, ce qui pose souvent problème car ils sont prévus pour être disponibles en permanence). Il alors nécessaire de faire appel à une procédure « d'export » des données, dont on sauvegardera alors le *dump*. Souvent volumineux, celui-ci se compacte bien. Pour réduire l'espace de stockage nécessaire, on ne stockera qu'un fichier texte complet par semaine et un **diff** chaque jour. Le programme **xdelta** produira les différences incrémentales des *dumps* binaires.

B.A.-BA **Le lien dur, un deuxième nom pour le fichier**

Un lien dur (*hardlink*), contrairement au lien symbolique, ne peut être différencié du fichier pointé. Créer un lien dur revient en fait à affecter un deuxième nom au fichier cible. C'est pourquoi la suppression d'un lien dur ou de la cible ne supprime en fait qu'un des noms associés au fichier. Tant qu'un nom est encore affecté au fichier, les données de celui-ci restent présentes sur le système de fichiers. Il est intéressant de noter que contrairement à une copie, le lien dur ne consomme pas d'espace disque supplémentaire. Le lien dur se crée avec la commande **ln *cible lien***. Le fichier *lien* est alors un nouveau nom du fichier *cible*. Les liens durs ne peuvent être créés qu'au sein d'un même système de fichiers, alors que les liens symboliques ne souffrent pas de cette limitation.

CULTURE **TAR, standard de sauvegarde sur bande**

Historiquement, le moyen le plus simple de réaliser une sauvegarde sous Unix était de stocker sur bande une archive au format *TAR*. La commande **tar** tire d'ailleurs son nom de *Tape ARchive* (« archive sur bande »).

Branchements « à chaud » : `hotplug`

Le programme **`hotplug`** collabore avec le noyau pour y charger les pilotes des périphériques qui peuvent se connecter à chaud. Cela inclut les périphériques USB, PCMCIA, IEEE 1394, et même, pour certains serveurs haut de gamme, SCSI ou PCI. La base de données de **`hotplug`** associe à chaque identifiant de périphérique le pilote requis. Dans le cas des cartes réseau, **`hotplug`** tente de configurer l'interface correspondante avec **`ifup`** (en suivant donc les paramétrages indiqués dans le fichier /etc/network/interfaces).

Non content de détecter les périphériques insérés alors que l'ordinateur est sous tension, le script d'initialisation du paquet **`hotplug`** effectue un *coldplug* (branchement à froid), c'est-à-dire une configuration des périphériques déjà connectés comme s'ils venaient d'être insérés.

`hotplug` peut être désactivé avec la commande **`dpkg-reconfigure hotplug`**.

Gestion de l'énergie

La question de la gestion de l'énergie reste souvent problématique. En effet, une mise en veille réussie requiert que les pilotes de tous les périphériques de l'ordinateur sachent se désactiver et surtout reconfigurer le périphérique au réveil. Malheureusement, il subsiste de nombreux périphériques incapables de bien se mettre en veille sous Linux car leurs constructeurs n'en ont pas fourni les spécifications.

À SUIVRE **Software suspend**

La bannière *software suspend* rassemble plusieurs efforts récents visant à intégrer à Linux une hibernation fiable, sur disque ou en mémoire. Le fruit de ces efforts trouvera probablement son chemin dans l'un des futurs noyaux de la série 2.6.

Gestion avancée de l'énergie : APM

La prise en charge de l'APM (*Advanced Power Management*), présente dans tous les noyaux de Debian, y est désactivée par défaut. Pour l'activer, on passera l'option `apm=on` à la ligne de commande démarrant le noyau. Avec LILO, on ajoutera la directive `append="apm=on"` au bloc décrivant l'image à démarrer (dans le fichier /etc/lilo.conf). Dans le cas de GRUB, il suffit d'ajouter `apm=on` à la ligne débutant par `kernel` (il s'agit ici du fichier /boot/grub/menu.lst).

Le paquet *apmd* fournit un démon qui attend des événements liés à la gestion de l'énergie (basculement entre le mode secteur et la batterie pour un ordinateur portable, etc.) et permet d'exécuter des commandes particulières en réaction.

Économie d'énergie moderne : ACPI

Les derniers noyaux 2.4 et les noyaux 2.6 proposent l'ACPI (*Advanced Configuration and Power Interface*, interface avancée de configuration et d'énergie), le

MATÉRIEL **Apple Powerbook et la gestion d'énergie**

Sur les Apple Powerbook, on remplacera **apmd** par **pmud**.

standard le plus récent en matière d'économie d'énergie. Plus souple et plus puissant, il est aussi plus compliqué à mettre en œuvre. Le paquet *acpid* est le pendant d'*apmd* pour le monde ACPI.

Si vous savez que votre BIOS gère correctement ACPI, alors il doit être préféré à APM (quitte à mettre à jour le BIOS), sinon le plus sage est de rester avec APM. En migrant de l'un à l'autre, il faut veiller à supprimer le paquet *apmd* car sa cohabitation avec *acpid* pourrait poser problème (et vice versa).

ATTENTION **Carte graphique et mise en veille**

Le pilote de la carte graphique pose souvent problème lors de la mise en veille. En cas de souci, il convient donc de tester la dernière version du serveur graphique *XFree86*.

Cartes pour portables : PCMCIA

PCMCIA requiert deux paquets : *pcmcia-cs* (PCMCIA *Card Services*) héberge les démons qui réagissent à l'insertion de nouvelles cartes PCMCIA en chargeant le pilote correspondant, tandis que *kernel-pcmcia-modules-version-noyau* fournit les pilotes eux-mêmes — des modules noyau comme la majorité des pilotes Linux.

Les utilisateurs de noyaux 2.4 peuvent aussi trouver des paquets *pcmcia-modules-version-noyau* contenant des pilotes plus à jour. À l'époque le développement des pilotes PCMCIA était réalisé en dehors du noyau officiel et deux versions coexistaient donc en parallèle (celle du noyau était systématiquement plus ancienne puisqu'elle n'était synchronisée avec le tronc de développement principal qu'assez irrégulièrement).

ASTUCE **Vérifier la compatibilité d'une carte PCMCIA**

À chaque achat d'un périphérique, se pose la question de sa compatibilité avec Linux. Heureusement, en ce qui concerne les cartes PCMCIA, David Hinds maintient une liste des cartes reconnues et fonctionnelles :

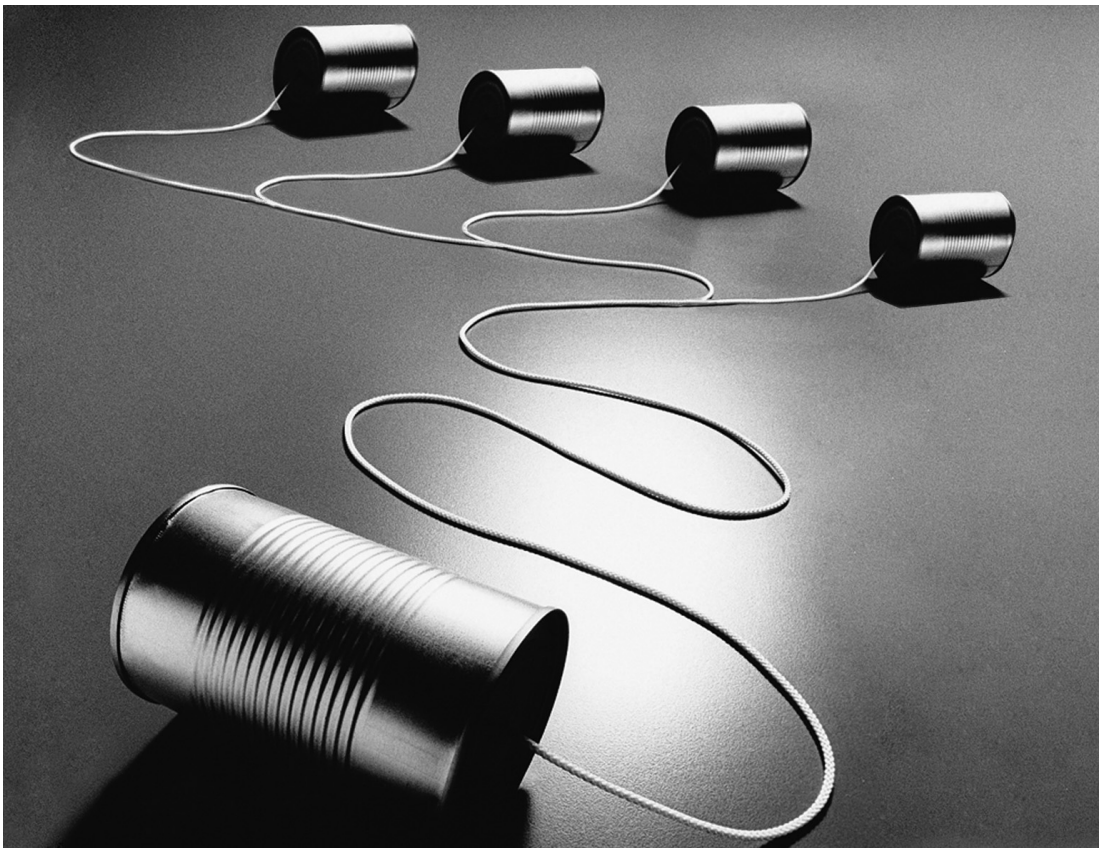
▸ http://pcmcia-cs.sourceforge.net/ftp/SUPPORTED.CARDS

Le paquet *wireless-tools* sera aussi nécessaire à la bonne prise en charge des cartes de connexion sans fil *Wifi*.

À chaque insertion ou éjection d'une carte, le démon la configure ou déconfigure en exécutant un script du répertoire /etc/pcmcia, qui trouve ses paramètres dans les fichiers /etc/pcmcia/*.opts. Ces fichiers ont été légèrement adaptés pour s'intégrer à un système Debian : la configuration du réseau est ainsi déléguée à **ifup** si le fichier /etc/pcmcia/network.opts en est dépourvu. Il en va de même pour la configuration d'un réseau sans fil, qui peut être spécifiée dans /etc/network/interfaces au lieu de /etc/pcmcia/wireless.opts. Le fichier /usr/share/doc/wireless-tools/README.Debian décrit d'ailleurs la syntaxe à employer.

Après ce survol des services de base communs à de nombreux Unix, nous nous focaliserons sur l'environnement dans lequel évoluent les machines administrées : le réseau. De nombreux services sont en effet nécessaires à son bon fonctionnement — je vous propose de les découvrir dans le chapitre qui suit.

10



Infrastructure réseau

Linux n'a plus à faire ses preuves dans le domaine des réseaux, et Debian dispose de toute la panoplie des outils existants pour les créer et gérer. Ce chapitre les passe en revue.

SOMMAIRE

- Passerelle
- Pare-feu ou filtre de paquets
- Réseau privé virtuel
- Qualité de service
- Routage dynamique
- IPv6
- Serveur de noms (DNS)
- DHCP
- Détection d'intrusion (IDS/NIDS)

MOTS-CLEFS

- Réseau
- Passerelle
- TCP/IP
- IPV6
- DNS
- Bind
- Pare-feu
- Netfilter
- DHCP
- QoS

Passerelle

Une passerelle relie plusieurs réseaux entre eux. Ce terme désigne souvent la « porte de sortie » d'un réseau local, point de passage obligé pour atteindre toutes les adresses IP externes. La passerelle est connectée à chacun des réseaux qu'elle relie et agit en tant que routeur pour rediriger les paquets IP entre ses différentes interfaces.

Lorsqu'un réseau local utilise une plage d'adresses privées (non routables sur l'Internet), la passerelle doit effectuer du *masquerading* (masquage d'adresses IP) pour que ses machines puissent communiquer avec l'extérieur. L'opération consiste à remplacer chaque connexion sortante par une connexion provenant de la passerelle elle-même (disposant, elle, d'une adresse valable sur le réseau externe) puis à faire suivre les données reçues en réponse à la machine ayant initié la connexion. Pour mener à bien cette tâche, la passerelle dispose d'une plage de ports TCP dédiés au *masquerading* (il s'agit souvent de numéros de port très élevés, supérieurs à 60000). Chaque nouvelle connexion issue d'une machine interne apparaîtra à l'extérieur comme provenant de l'un de ces ports réservés. Lorsque la passerelle reçoit une réponse sur l'un d'entre eux, elle sait à quelle machine la faire suivre.

La passerelle peut également effectuer une traduction d'adresses réseau (*NAT*, ou *Network Address Translation*). Il en existe de deux types. Le *Destination NAT* (DNAT) est une technique pour altérer l'adresse IP (et/ou le port TCP ou UDP) destinataire d'une nouvelle connexion (généralement entrante). Le mécanisme de « suivi des connexions » (*connection tracking*) altérera aussi les autres paquets de la même connexion pour assurer la continuité de la communication. Son pendant, le *Source NAT* (SNAT), et dont le *masquerading* est un cas particulier, altère l'adresse IP (et/ou le port TCP ou UDP) source d'une nouvelle connexion (généralement sortante). Comme pour le DNAT, le suivi des connexions gère de manière adéquate les paquets suivants.

Après la théorie, place à la pratique. Il est très facile de transformer un système Debian en passerelle : il suffit d'activer l'option adéquate du noyau Linux. On peut pour cela procéder par l'intermédiaire du système de fichiers virtuels */proc* :

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Pour activer cette option automatiquement à chaque démarrage, on positionnera dans le fichier /etc/network/options l'option `ip_forward` à `yes`.

EXEMPLE **Fichier /etc/network/options**

```
ip_forward=yes
spoofprotect=yes
syncookies=no
```

Activer le *masquerading* est une opération plus complexe, nécessitant de configurer le pare-feu *netfilter*. Le paquet *ipmasq* s'en charge heureusement : après son

B.A.-BA **Paquet IP**

Les réseaux employant le protocole IP pour échanger des informations fonctionnent par paquets : les données y circulent de manière intermittente dans des blocs de tailles limitées. Chaque paquet contient en plus des données toutes les informations nécessaires à son acheminement (ou routage).

B.A.-BA **Port TCP/UDP**

Un port TCP ou UDP est un point d'attache pour établir une connexion avec une machine. Ce concept permet d'avoir plusieurs communications différenciées avec le même interlocuteur, le numéro de port étant le critère distinctif.
Certains de ces numéros — standardisés par l'IANA (*Internet Assigned Numbers Authority*, ou autorité Internet d'attribution des numéros) — sont associés à des services réseau. Par exemple, le port 25 est généralement employé par le serveur de courrier électronique.

▸ http://www.iana.org/assignments/port-numbers

CULTURE **Plages d'adresses privées**

La RFC 1918 définit trois plages d'adresses IP à ne pas router sur l'Internet, prévues pour un usage dans des réseaux locaux. La première, `10.0.0.0/8`, est une plage de classe A (contenant 2^{24} adresses IP). La deuxième, `172.16.0.0/12`, rassemble 16 plages de classe B (`172.16.0.0/16` à `172.31.0.0/16`) pouvant contenir chacune 2^{16} adresses IP. La dernière, `192.168.0.0/16`, est une plage de classe B (regroupant les 256 plages de classe C `192.168.0.0/24` à `192.168.255.0/24`, de 256 adresses IP chacune).

▸ http://www.faqs.org/rfcs/rfc1918.html

B.A.-BA **Port forwarding**

Le *port forwarding*, dont le principe est de rediriger (« faire suivre ») une connexion entrant sur un port donné vers un port d'une autre machine, se réalise facilement à partir d'une technique de DNAT. D'autres solutions techniques existent cependant pour obtenir un résultat similaire : notamment avec des redirections au niveau applicatif grâce à **`ssh`** ou **`redir`**.

installation, il configure automatiquement le *masquerading* à chaque démarrage de l'ordinateur. Une option permet aussi de n'activer celui-ci qu'à l'ouverture d'une connexion PPP (cas d'une connexion sortante intermittente).

L'emploi du NAT nécessite lui aussi de configurer *netfilter*. L'absence de configuration standard explique qu'il n'y ait pas de solution prête à l'emploi. Des outils permettent en revanche de simplifier la configuration du pare-feu *netfilter* en visualisant graphiquement les règles définies. L'un des meilleurs est sans doute **`fwbuilder`**, abordé dans la section suivante traitant des pare-feu.

Pare-feu ou filtre de paquets

Un pare-feu est une passerelle filtrante : il applique des règles de filtrage aux paquets qui le traversent (c'est pourquoi il n'est utile qu'en tant que point de passage obligé).

B.A.-BA **Pare-feu**

Un pare-feu (*firewall*) est un ensemble matériel ou logiciel qui trie les paquets qui circulent par son intermédiaire en provenance ou vers le réseau local, et ne laisse passer que ceux qui vérifient certaines conditions.

CAS PARTICULIER **Pare-feu local**

Un pare-feu peut limiter son action à une seule machine (et non pas un réseau local complet) ; son rôle principal est alors de refuser ou limiter l'accès à certains services.

Les noyaux Linux des séries 2.4 et 2.6 intègrent le pare-feu *netfilter*, que l'outil **`iptables`** permet de configurer.

Fonctionnement de *netfilter*

netfilter dispose de trois tables distinctes, donnant les règles régissant trois types d'opérations sur les paquets :

- `filter` pour les règles de filtrage (accepter, refuser, ignorer un paquet) ;
- `nat` pour modifier les adresses IP et les ports source ou destinataires des paquets ;
- `mangle` pour modifier d'autres paramètres des paquets IP (notamment le champ ToS — *Type Of Service* — et les options).

Chaque table contient des listes de règles appelées « chaînes » ; les chaînes standards servent au pare-feu pour traiter les paquets dans différentes circonstances prédéfinies. L'administrateur peut créer d'autres chaînes, qui ne seront employées que si l'une des chaînes standards les appelle.

La table `filter` compte trois chaînes standards :

- `INPUT` : concerne les paquets destinés au pare-feu ;
- `OUTPUT` : concerne les paquets émis par le pare-feu ;
- `FORWARD` : appliquée aux paquets transitant via le pare-feu (et dont il n'est donc ni la source ni le destinataire).

La table `nat` dispose également de trois chaînes standards :

- `PREROUTING` : modifie les paquets dès qu'ils arrivent ;
- `POSTROUTING` : modifie les paquets alors qu'ils sont prêts à partir ;
- `OUTPUT` : modifie les paquets générés par le pare-feu lui-même.

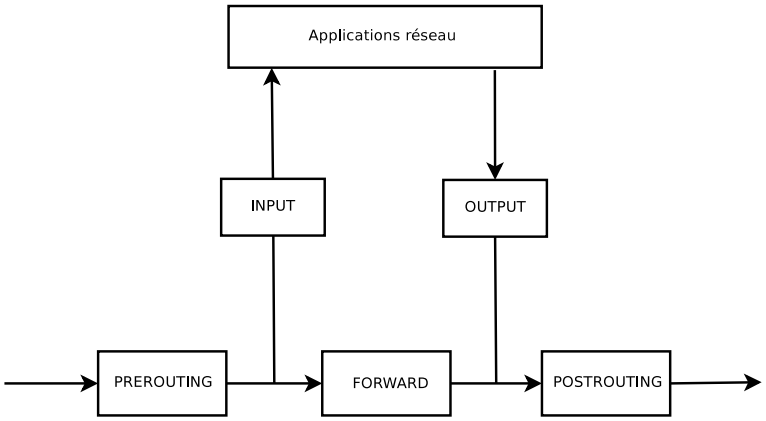


Figure 10–1 Ordre d'emploi des chaînes de netfilter

Chaque chaîne est une liste de règles, prévoyant une action à exécuter quand certaines conditions sont remplies. Le pare-feu parcourt séquentiellement la chaîne s'appliquant au paquet traité, exécutant l'action indiquée dès qu'une règle est satisfaite. Les actions possibles sont les suivantes :

- `ACCEPT` : autoriser le paquet à poursuivre son parcours ;
- `REJECT` : rejeter le paquet (ICMP signale une erreur, l'option **`--reject-with`** ***`type`*** d'**`iptables`** permet de choisir le type d'erreur renvoyée) ;
- `DROP` : supprimer (ignorer) le paquet ;
- `LOG` : enregistrer un message de log via **`syslogd`** (ce message contient des informations sur le paquet traité — on emploie souvent cette fonctionnalité juste avant un refus afin de garder une trace d'éventuels problèmes) ;
- `ULOG` : enregistrer un message de log via **`ulogd`**, plus adapté et plus efficace que **`syslogd`** pour gérer de grandes quantités de messages ;
- *nom_de_chaîne* : évaluer les règles de la chaîne indiquée ;
- `RETURN` : stopper l'évaluation de la chaîne courante et revenir sur la chaîne appelante (si la chaîne courante est une chaîne standard, dépourvue de chaîne appelante, effectuer l'action par défaut) ;
- `SNAT` : effectuer du *Source NAT* (des options précisent les modifications à effectuer) ;
- `DNAT` : effectuer du *Destination NAT* (des options précisent les modifications à effectuer) ;
- `MASQUERADE` : effectuer du **`masquerading`** (SNAT particulier) ;
- `REDIRECT` : rediriger un paquet vers un port particulier du pare-feu lui-même ; action notamment utile pour mettre en place un mandataire (ou proxy) web transparent.

B.A.-BA **ICMP**

ICMP (*Internet Control Message Protocol*, ou protocole des messages de contrôle sur Internet) est très employé pour transmettre des compléments d'informations sur les communications : il permet de tester le fonctionnement du réseau avec la commande **`ping`**, signale le refus d'un paquet par un pare-feu, indique la saturation d'un tampon de réception, propose une meilleure route, etc. Plusieurs RFC définissent ce protocole ; les premières, 777 et 792, furent rapidement complétées et étendues.

▸ http://www.faqs.org/rfcs/rfc777.html

▸ http://www.faqs.org/rfcs/rfc792.html

Rappelons qu'un tampon de réception est une petite zone mémoire contenant les données reçues par le réseau avant qu'elles ne soient traitées par le noyau. Si cette zone est pleine, il est alors impossible de recevoir d'autres données et ICMP signale le problème de sorte que le correspondant réduise la vitesse de transfert pour essayer d'atteindre un équilibre.

D'autres actions, concernant davantage la table `mangle`, ne sont pas mentionnées ici. Vous en trouverez la liste exhaustive dans la page de manuel `iptables(8)`.

Syntaxe d'`iptables`

La commande **`iptables`** permet de manipuler les tables, les chaînes et les règles. L'option **`-t table`** indique la table sur laquelle opérer (par défaut, c'est `filter`).

Les commandes

L'option **`-N chaîne`** crée une nouvelle chaîne ; l'option **`-X chaîne`** supprime une chaîne vide et inutilisée L'option **`-A chaîne règle`** ajoute une règle à la fin de la chaîne indiquée. L'option **`-I chaîne numrègle règle`** insère une règle avant la règle numérotée *numrègle*. L'option **`-D chaîne numrègle`** ou **`-D chaîne règle`** supprime une règle dans la chaîne (la première syntaxe l'identifie par son numéro et la seconde par son contenu). L'option **`-F chaîne`** supprime toutes les règles de la chaîne (si celle-ci n'est pas mentionnée, elle supprime toutes les règles de la table). L'option **`-L chaîne`** affiche le contenu de la chaîne. Enfin, l'option **`-P chaîne action`** définit l'action par défaut pour la chaîne donnée (seules les chaînes standards peuvent en avoir une).

Les règles

Chaque règle s'exprime sous la forme ***`conditions -j action options_de_l'action`***. En écrivant bout à bout plusieurs conditions dans la même règle, on en produit la conjonction (elles sont liées par des *et* logiques), donc une condition plus restrictive.

La condition **`-p protocole`** sélectionne selon le champ protocole du paquet IP, dont les valeurs les plus courantes sont `tcp`, `udp`, et `icmp`. Préfixer le protocole par un point d'exclamation inverse la condition (qui correspond alors à tous les paquets n'ayant pas le protocole indiqué). Cette manipulation est possible pour toutes les autres conditions énoncées ci-dessous.

La condition **`-s adresse`** ou **`-s réseau/masque`** vérifie l'adresse source du paquet ; **`-d adresse`** ou **`-d réseau/masque`** en est le pendant pour l'adresse de destination.

La condition **`-i interface`** sélectionne les paquets provenant de l'interface réseau indiquée ; **`-o interface`** sélectionne les paquets en fonction de leur interface réseau d'émission.

D'autres conditions plus spécifiques existent, qui dépendent des conditions génériques déjà définies. La condition **`-p tcp`** peut par exemple être accompagnée de conditions sur les ports TCP avec **`--source-port port`** et **`--destination-port port`**.

L'option **`--state` *`état`*** indique le statut du paquet dans une connexion (le module **`ipt_conntrack`**, qui implémente le suivi des connexions, lui est nécessaire). L'état `NEW` désigne un paquet qui débute une nouvelle connexion. L'état `ESTABLISHED` concerne les paquets d'une connexion existante et l'état `RELATED` les paquets d'une nouvelle connexion liée à une connexion existante (c'est le cas des connexions `ftp-data` d'une session `ftp`).

La section précédente détaille la liste des actions possibles, mais pas les options qui leur sont associées. L'action `LOG` dispose ainsi de plusieurs options visant à :

- indiquer la priorité du message à **`syslog`** (**`--log-priority`**, de valeur par défaut `warning`);
- préciser un préfixe textuel pour différencier les messages (**`--log-prefix`**);
- et indiquer les données à intégrer dans le message (**`--log-tcp-sequence`** pour le numéro de séquence TCP, **`--log-tcp-options`** pour les options TCP et **`--log-ip-options`** pour les options IP).

L'action *DNAT* dispose de l'option **`--to-destination` *`adresse`*`:`*`port`*** pour indiquer la nouvelle adresse IP et/ou le nouveau port de destination. De la même manière, l'action *SNAT* dispose de l'option **`--to-source` *`adresse`*`:`*`port`*** pour indiquer la nouvelle adresse et/ou le nouveau port source.

L'action *REDIRECT* dispose de l'option **`--to-ports` *`port(s)`*** pour indiquer le port ou l'intervalle de ports vers lesquels rediriger les paquets.

Créer les règles

Il faut invoquer **`iptables`** une fois par règle à créer ; c'est pourquoi on consigne habituellement tous les appels à cette commande dans un fichier de script pour mettre en place la même configuration à chaque redémarrage de la machine. On peut écrire ce script à la main mais il est souvent intéressant de le préparer à l'aide d'un outil de plus haut niveau, tel que **`fwbuilder`**.

Son principe est simple. Dans une première étape, il faut décrire tous les éléments susceptibles d'intervenir dans les différentes règles :

- le pare-feu et ses interfaces réseau ;
- les réseaux (et plages d'IP associées) ;
- les serveurs ;
- les ports correspondant aux services hébergés sur les différents serveurs.

On crée ensuite les règles par simple glisser/déposer des différents objets, quelques menus contextuels permettant de modifier la condition (l'inverser, par exemple) Il ne reste qu'à saisir l'action souhaitée et à la paramétrer.

`fwbuilder` peut alors générer un script de configuration du pare-feu selon les règles saisies. Son architecture modulaire lui permet de générer des scripts pour les différents pare-feu existants (**`iptables`** pour Linux 2.4/2.6, **`ipchains`** pour Linux 2.2, **`ipf`** pour FreeBSD et **`pf`** pour OpenBSD).

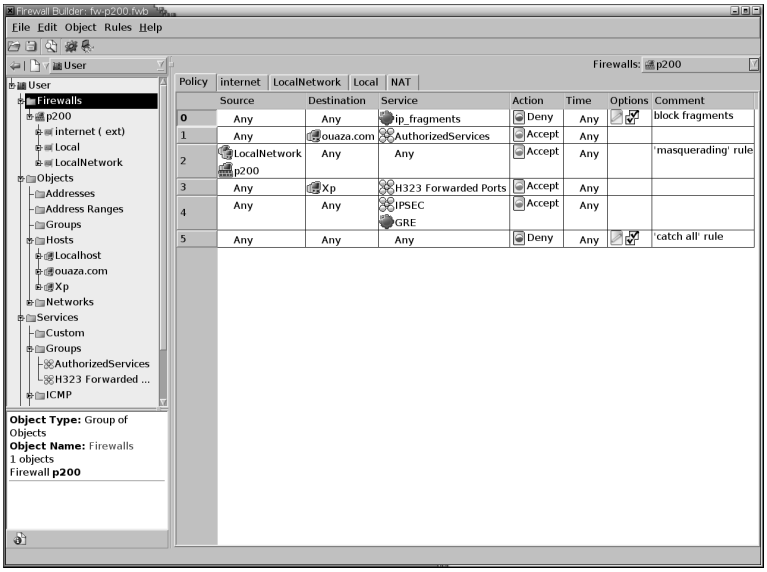


Figure 10–2 Fwbuilder en action

Le paquet *fwbuilder* contient l'interface graphique tandis que *fwbuilder-linux* contient les modules pour les pare-feu Linux (et notamment celui dédié à **`iptables`**, que l'on souhaite employer pour configurer notre pare-feu *netfilter*) :

```
# apt-get install fwbuilder fwbuilder-linux
```

Installer les règles à chaque démarrage

Si le pare-feu doit protéger une connexion réseau intermittente par PPP, le plus simple est de changer le nom du script de configuration du pare-feu et de l'installer sous /etc/ppp/ip-up.d/0iptables (attention, le nom de fichier ne doit pas contenir de point, sinon il ne sera pas pris en compte). Ainsi, il sera rechargé à chaque démarrage d'une connexion PPP.

Dans les autres cas, le plus simple est d'inscrire le script de configuration du pare-feu dans une directive `up` du fichier /etc/network/interfaces. Dans l'exemple ci-dessous, ce script s'appelle /usr/local/etc/arrakis.fw.

EXEMPLE **Fichier `interfaces` avec appel du script de pare-feu**

```
auto eth0
iface eth0 inet static
    address 192.168.0.1
    network 192.168.0.0
    netmask 255.255.255.0
    broadcast 192.168.0.255
    up /usr/local/etc/arrakis.fw
```

Réseau privé virtuel

Un réseau privé virtuel (*Virtual Private Network*, ou VPN) est un moyen de relier par l'Internet deux réseaux locaux distants via un tunnel (généralement chiffré pour des raisons de confidentialité). Souvent, cette technique sert simplement à intégrer une machine distante au sein du réseau local de l'entreprise.

Il y a plusieurs manières d'obtenir ce résultat. L'une est d'employer SSH pour le tunnel chiffré et PPP pour effectuer le lien entre les deux réseaux. Une autre méthode repose sur le protocole IPsec, qui permet de chiffrer les communications IP de manière transparente entre deux hôtes. Il est encore possible de faire appel au protocole PPTP de Microsoft. Ce livre fera l'impasse sur les autres types de VPN existants.

SSH et PPP

Cette méthode fonctionnelle est très simple à mettre en œuvre. En revanche, ce n'est pas la plus efficace : elle n'est pas adaptée aux gros débits sur le réseau privé virtuel. Pour la mettre en place, suivez les instructions du HOWTO correspondant :

▸ http://www.tldp.org/HOWTO/ppp-ssh/

Concrètement, en utilisant le protocole PPP sur un tunnel SSH, on emploie deux fois le protocole TCP (aux niveaux de PPP et de SSH du SSH). Ce double emploi — dû à l'encapsulation d'une pile de protocole TCP/IP (PPP) dans une connexion TCP/IP (SSH) — pose quelques problèmes, notamment à cause de la capacité de TCP à s'adapter aux conditions du réseau en variant les délais de *timeout* (délai maximal d'attente). Le site suivant détaille ces problèmes :

▸ http://sites.inka.de/sites/bigred/devel/tcp-tcp.html

IPsec

Plusieurs logiciels permettent de gérer IPsec : citons **`freeswan`**, **`openswan`** et **`racoon`**. Le premier, sans doute le plus connu, n'est malheureusement plus développé ; désormais, **`openswan`** le remplace avantageusement. D'ailleurs Debian va probablement supprimer **`freeswan`** en considérant que le paquet **`openswan`** le supplante.

IPsec est le standard en matière de réseau privé virtuel, mais cette solution est nettement plus difficile à mettre en œuvre. Les noyaux Debian intégrant en standard sa prise en charge, il n'est plus nécessaire de recompiler un noyau spécifique. **`openswan`** apporte seulement les outils utilisateur nécessaires à IPsec, notamment le démon IKE (*IPsec Key Exchange*, ou échange de clés IPsec) qui permet d'échanger des clés cryptographiques entre deux hôtes.

L'installation du paquet simplifie le travail en créant un certificat X.509 ou un couple de clés RSA.

Reste ensuite à configurer le fichier /etc/ipsec.conf pour y paramétrer les « tunnels *IPsec* » (ou *Security Association* dans le vocabulaire *IPsec*) à créer. On peut reprendre l'exemple qu'il donne, mais il est indispensable d'avoir lu au préalable la documentation /usr/share/doc/openswan/doc/config.html pour comprendre le sens de chaque directive.

SÉCURITÉ **IPsec et pare-feu**

Le fonctionnement d'*IPsec* induit des échanges de données sur le port UDP 500 pour les échanges de clés (et aussi sur le port UDP 4500 si NAT-T est employé). De plus, les paquets *IPsec* utilisent deux protocoles IP dédiés que le pare-feu doit aussi laisser passer : les protocoles 50 (ESP) et 51 (AH).

ATTENTION **IPsec et NAT**

IPsec cohabite difficilement avec NAT sur un pare-feu. En effet, *IPsec* signant les paquets, toute modification de ceux-ci à la volée invalidera leur signature et les fera refuser. Les différentes implémentations d'*IPsec* proposent désormais la technique *NAT-T* (*NAT Traversal*, ou traversée de NAT), qui consiste à encapsuler le paquet *IPsec* dans un paquet UDP.

PPTP

PPTP (*Point-to-Point Tunneling Protocol*, ou protocole de tunnel en point à point) emploie deux canaux de communication, pour échanger respectivement des informations de contrôle et des données (ces dernières emploient le protocole GRE — *Generic Routing Encapsulation*, ou encapsulation de routage générique). Une connexion PPP standard s'établit sur le canal d'échange de données.

Configuration du client

Le paquet *pptp-linux*, facile à configurer, est le client PPTP pour Linux. Les instructions suivantes sont inspirées de sa documentation officielle :

▸ http://pptpclient.sourceforge.net/howto-debian.phtml

Les administrateurs de Falcot ont créé plusieurs fichiers : /etc/ppp/options.pptp, /etc/ppp/peers/falcot, /etc/ppp/ip-up.d/falcot, et /etc/ppp/ip-down.d/falcot.

EXEMPLE **Fichier /etc/ppp/options.pptp**

```
# Options PPP employées pour une connexion PPTP
lock
noauth
nobsdcomp
nodeflate
```

EXEMPLE **Fichier /etc/ppp/peers/falcot**

```
# vpn.falcot.com est le serveur PPTP
pty "pptp vpn.falcot.com --nolaunchpppd"
# la connexion s'identifiera comme utilisateur «vpn»
user vpn
remotename pptp
# la prise en charge du chiffrement est nécessaire
require-mppe-128
file /etc/ppp/options.pptp
ipparam falcot
```

EXEMPLE **Fichier /etc/ppp/ip-up.d/falcot**

```
# Créer la route vers le réseau local de Falcot
if [ "$6" = "falcot" ]; then
  # 192.168.0.0/24 est le réseau distant chez Falcot
  route add -net 192.168.0.0 netmask 255.255.255.0 dev $1
fi
```

EXEMPLE **Fichier /etc/ppp/ip-down.d/falcot**

```
# Supprimer la route vers le réseau local de Falcot
if [ "$6" = "falcot" ]; then
  # 192.168.0.0/24 est le réseau distant chez Falcot
  route del -net 192.168.0.0 netmask 255.255.255.0 dev $1
fi
```

SÉCURITÉ **MPPE**

La sécurisation de PPTP recourt à MPPE (*Microsoft Point-to-Point Encryption*, ou chiffrement point à point de Microsoft), malheureusement pas pris en charge par les noyaux Debian standards. Il faut donc compiler un noyau spécifique, incluant le patch du paquet Debian *kernel-patch-mppe*. Les instructions sont données dans le document suivant :

▸ http://pptpclient.sourceforge.net/howto-debian-build.phtml

On peut aussi faire appel au noyau préparé et proposé par le contributeur Darik Horn. Attention ! ce paquet n'étant pas officiel, son fonctionnement et son contenu ne sont aucunement garantis.

▸ http://pptpclient.sourceforge.net/howto-debian-prepackaged.phtml

Configuration du serveur

ATTENTION **PPTP et pare-feu**

Les pare-feu intermédiaires doivent autoriser les paquets IP employant le protocole 47 (GRE). De plus, le port 1723 du serveur PPTP doit être ouvert pour qu'une communication puisse prendre place.

pptpd est le serveur PPTP pour Linux. Son fichier de configuration principal /etc/pptpd.conf n'a presque pas besoin de modifications ; il faut juste y renseigner *localip* (adresse IP locale) et *remoteip* (adresse IP distante). Dans le fichier ci-dessous, le serveur PPTP a toujours l'adresse IP `192.168.0.199` et les clients PPTP reçoivent des adresses IP comprises entre `192.168.0.200` et `192.168.0.250`.

EXEMPLE **Fichier /etc/pptpd.conf**

```
# TAG: speed
#
#       Specifies the speed for the PPP daemon to talk at.
#
speed 115200

# TAG: option
#
#       Specifies the location of the PPP options file.
#       By default PPP looks in '/etc/ppp/options'
#
option /etc/ppp/pptpd-options

# TAG: debug
#
#       Turns on (more) debugging to syslog
#
# debug
```

```
# TAG: localip
# TAG: remoteip
#
#       Specifies the local and remote IP address ranges.
#
#       You can specify single IP addresses seperated by commas or you
  can
#       specify ranges, or both. For example:
#
#               192.168.0.234,192.168.0.245-249,192.168.0.254
#
#       IMPORTANT RESTRICTIONS:
#
#       1. No spaces are permitted between commas or within addresses.
#
#       2. If you give more IP addresses than MAX_CONNECTIONS, it will
#          start at the beginning of the list and go until it gets
#          MAX_CONNECTIONS IPs. Others will be ignored.
#
#       3. No shortcuts in ranges! ie. 234-8 does not mean 234 to 238,
#          you must type 234-238 if you mean this.
#
#       4. If you give a single localIP, that's ok - all local IPs will
#          be set to the given one. You MUST still give at least one
  remote
#          IP for each simultaneous client.
#
#localip 192.168.0.234-238,192.168.0.245
#remoteip 192.168.1.234-238,192.168.1.245
#localip 10.0.1.1
#remoteip 10.0.1.2-100
localip 192.168.0.199
remoteip 192.168.0.200-250
```

Il faut aussi modifier la configuration PPP employée par le serveur PPTP, consignée dans le fichier /etc/ppp/pptpd-options. Les paramètres importants à changer sont les noms du serveur (`pptp`), du domaine (`falcot.com`), et les adresses IP des serveurs DNS et Wins.

EXEMPLE Fichier /etc/ppp/pptpd-options

```
## turn pppd syslog debugging on
#debug

## change 'servername' to whatever you specify as your server name in
  chap-secrets
name pptp
## change the domainname to your local domain
domain falcot.com

## these are reasonable defaults for WinXXXX clients
## for the security related settings
# The Debian pppd package now supports both MSCHAP and MPPE, so enable
  them
# here. Please note that the kernel support for MPPE must also be present
  !
auth
require-chap
require-mschap
require-mschap-v2
require-mppe-128

## Fill in your addresses
ms-dns 192.168.0.1
ms-wins 192.168.0.1
```

```
## Fill in your netmask
netmask 255.255.255.0

## some defaults
nodefaultroute
proxyarp
lock
```

La dernière étape est d'enregistrer l'utilisateur `vpn` et le mot de passe associé dans le fichier /etc/ppp/chap-secrets. Le nom du serveur doit y être renseigné explicitement, l'astérisque (*) habituelle ne fonctionnant pas. Par ailleurs, il faut savoir que les clients PPTP sous Windows s'identifient sous la forme *DOMAINE**UTILISATEUR* au lieu de se contenter du nom d'utilisateur. C'est pourquoi on trouve aussi dans ce fichier l'utilisateur FALCOT\\vpn. On peut encore y spécifier individuellement les adresses IP des utilisateurs, ou indiquer un astérisque dans ce champ si l'on ne souhaite pas d'adresses fixes.

SÉCURITÉ **Failles de PPTP**

La première implémentation par Microsoft de PPTP fut sévèrement critiquée car elle souffrait de nombreuses failles de sécurité, pour la plupart corrigées dans la dernière version du protocole. C'est cette dernière version qui est employée par la configuration documentée dans cette section. Attention cependant, car la suppression de certaines options (notamment `require-mppe-128` et `require-mschap-v2`) rendrait le service à nouveau vulnérable.

EXEMPLE **Fichier /etc/ppp/chap-secrets**

```
# Secrets for authentication using CHAP
# client        server  secret      IP addresses
vpn             pptp    f@Lc3au     *
FALCOT\\vpn     pptp    f@Lc3au     *
```

Qualité de service

Principe et fonctionnement

Le terme QoS (*Quality of Service*) désigne l'ensemble des techniques permettant de garantir ou d'améliorer sensiblement la qualité de service apportée à des applications. La plus populaire consiste à traiter différemment chaque type de trafic réseau ; son application principale est le *shaping*. Cela permet de limiter les débits attribués à certains services et/ou à certaines machines, notamment pour ne pas saturer la bande passante. Cette technique s'adapte bien au flux TCP car ce protocole s'adapte automatiquement au débit disponible.

CULTURE **Les files d'attente**

Les différents modèles de gestion de files d'attente et leurs intérêts respectifs sont présentés dans cette étude :

▸ http://www.supinfo-projects.com/fr/2004/linux%5Fqos/

CULTURE **LARTC — Linux Advanced Routing & Trafic Control**

Le HOWTO du routage avancé et du contrôle de trafic sous Linux est un document de référence qui couvre de manière assez exhaustive tout ce qui concerne la qualité de service.

▸ http://www.linux-france.org/prj/inetdoc/guides/lartc/lartc.html

On peut encore modifier les priorités du trafic, ce qui permet généralement de traiter d'abord les paquets relatifs à des services interactifs (**`ssh`**, **`telnet`**) ou à des services échangeant de petits blocs de données.

Les noyaux Debian intègrent le QoS et toute la panoplie des modules associés. Ils sont nombreux, et chacun offre un service différent — notamment par le biais de files d'attente pour les paquets IP (*scheduler*, ou ordonnanceur), dont les mécanismes variés couvrent tout le spectre des besoins possibles.

Configuration et mise en œuvre

Le QoS se paramètre avec le logiciel **`tc`**, du paquet Debian *iproute*. Son interface étant extrêmement complexe, il est préférable d'employer des outils de plus haut niveau.

Minimiser le temps de latence : wondershaper

L'objectif de **wondershaper** (du paquet Debian éponyme) est de minimiser les temps de latence quelle que soit la charge réseau. Il l'atteint en limitant le trafic total juste en deçà de la valeur de saturation de la ligne.

Après la configuration d'une interface réseau, il est possible de mettre en place ce contrôle du trafic par la commande **wondershaper *interface débit_descendant débit_montant***. L'*interface* sera par exemple *eth0* ou *ppp0* et les deux débits (descendant et montant) s'expriment en kilobits par seconde. La commande **wondershaper remove *interface*** désactive le contrôle du trafic sur l'interface indiquée.

Pour une connexion Ethernet, le plus simple est d'appeler automatiquement ce script après la configuration de l'interface en modifiant le fichier /etc/network/interfaces pour y ajouter des directives up (indiquant une commande à exécuter après configuration de l'interface) et down (indiquant une commande à exécuter après déconfiguration de l'interface) comme suit :

EXEMPLE **Modification du fichier /etc/network/interfaces**

```
iface eth0 inet dhcp
    up /sbin/wondershaper eth0 500 100
    down /sbin/wondershaper remove eth0
```

Dans le cas de PPP, le dépôt d'un script dans /etc/ppp/ip-up.d activera le contrôle de trafic dès le démarrage de la connexion.

POUR ALLER PLUS LOIN **Configuration optimale**

Le fichier /usr/share/doc/wondershaper/README.Debian.gz détaille la méthode de configuration recommandée par le mainteneur du paquet. Il conseille d'effectuer des mesures de vitesses de téléchargement pour mieux évaluer les limites réelles.

Définir des priorités et des limites : shaper

Ce script (issu du paquet éponyme) définit des priorités et des limites de trafic par « classe ». Chaque classe correspond à un type de trafic, identifié par une combinaison de différents critères tels que la source, la destination, l'horaire, etc.

Les différentes classes sont décrites par des fichiers du répertoire /etc/shaper, dont le format est détaillé dans la documentation /usr/share/doc/shaper/README.shaper.gz.

Le paquet installe un script de démarrage qui applique automatiquement les règles de contrôle du trafic.

Configuration standard

En l'absence d'une configuration particulière de QoS, le noyau Linux emploie la file d'attente `pfifo_fast`, qui propose déjà quelques fonctionnalités intéressantes. Pour établir les priorités des paquets IP, elle utilise leur champ ToS (*Type of Service*, ou type de service) — qu'il suffira donc de modifier pour bénéficier de cette file. Ce champ peut recevoir cinq valeurs :

- *Normal-Service* (0) (service normal) ;
- *Minimize-Cost* (2) (minimiser le coût) ;
- *Maximize-Reliability* (4) (maximiser la fiabilité) ;
- *Maximize-Throughput* (8) (maximiser le débit) ;
- *Minimize-Delay* (16) (minimiser le délai).

Le champ ToS peut être positionné par les applications qui génèrent des paquets IP ou modifié à la volée par *netfilter*. Avec la règle ci-dessous, il est ainsi possible d'améliorer l'interactivité du service SSH d'un serveur :

```
iptables -t mangle -A PREROUTING -p tcp --sport ssh -j TOS --set-tos
   Minimize-Delay
iptables -t mangle -A PREROUTING -p tcp --dport ssh -j TOS --set-tos
   Minimize-Delay
```

Routage dynamique

B.A.-BA **Routage dynamique**

Le routage dynamique permet aux routeurs d'ajuster en temps réel les chemins employés pour faire circuler les paquets IP. Chaque protocole a sa propre méthode de définition des routes (calcul du chemin le plus court, récupération des routes annoncées par les partenaires, etc.).
Pour le noyau Linux, une route associe un périphérique réseau à un ensemble de machines qu'il peut atteindre. La commande **`route`** permet de les définir et de les consulter.

Le logiciel **`quagga`** (du paquet Debian éponyme) est désormais la référence en matière de routage dynamique (il a supplanté **`zebra`**, dont le développement s'est arrêté). Cependant, pour des raisons de compatibilité, le projet **`quagga`** a conservé les noms des exécutables : c'est pourquoi on retrouve le programme **`zebra`** plus loin.

C'est un ensemble de démons qui coopèrent pour définir les tables de routage employées par le noyau Linux, chaque protocole de routage (notamment BGP, OSPF, RIP) disposant de son propre démon. Le démon **`zebra`** centralise les informations reçues des autres démons et gère les tables de routage statiques. Les autres démons s'appellent **`bgpd`**, **`ospfd`**, **`ospf6d`**, **`ripd`**, et **`ripngd`**.

On active un démon en modifiant le fichier /etc/quagga/daemons et en créant dans le répertoire /etc/quagga son fichier de configuration, qui doit porter son nom suivi de .conf et appartenir à l'utilisateur `quagga` et au groupe `quaggavty` — dans le cas contraire, le script /etc/init.d/quagga n'invoquera pas ce démon.

La configuration de chacun de ces démons impose de connaître le fonctionnement du protocole de routage concerné. Il n'est pas possible de tous les détailler ici, mais sachez que le manuel au format *info* du paquet *quagga-doc* n'est pas avare d'explications. Par ailleurs, la syntaxe est très similaire à l'interface de configuration d'un routeur traditionnel et un administrateur réseau s'adaptera rapidement

à **quagga**. Par souci d'ergonomie, il est aussi possible de consulter ce manuel au format HTML à l'adresse suivante :

- http://www.quagga.net/docs/docs-info.php

CULTURE **Articles traitant de Quagga**

Pacôme Massol a rédigé quelques articles pour Linux Magazine France traitant de Zebra (le prédécesseur de Quagga, dont il partage la syntaxe). Ces textes vous fourniront donc des éléments sur ce logiciel.

- http://perso.wanadoo.fr/pmassol/main.html
- http://perso.wanadoo.fr/pmassol/lm/zebrastat.html
- http://perso.wanadoo.fr/pmassol/lm/zebraospf.html
- http://perso.wanadoo.fr/pmassol/lm/zebrarip.html

EN PRATIQUE **OPSF, BGP ou RIP ?**

OSPF est probablement le protocole à privilégier pour du routage dynamique sur des réseaux privés, mais c'est BGP qu'on trouve majoritairement sur l'Internet. RIP est un ancêtre désormais assez peu utilisé.

IPv6

IPv6 — successeur d'IPv4 — est une nouvelle version du protocole IP, qui doit en corriger les défauts et notamment le nombre trop faible d'adresses IP existantes. Ce protocole gère la couche réseau, c'est-à-dire qu'il offre la possibilité d'adresser les machines et de fragmenter les données.

Les noyaux Debian disposent de modules pour la prise en charge d'IPv6 (les commandes ci-dessous ne fonctionneront qu'après chargement du module **ipv6**). Pour exploiter au mieux IPv6, on optera pour le noyau 2.6. Les outils de base comme **ping** et **traceroute** ont pour équivalents IPv6 **ping6** et **traceroute6**, respectivement disponibles dans les paquets Debian *iputils-ping* et *iputils-tracepath*.

On peut configurer le réseau IPv6 comme un réseau IPv4, à travers le fichier /etc/network/interfaces. Pour ne pas se contenter d'un réseau IPv6 privé, il faut cependant disposer d'un routeur capable de relayer le trafic sur le *6bone* — le réseau public expérimental employant IPv6.

EXEMPLE **Exemple de configuration IPv6**

```
iface eth0 inet6 static
    address 3ffe:ffff:1234:5::1:1
    netmask 64
    # Pour désactiver l'auto-configuration:
    # up echo 0 >/proc/sys/net/ipv6/conf/all/autoconf
    # Le routeur est auto-configuré, et n'a pas d'adresse fixe.
    # (/proc/sys/net/ipv6/conf/all/accept_ra). Sinon pour le forcer:
    # gateway 3ffe:ffff:1234:5::1
```

En l'absence d'un point de connexion au *6bone*, on peut toujours s'y connecter via un tunnel sur IPv4. Freenet6 est un fournisseur gratuit de tels tunnels :

- http://www.freenet6.net

Pour exploiter cette possibilité, il faut s'inscrire sur ce site web puis installer le paquet Debian *freenet6* et configurer ce tunnel. On intégrera au fichier /etc/freenet6/tspc.conf les lignes `userid` et `password` reçues par courrier électronique.

On proposera une connectivité IPv6 à toutes les machines du réseau local en modifiant dans le fichier /etc/freenet6/tspc.conf les trois directives ci-dessous (le réseau local est supposé connecté à l'interface *eth0*) :

```
host_type=router
prefix_len=48
if_prefix=eth0
```

La machine est alors le routeur d'accès à un sous-réseau dont le préfixe fait 48 bits. Le tunnel désormais averti, il faut encore informer le réseau local de cette caractéristique en installant le démon *radvd* (du paquet éponyme). C'est un démon de configuration IPv6 jouant le même rôle que **dhcpd** pour le monde IPv4.

Il faut ensuite créer son fichier de configuration /etc/radvd.conf (par exemple en adaptant le fichier /usr/share/doc/radvd/examples/simple-radvd.conf). En l'occurrence, le seul changement nécessaire est le préfixe, qu'il faut remplacer par celui fourni par Freenet6 (que l'on retrouvera dans la sortie de la commande **ifconfig** dans le bloc relatif à l'interface *sit1*).

Après les commandes **/etc/init.d/freenet6 restart** et **/etc/init.d/radvd start**, le réseau IPv6 sera enfin fonctionnel.

ATTENTION **IPv6 et pare-feu**

Le bon fonctionnement de l'IPv6 natif (par opposition à un tunnel sur IPv4) exige que le pare-feu laisse passer son trafic, qui emploie le protocole IP numéro 41. C'est possible : il existe une adaptation de **netfilter** pour l'IPv6 compilée dans les noyaux Debian. Elle se configure comme la version classique, mais avec le programme **ip6tables**.

ASTUCE **Programmes compilés avec IPv6**

De nombreux logiciels doivent être adaptés pour la prise en charge d'IPv6. Certains logiciels de Debian en sont capables en standard, mais pas encore tous. Heureusement, quelques volontaires ont créé une archive de paquets regroupant des logiciels spécialement recompilés pour IPv6.

▸ http://debian.fabbione.net

Serveur de noms (DNS)

Principe et fonctionnement

Le service de gestion des noms (*Domain Name Service*) est fondamental sur l'Internet : il permet d'associer des noms à des adresses IP (et vice versa), ce qui permet de saisir *www.yahoo.fr* en lieu et place de *217.12.3.11*.

Les informations DNS sont regroupées par zones, correspondant chacune à un domaine ou à une plage d'adresses IP. Un serveur primaire fait autorité sur le contenu d'une zone ; un serveur secondaire, normalement hébergé sur une autre machine, se contente de proposer une copie de la zone primaire, qu'il met à jour régulièrement.

Chaque zone peut contenir différents types d'enregistrements (*Resource Records*) :

- *A* : attribution d'une adresse IPv4.
- *CNAME* : définition d'un alias.

- *MX* : définition d'un serveur de courrier électronique, information exploitée par les serveurs de messagerie pour retrouver le serveur correspondant l'adresse de destination d'un courrier électronique. Chaque enregistrement *MX* a une priorité associée. Le serveur de plus haute priorité (portant le nombre le plus petit) recevra les connexions SMTP. S'il ne répond pas, le deuxième serveur sera contacté, etc.
- *PTR* : correspondance adresse IP vers nom. Elle est stockée dans une zone dédiée à la résolution inverse, nommée en fonction de la plage d'adresses IP : par exemple `1.168.192.in-addr.arpa` pour toutes les adresses du réseau `192.168.1.0/24`.
- *AAAA* : correspondance nom vers adresse IPv6.
- *NS* : correspondance nom vers serveur de noms.

  Chaque domaine doit compter au moins un enregistrement NS. Tous ces enregistrements pointent sur un serveur DNS capable de répondre aux requêtes portant sur ce domaine ; ils signaleront les serveurs primaires et secondaires du domaine concerné. Ces enregistrements permettent aussi de mettre en place une délégation DNS. On pourra ainsi indiquer que le domaine `interne.falcot.com` est géré par un autre serveur de noms et déléguer ainsi une partie du service. Évidemment, le serveur concerné devra déclarer une zone `interne.falcot.com`.

Le logiciel serveur de nom de référence, Bind, est développé par l'ISC (*Internet Software Consortium*, ou consortium du logiciel Internet). Debian en propose deux : le paquet *bind* correspond à la version 8.x du serveur tandis que le paquet *bind9* abrite sa version 9.x. Cette dernière apporte deux nouveautés majeures. Il est désormais possible d'employer le serveur DNS sous une identité utilisateur non privilégiée de sorte qu'une faille de sécurité ne donne pas systématiquement les droits de root à l'attaquant, comme cela a souvent été le cas avec la version 8.x.

D'autre part, elle prend en charge *DNSSEC*, norme qui permet de signer et donc d'authentifier les enregistrements DNS, empêchant ainsi toute falsification de ces données par des intermédiaires mal intentionnés.

CULTURE **DNSSEC**

La norme DNSSEC est assez complexe ; pour en comprendre tous les tenants et aboutissants, je vous suggère de consulter les informations disponibles sur le site du NIC France (organisme gérant l'attribution des domaines en « `.fr` »), et particulièrement les supports de cours. Il faut savoir que cette norme, encore relativement expérimentale, n'est pas systématiquement employée (même si elle coexiste parfaitement avec des serveurs DNS qui ne la connaissent pas).

▸ http://www.afnic.fr/afnic/r_d/dnssec

Configuration

Quelle que soit la version de **`bind`** employée, le module *webmin-bind* de l'outil Webmin conviendra puisque les fichiers de configuration ont la même structure.

ATTENTION **Noms des zones inverses**

Une zone inverse porte un nom particulier. La zone couvrant le réseau `192.168.0.0/16` s'appellera ainsi `168.192.in-addr.arpa` : les composants de l'adresse IP sont inversés et suivis du suffixe *in-addr.arpa*.
Webmin effectuant automatiquement cette conversion, il faudra employer le nom `192.168`.

ATTENTION **Syntaxe d'un nom**

La syntaxe désignant les noms de machine est particulière. `machine` sous-entend ainsi `machine.`*`domaine`*. S'il ne faut pas ajouter automatiquement le nom du domaine, il convient d'écrire `machine.` (en suffixant ce nom d'un point). Pour indiquer un nom DNS extérieur au domaine géré, on écrira donc `machine.autredomaine.com.` avec un point.

Les administrateurs de Falcot ont créé une zone primaire `falcot.com` pour stocker les informations relatives à ce domaine et une zone `168.192.in-addr.arpa` pour les résolutions inverses des adresses IP des différents réseaux locaux.

ASTUCE **Tester le serveur DNS**

La commande **`host`** (du paquet *host* ou *bind9-host*) interroge un serveur DNS, par exemple celui qu'on vient de configurer. La commande **`host machine.falcot.com localhost`** contrôlera donc la réponse du serveur DNS local pour la requête `machine.falcot.com`. La commande **`host`** ***`adresseip`*** **`localhost`** testera la résolution inverse.

Ceux qui souhaitent configurer leur serveur DNS manuellement pourront s'inspirer des extraits suivants, issus des fichiers de configuration de la société Falcot.

EXEMPLE **Extrait du fichier `/etc/bind/named.conf.local`**

```
zone "falcot.com" {
        type master;
        file "/etc/bind/db.falcot.com";
        allow-query { any; };
        allow-transfer {
                195.20.105.149/32 ; // ns0.xname.org
                193.23.158.13/32 ; // ns1.xname.org
        };
};

zone "interne.falcot.com" {
        type master;
        file "/etc/bind/db.interne.falcot.com";
        allow-query { 192.168.0.0/16; };
};

zone "168.192.in-addr.arpa" {
        type master;
        file "/etc/bind/db.192.168";
        allow-query { 192.168.0.0/16; };
};
```

EXEMPLE **Extrait du fichier `/etc/bind/db.falcot.com`**

```
; Zone falcot.com
; admin.falcot.com. => contact pour la zone: admin@falcot.com
$TTL    604800
@       IN      SOA     falcot.com. admin.falcot.com. (
                        20040121        ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
; Le @ fait référence au nom de la zone («falcot.com.» en l'occurrence)
; ou à $ORIGIN si cette directive a été employée
;
@       IN      NS      ns
@       IN      NS      ns0.xname.org

interne IN      NS      192.168.0.2

@       IN      A       212.94.201.10
@       IN      MX      5 mail
@       IN      MX      10 mail2
```

```
ns      IN      A       212.94.201.10
mail    IN      A       212.94.201.10
mail2   IN      A       212.94.201.11
www     IN      A       212.94.201.11

dns     IN      CNAME   ns
```

EXEMPLE **Extrait du fichier /etc/bind/db.192.168**

```
; Zone inverse pour 192.168.0.0/16
; admin.falcot.com. => contact pour la zone: admin@falcot.com
$TTL    604800
@       IN      SOA     ns.interne.falcot.com. admin.falcot.com. (
                        20040121        ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL

        IN      NS      ns.interne.falcot.com.

; 192.168.0.1 -> arrakis
1.0     IN      PTR     arrakis.interne.falcot.com.
; 192.168.0.2 -> neptune
2.0     IN      PTR     neptune.interne.falcot.com.

; 192.168.3.1 -> pau
1.3     IN      PTR     pau.interne.falcot.com.
```

DHCP

Présentation

DHCP (*Dynamic Host Configuration Protocol*, ou protocole de configuration dynamique des hôtes) est un moyen de rapatrier automatiquement sa configuration pour une machine qui vient de démarrer et souhaite configurer son interface réseau. De cette manière, on peut centraliser la gestion des configurations réseau et toutes les machines bureautiques pourront recevoir des réglages identiques.

Un serveur DHCP fournit de nombreux paramètres réseau, et notamment une adresse IP et le réseau d'appartenance de la machine. Mais il peut aussi indiquer d'autres informations, telles que les serveurs DNS, WINS, NTP.

C'est encore l'*Internet Software Consortium* qui développe le serveur DHCP (avec *bind*). Le paquet Debian correspondant est *dhcp3-server*.

Configuration

Les premiers éléments à modifier dans le fichier de configuration du serveur DHCP, /etc/dhcp3/dhcpd.conf, sont le nom de domaine et les serveurs DNS. Il faut aussi activer (en la décommentant) l'option *authoritative* si ce serveur est le seul sur le réseau local (tel que défini par la limite de propagation du *broadcast*, mécanisme employé pour joindre le serveur DHCP). On créera aussi une section *subnet* décrivant le réseau local et les informations de configuration

diffusées. L'exemple ci-dessous convient pour le réseau local `192.168.0.0/24`, qui dispose d'un routeur (`192.168.0.1`) faisant office de passerelle externe. Les adresses IP disponibles sont comprises entre `192.168.0.128` et `192.168.0.254`.

EXEMPLE Extrait du fichier `/etc/dhcp3/dhcpd.conf`

```
#
# Sample configuration file for ISC dhcpd for Debian
#

# The ddns-updates-style parameter controls whether or not the server will
# attempt to do a DNS update when a lease is confirmed. We default to the
# behavior of the version 2 packages ('none', since DHCP v2 didn't
# have support for DDNS.)
ddns-update-style interim;

# option definitions common to all supported networks...
option domain-name "interne.falcot.com";
option domain-name-servers ns.interne.falcot.com;

default-lease-time 600;
max-lease-time 7200;

# If this DHCP server is the official DHCP server for the local
# network, the authoritative directive should be uncommented.
authoritative;

# Use this to send dhcp log messages to a different log file (you also
# have to hack syslog.conf to complete the redirection).
log-facility local7;

# My subnet
subnet 192.168.0.0 netmask 255.255.255.0 {
    option routers 192.168.0.1;
    option broadcast-address 192.168.0.255;
    range 192.168.0.128 192.168.0.254;
    ddns-domainname "interne.falcot.com";
}
```

DHCP et DNS

Une fonctionnalité appréciée est l'enregistrement automatique des clients DHCP dans la zone DNS de sorte que chaque machine ait un nom significatif (et pas automatique comme `machine-192-168-0-131.interne.falcot.com`). Pour exploiter cette possibilité, il faut autoriser le serveur DHCP à mettre à jour la zone DNS `interne.falcot.com` et configurer celui-ci pour qu'il s'en charge.

Dans le cas de **`bind`**, on ajoutera la directive *allow-update* aux deux zones que le serveur DHCP devra modifier (celle du domaine *interne.falcot.com* et celle de la résolution inverse). Cette directive donne la liste des adresses autorisées à effectuer la mise à jour ; on y consignera donc les adresses possibles du serveur DHCP (adresses IP locales et publiques le cas échéant).

```
allow-update { 127.0.0.1 192.168.0.1 212.94.201.10 !any };
```

Attention ! Une zone modifiable sera changée par *bind*, qui va donc réécrire régulièrement ses fichiers de configuration. Cette procédure automatique produisant des fichiers moins lisibles que les productions manuelles, les administrateurs de Falcot gèrent le sous-domaine *interne.falcot.com* à l'aide d'un serveur DNS délégué. Le fichier de la zone *falcot.com* reste ainsi entièrement sous leur contrôle.

L'exemple de fichier de configuration de serveur DHCP de la section précédente comporte déjà les directives nécessaires à l'activation de la mise à jour du DNS : il s'agit des lignes `ddns-update-style interim;` et `ddns-domain-name "interne.falcot.com";` dans le bloc décrivant le réseau.

Détection d'intrusion (IDS/NIDS)

snort (du paquet Debian éponyme) est un outil de détection d'intrusions : il écoute en permanence le réseau pour repérer les tentatives d'infiltration et/ou les actes malveillants (notamment les dénis de service). Tous ces événements sont enregistrés puis signalés quotidiennement à l'administrateur par un message électronique résumant les dernières 24 heures.

Son installation demande plusieurs informations. Il faut ainsi y préciser la plage d'adresses couvertes par le réseau local : il s'agit en réalité d'indiquer toutes les cibles potentielles d'attaques. On précisera également l'interface réseau à surveiller. Il s'agit en général d'*eth0* pour une connexion Ethernet, mais on pourra aussi trouver *ppp0* pour une connexion ADSL ou RTC (Réseau Téléphonique Commuté, ou modem classique).

Le fichier de configuration de **snort** (/etc/snort/snort.conf) est très long et ses abondants commentaires y détaillent le rôle de chaque directive. Il est fortement recommandé de le parcourir et de l'adapter à la situation locale pour en tirer le meilleur parti. En effet, il est possible d'y indiquer les machines hébergeant chaque service pour limiter le nombre d'incidents rapportés par **snort** (un déni de service sur une machine bureautique n'est pas aussi dramatique que sur un serveur DNS). On peut encore y renseigner les correspondances entre adresses IP et MAC (il s'agit d'un numéro unique identifiant chaque carte réseau) pour détecter les attaques par *ARP-spoofing* (travestissement d'ARP), qui permettent à une machine compromise de se substituer à une autre (un serveur sensible par exemple).

ATTENTION **Rayon d'action**

snort est limité par le trafic qu'il voit transiter sur son interface réseau : il ne pourra évidemment rien détecter s'il n'observe rien. Branché sur un commutateur (*switch*), il ne surveillera que les attaques ciblant la machine l'hébergeant, ce qui n'a qu'un intérêt assez limité. Pensez donc à relier la machine employant **snort** au port « miroir », qui permet habituellement de chaîner les commutateurs et sur lequel tout le trafic est dupliqué.
Pour un petit réseau doté d'un concentrateur (*hub*), le problème ne se pose pas : toutes les machines reçoivent tout le trafic.

B.A.-BA **Dénis de service**

Une attaque de type « déni de service » a pour seul objectif de rendre un service réseau inexploitable. Que cela soit en surchargeant le serveur de requêtes ou en exploitant un bogue de celui-ci, le résultat est toujours le même : le service en question n'est plus fonctionnel, les utilisateurs habituels sont mécontents et l'hébergeur du service réseau visé s'est fait une mauvaise publicité.

ALTERNATIVE **Un autre NIDS : prelude**

Le NIDS *snort* (*Network Intrusion Detection System*, ou système de détection d'intrusions sur le réseau) est très répandu, mais il compte depuis peu un concurrent moins éprouvé que lui : **prelude**, qui jouit d'une architecture plus modulaire. Un serveur (le *manager* du paquet *prelude-manager*) y centralise les alertes détectées par des capteurs (*sensors*) de plusieurs types. Le paquet *prelude-nids* propose un capteur qui, comme **snort**, surveille le trafic réseau. Le paquet *prelude-lml* (*Log Monitor Lackey*, ou laquais de surveillance de journaux système) surveille quant à lui les fichiers de *logs*, à l'instar de **logcheck**, déjà étudié.

11



Services réseau : Postfix, Apache, NFS, Samba, Squid, LDAP

Les services réseau sont les programmes interagissant directement avec les utilisateurs dans leur travail quotidien. C'est la partie émergée de l'iceberg « système d'information » présenté dans ce chapitre. La partie immergée, l'infrastructure, sur laquelle ils s'appuient, reste en arrière-plan.

SOMMAIRE

- Serveur de messagerie électronique
- Serveur web (HTTP)
- Serveur de fichiers NFS
- Partage Windows avec Samba
- Mandataire HTTP/FTP
- Annuaire LDAP

MOTS-CLEFS

- Postfix
- Apache
- NFS
- Samba
- Squid
- OpenLDAP

Serveur de messagerie électronique

Les administrateurs de Falcot SA ont retenu Postfix comme serveur de courrier électronique en raison de sa simplicité de configuration et de sa fiabilité. En effet, sa conception réduit au maximum les droits de chacune de ses sous-tâches, ce qui limite l'impact de toute faille de sécurité.

DÉCOUVERTE **Documentation en français**

Xavier Guimard a traduit la documentation officielle de Postfix. Je ne peux que vous encourager à la consulter pour découvrir encore plus en détail la configuration de ce serveur de courrier électronique.

▸ http://x.guimard.free.fr/postfix/

ALTERNATIVE **Le serveur Exim**

Debian emploie Exim comme serveur de messagerie par défaut (il est donc automatiquement installé pendant l'installation initiale). Son programme de configuration **`eximconfig`**, fourni dans le paquet, fonctionne presque comme celui de **`postfix`**, à la différence près qu'il n'exploite pas encore **`debconf`** pour interagir avec l'utilisateur.

Installation de Postfix

B.A.-BA **SMTP**

SMTP (*Simple Mail Transfer Protocol*) est le protocole employé par les serveurs de messagerie pour s'échanger les courriers électroniques.

Le paquet Debian *postfix* contient le démon SMTP principal. Divers modules (comme *postfix-ldap* ou *postfix-pgsql*) offrent des fonctionnalités supplémentaires à Postfix (notamment en termes d'accès à des bases de données de correspondances). Ne les installez que si vous en avez déjà perçu le besoin.

Au cours de l'installation du paquet, plusieurs questions sont posées par l'intermédiaire de *debconf*. Les réponses permettront de générer un premier fichier /etc/postfix/main.cf.

La première question porte sur le type d'installation. Parmi les choix proposés, seuls deux sont pertinents dans le cadre d'un serveur connecté à l'Internet. Il s'agit de « Site Internet » et d'« Internet par un FAI ». Le premier choix est adapté à un serveur qui reçoit et envoie le courrier directement à ses destinataires, mode retenu par les administrateurs de Falcot. Le second correspond à un serveur qui reçoit directement le courrier mais en envoie par l'intermédiaire d'un serveur SMTP intermédiaire — désigné par le terme *smarthost* — plutôt qu'en l'expédiant directement au serveur du destinataire. C'est surtout utile pour les particuliers disposant d'une adresse IP dynamique, parce que certains serveurs de messagerie refusent tout message provenant directement d'une telle adresse IP. Le *smarthost* sera ici le serveur SMTP du FAI, qui est toujours configuré pour transmettre le courrier provenant de ses clients.

La deuxième question porte sur le nom complet de la machine, employé pour générer une adresse de courrier électronique depuis un nom d'utilisateur local (c'est la partie suivant l'arobase « @ »). Pour Falcot, la réponse est `mail.falcot.com`.

Plus tard dans la procédure d'installation, l'ordinateur demande de saisir tous les noms de domaines associés à cette machine. La liste proposée inclut le nom complet de la machine et des synonymes de *localhost*, mais pas le domaine principal *falcot.com*, qu'il faut ajouter manuellement. D'une manière générale, il convient habituellement de donner ici tous les noms de domaines pour lesquels cette

machine fait office de serveur *MX* (c'est-à-dire tous ceux pour lesquels le DNS indique qu'elle est apte à accepter du courrier). Ces informations sont ensuite stockées dans la variable `mydestination` du fichier /etc/postfix/main.cf.

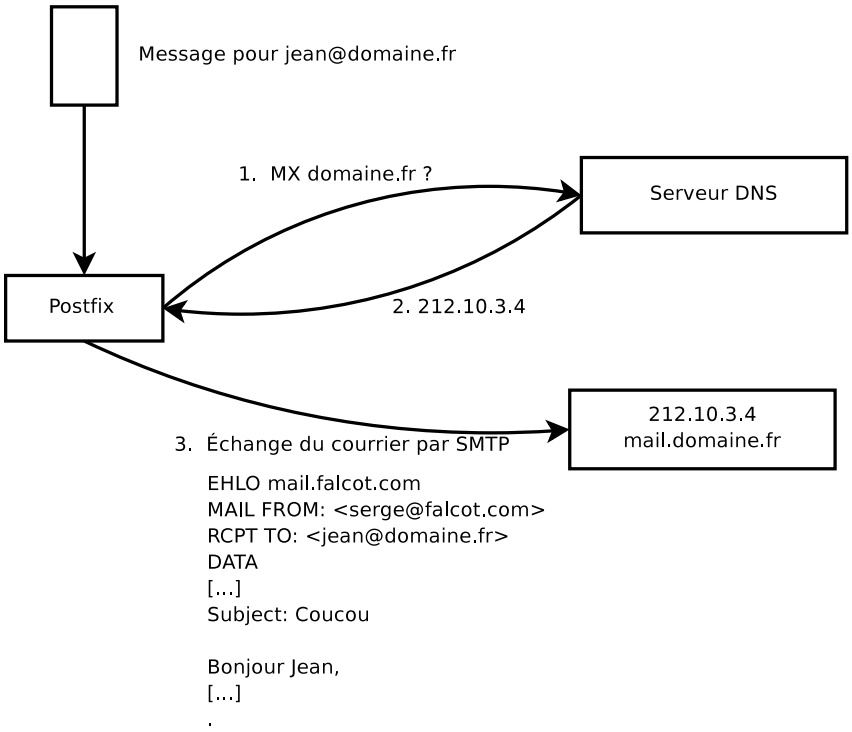


Figure 11–1 Rôle de l'enregistrement DNS MX dans un envoi de courrier électronique

Selon les cas, l'installation peut également demander d'indiquer les réseaux habilités à envoyer du courrier par l'intermédiaire de cette machine. Par défaut, Postfix est configuré pour n'accepter que des courriers électroniques issus de la machine elle-même ; il faut généralement ajouter le réseau local. Les administrateurs ont donc ajouté `192.168.0.0/16` à la réponse par défaut. Si la question n'est pas posée, il faut modifier le fichier de configuration et y changer la variable `mynetworks`.

L'emploi de **`procmail`** peut aussi être proposé pour délivrer le courrier localement. Cet outil permet aux utilisateurs de trier leur courrier entrant, ce pour quoi ils doivent indiquer des règles de tri dans leur fichier ~/.procmailrc.

Après cette première étape, les administrateurs ont obtenu le fichier de configuration ci-dessous. Il va servir de base pour les sections suivantes, qui le modifieront pour activer certaines fonctionnalités.

EXEMPLE **Fichier /etc/postfix/main.cf initial**

```
# See /usr/share/postfix/main.cf.dist for a commented, more complete
# version

smtpd_banner = $myhostname ESMTP $mail_name (Debian/GNU)
biff = no

# appending .domain is the MUA's job.
append_dot_mydomain = no

# Uncomment the next line to generate "delayed mail" warnings
#delay_warning_time = 4h

myhostname = localhost
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
myorigin = /etc/mailname
mydestination = mail.falcot.com, falcot.com, localhost, localhost.
   localdomain
relayhost =
mynetworks = 127.0.0.0/8 192.168.0.0/16
mailbox_command = procmail -a "$EXTENSION"
mailbox_size_limit = 0
recipient_delimiter = +
```

ATTENTION **Domaines virtuels et domaines canoniques**
Aucun des domaines virtuels ne doit être indiqué dans la variable `mydestination`. Celle-ci contient uniquement les noms des domaines « canoniques », directement associés à la machine et à ses utilisateurs locaux.

Configuration de domaines virtuels

Le serveur de messagerie peut recevoir le courrier pour d'autres domaines que le domaine « principal » ; on parle alors de « domaines virtuels ». Dans ces situations, il est rare que le courrier soit destiné aux utilisateurs locaux. Postfix offre deux fonctionnalités intéressantes pour gérer ces domaines virtuels.

Domaine virtuel d'alias

Un domaine virtuel d'alias ne contient que des alias, c'est-à-dire des adresses électroniques renvoyant le courrier vers d'autres adresses électroniques.

Pour activer un tel domaine, il faut préciser son nom dans la variable `virtual_alias_domains` et indiquer le fichier stockant les correspondances d'adresses dans la variable `virtual_alias_maps`.

EXEMPLE **Directives à ajouter au fichier /etc/postfix/main.cf**

```
virtual_alias_domains = marqueafalcot.tm.fr
virtual_alias_maps = hash:/etc/postfix/virtual
```

Le fichier /etc/postfix/virtual, décrivant les correspondances, emploie un format relativement simple. Chaque ligne contient deux champs séparés par une série de blancs, dont le premier est le nom de l'alias et le second une liste d'adresses électroniques vers lesquelles il pointe. La syntaxe spéciale « `@domaine.fr` » englobe tous les alias restants d'un domaine.

EXEMPLE **Exemple de fichier `/etc/postfix/virtual`**

```
webmaster@marqueafalcot.tm.fr  jean@falcot.com
contact@marqueafalcot.tm.fr    laure@falcot.com, sophie@falcot.com
# L'alias ci-dessous est générique, il englobe toutes les
# adresses électronique du domaine marqueafalcot.tm.fr
# non employées ailleurs dans ce fichier.
# Ces adresses sont renvoyées au même nom d'utilisateur
# mais dans le domaine falcot.com
@marqueafalcot.tm.fr           @falcot.com
```

Domaine virtuel de boîtes aux lettres

Les courriers destinés à un domaine virtuel de boîtes aux lettres sont stockés dans des boîtes aux lettres qui ne sont pas associées à un utilisateur local du système.

Pour activer un domaine virtuel de boîtes aux lettres, il faut l'écrire dans la variable `virtual_mailbox_domains` et préciser le fichier donnant les boîtes aux lettres avec la variable `virtual_mailbox_maps`. Le paramètre `virtual_mailbox_base` indique le répertoire sous lequel les différentes boîtes aux lettres seront stockées.

Les paramètres `virtual_uid_maps` et `virtual_gid_maps` définissent des tables de correspondances entre l'adresse électronique, l'utilisateur, et le groupe Unix propriétaire de la boîte aux lettres. Pour indiquer systématiquement le même propriétaire, la syntaxe `static:5000` dénote un UID/GID fixe.

ATTENTION **Domaine virtuel mixte ?**
Il n'est pas permis d'indiquer le même domaine dans les variables `virtual_alias_domains` et `virtual_mailbox_domains`. En revanche, tout domaine de `virtual_mailbox_domains` est implicitement compris dans `virtual_alias_domains`. Il est donc possible de mélanger alias et boîtes aux lettres au sein d'un domaine virtuel.

EXEMPLE **Directives à ajouter au fichier `/etc/postfix/main.cf`**

```
virtual_mailbox_domains = falco.org
virtual_mailbox_maps = hash:/etc/postfix/vmailbox
virtual_mailbox_base = /var/mail/vhosts
```

Le format du fichier /etc/postfix/vmailbox est à nouveau très simple (deux champs séparés par des blancs). Le premier indique une adresse électronique de l'un des domaines virtuels, et le second l'emplacement relatif de la boîte aux lettres associée (par rapport au répertoire donné par *virtual_mailbox_base*). Si le nom de la boîte aux lettres se termine par une barre de division (`/`), cette boîte sera stockée au format *maildir* ; dans le cas contraire, c'est le traditionnel *mbox* qui sera retenu. Le format *maildir* emploie un répertoire complet pour représenter la boîte aux lettres et chaque message est stocké dans un fichier. A contrario, une boîte aux lettres au format *mbox* est stockée dans un seul fichier, et chaque ligne débutant par « `From` » (« `From` » suivi d'une espace) marque le début d'un nouveau message électronique.

EXEMPLE **Fichier `/etc/postfix/vmailbox`**

```
# le courrier de jean est stocké au format maildir
# (1 fichier par courrier dans un répertoire privé)
jean@falco.org falco.org/jean/
# le courrier de sophie est stocké dans un fichier
# "mbox" traditionnel (tous les courriers concaténés
# dans un fichier)
sophie@falco.org falco.org/sophie
```

Restrictions à la réception et à l'envoi

Avec le nombre croissant de messages non sollicités (*spams*), il est nécessaire d'être de plus en plus strict sur les messages que le serveur accepte. Cette section présente les différentes stratégies intégrées à Postfix.

Restreindre l'accès en fonction de l'adresse IP

La directive `smtpd_client_restrictions` contrôle les machines autorisées à communiquer avec le serveur de courrier électronique.

EXEMPLE Restrictions en fonction de l'adresse du client

```
smtpd_client_restrictions = permit_mynetworks,
    warn_if_reject reject_unknown_client,
    check_client_access hash:/etc/postfix/access_clientip,
    reject_rbl_client sbl-xbl.spamhaus.org,
    reject_rbl_client list.dsbl.org
```

Lorsqu'une variable contient une liste de règles comme dans l'exemple ci-dessus, il faut savoir que celles-ci sont évaluées dans l'ordre, de la première à la dernière. Chaque règle peut accepter le message, le refuser ou le laisser poursuivre son chemin à travers celles qui suivent. L'ordre a donc une importance, et l'inversion de deux règles peut mettre en place un comportement très différent.

La directive `permit_mynetworks`, placée en tête de la liste des règles, accepte inconditionnellement toute machine du réseau local (tel que défini par la variable *mynetworks* dans la configuration).

La deuxième directive refuse normalement les machines dépourvues de configuration DNS totalement valide. Une configuration valide dispose d'une résolution inverse fonctionnelle et le nom DNS renvoyé pointe sur l'adresse IP employée. Cette restriction est généralement trop sévère, de nombreux serveurs de courrier électronique ne disposant pas de DNS inverse. C'est pourquoi les administrateurs de Falcot ont précédé la directive `reject_unknown_client` de `warn_if_reject`, qui transforme le refus en simple avertissement enregistré dans les logs. Ils peuvent ainsi surveiller le nombre de messages qui auraient été refusés et décider plus tard d'activer ou non cette règle en connaissant pleinement ses effets.

La troisième directive permet à l'administrateur de mettre en place une liste noire et une liste blanche de serveurs de courriers électroniques, stockées dans le fichier /etc/postfix/access_clientip. Une liste blanche permet à l'administrateur d'y écrire les serveurs de confiance dispensés des règles suivantes.

Les trois dernières règles refusent tout message provenant d'un serveur présent dans l'une des différentes « listes noires » indiquées (RBL signifie *Remote Black Lists*, ou listes noires distantes). Celles-ci recensent les machines mal configurées employées par les spammeurs pour relayer leur courrier, ainsi que les relais inhabituels que constituent des machines infectées par des vers ou virus ayant cet effet.

ASTUCE Tables access

Les différents critères de restriction incluent des tables (modifiables par les administrateurs) de combinaisons expéditeurs/adresses IP/noms de machines autorisés ou interdits. Ces tables peuvent être créées en recopiant le fichier /etc/postfix/access sous le nom indiqué. C'est un modèle auto-documenté dans ses commentaires. Chaque table documentera ainsi sa propre syntaxe.
La table /etc/postfix/access_clientip donne la liste des adresses IP et réseau. La table /etc/postfix/access_helo fournit celle des noms de machines et de domaines. Enfin, la table /etc/postfix/access_sender précise les adresses électroniques. Après toute modification, chacun de ces fichiers doit être transformé en table de hachage par la commande **`postmap /etc/postfix/`*`fichier`***.

ASTUCE Liste blanche et RBL

Les listes noires recensent parfois un serveur légitime victime d'un incident. Tout courrier issu de ce serveur serait alors refusé à moins que vous ne l'ayez listé dans la liste blanche associée au fichier /etc/postfix/access_clientip.
Pour cette raison, il est prudent de placer dans une liste blanche les serveurs de messagerie de confiance et avec qui vous échangez beaucoup de courriers.

Vérifier la validité de la commande EHLO ou HELO

Chaque échange SMTP doit débuter par l'envoi d'une commande `HELO` (ou `EHLO`) suivie du nom du serveur de courrier électronique, dont il est possible de vérifier la validité.

EXEMPLE Restrictions sur le nom annoncé lors du EHLO

```
smtpd_helo_restrictions = permit_mynetworks, reject_invalid_hostname,
    check_helo_access hash:/etc/postfix/access_helo,
    reject_non_fqdn_hostname, warn_if_reject reject_unknown_hostname
```

La première directive `permit_mynetworks` autorise toutes les machines du réseau local à s'annoncer librement. C'est important car certains logiciels de courrier électronique respectent mal cette partie du protocole SMTP et peuvent donc annoncer des noms fantaisistes.

La règle `reject_invalid_hostname` refuse tout courrier dont l'annonce `EHLO` indique un nom de machine syntaxiquement incorrect. La règle `reject_non_fqdn_hostname` refuse tout message dont le nom de machine annoncé n'est pas complètement qualifié (un nom « qualifié » inclut le nom de domaine). La règle `reject_unknown_hostname` refuse le courrier si la machine annoncée n'existe pas dans la base de données du DNS. Cette dernière règle refusant malheureusement trop de messages, elle est atténuée par le `warn_if_reject` pour évaluer son impact avant de décider de l'activer ou non.

L'emploi de `permit_mynetworks` au début a l'effet secondaire intéressant de n'appliquer les règles suivantes qu'à des machines extérieures au réseau local. Il est ainsi possible de mettre en liste noire tous ceux qui s'annoncent membres du réseau `falcot.com`... ce qui s'effectue en ajoutant la ligne `falcot.com REJECT You're not in our network!` au fichier /etc/postfix/access_helo.

Accepter ou refuser en fonction de l'émetteur (annoncé)

Chaque message envoyé est associé à un expéditeur annoncé par la commande `MAIL FROM` du protocole SMTP, information qu'il est possible de vérifier de plusieurs manières.

EXEMPLE Vérifications sur l'expéditeur

```
smtpd_sender_restrictions =
    check_sender_access hash:/etc/postfix/access_sender,
    reject_unknown_sender_domain, reject_unlisted_sender,
    reject_non_fqdn_sender
```

La table /etc/postfix/access_sender associe des traitements particuliers à certains expéditeurs. En général, il s'agit simplement de les placer dans une liste blanche ou noire.

La règle `reject_unknown_sender_domain` requiert un domaine d'expéditeur valide, nécessaire à une adresse valide. La règle `reject_unlisted_sender`

refuse les expéditeurs locaux si leur adresse n'existe pas. Personne ne peut ainsi envoyer de courrier issu d'une adresse invalide dans le domaine `falcot.com`. Tout message d'expéditeur `tartempion@falcot.com` ne serait donc accepté que si cette adresse existe vraiment.

Enfin, la règle `reject_non_fqdn_sender` refuse les adresses électroniques dépourvues de domaine complètement qualifié. Concrètement, elle refusera un courrier provenant de `utilisateur@machine` : celui-ci doit s'annoncer comme `utilisateur@machine.domaine.fr` ou `utilisateur@domaine.fr`.

Accepter ou refuser en fonction du destinataire (annoncé)

Chaque courrier compte un ou plusieurs destinataires, communiqués par l'intermédiaire de la commande `RCPT TO` du protocole SMTP. On pourra également vérifier ces informations, même si c'est moins intéressant que pour l'expéditeur.

EXEMPLE Vérifications sur le destinataire

```
smtpd_recipient_restrictions = permit_mynetworks,
    reject_unauth_destination, reject_unlisted_recipient,
    reject_non_fqdn_recipient
```

`reject_unauth_destination` est la règle de base imposant à tout courrier provenant de l'extérieur de nous être destiné ; dans le cas contraire, il faut refuser de relayer le message. Sans cette règle, votre serveur est un relais ouvert qui permet aux spammeurs d'envoyer des courriers non sollicités par son intermédiaire. Elle est donc indispensable, et on la placera de préférence en début de liste pour qu'aucune autre règle ne risque d'autoriser le passage du courrier avant d'avoir éliminé les messages ne concernant pas ce serveur.

La règle `reject_unlisted_recipient` refuse les messages à destination d'utilisateurs locaux inexistants (ce qui est assez logique). Enfin, la règle `reject_non_fqdn_recipient` refuse les adresses électroniques non qualifiées. Il est ainsi impossible d'écrire à `jean` ou à `jean@machine` ; il faut impérativement employer la forme complète de l'adresse : `jean@machine.falcot.com` ou `jean@falcot.com`.

Restrictions associées à la commande DATA

La commande `DATA` du protocole SMTP précède l'envoi des données contenues dans le message. Elle ne fournit aucune information en soi, mais prévient de ce qui va suivre. Il est pourtant possible de lui mettre en place des contrôles.

EXEMPLE Restriction sur la commande DATA

```
smtpd_data_restrictions = reject_unauth_pipelining
```

La règle `reject_unauth_pipelining` refuse le message si le correspondant envoie une commande sans avoir attendu la réponse à la commande précédente. Les robots des spammeurs font régulièrement cela : pour travailler plus vite, ils se moquent des réponses et visent seulement à envoyer un maximum de courriers, dans le laps de temps le plus court.

Application des restrictions

Bien que toutes les règles évoquées ci-dessus soient prévues pour vérifier les informations à différents moments d'un échange SMTP, le refus réel n'est signifié par Postfix que lors de la réponse à la commande `RCPT TO` (annonce du destinataire).

Ainsi, même si le message est refusé suite à une commande `EHLO` invalide, Postfix connaîtra l'émetteur et le destinataire lorsqu'il annoncera le refus. Il peut donc enregistrer un message de log plus explicite que s'il avait interrompu la connexion dès le début. De plus, beaucoup de clients SMTP ne s'attendent pas à subir un échec sur l'une des premières commandes du protocole SMTP, et les clients mal programmés seront moins perturbés par ce refus tardif.

Dernier avantage de ce choix : les règles peuvent associer les informations obtenues à différents stades de l'échange SMTP. On pourra ainsi refuser une connexion non locale si elle s'annonce avec un émetteur local.

Filtrer en fonction du contenu du message

Le système de vérification et de restriction ne serait pas complet sans moyen de réagir au contenu du message. *Postfix* distingue deux types de vérifications : sur les en-têtes du courrier, et sur le corps du message.

EXEMPLE Activation des filtres sur le contenu

```
header_checks = regexp:/etc/postfix/header_checks
body_checks = regexp:/etc/postfix/body_checks
```

Les deux fichiers contiennent une liste d'expressions rationnelles (*regexp*). Chacune est associée à une action à exécuter si elle est satisfaite par les en-têtes ou le corps du message.

EXEMPLE Exemple de fichier `/etc/postfix/header_checks`

```
/^X-Mailer: GOTO Sarbacane/ REJECT I fight spam (GOTO Sarbacane)
/^Subject: *Your email contains VIRUSES/ DISCARD virus notification
```

La première vérifie l'en-tête indiquant le logiciel de courrier électronique envoyé : si elle trouve `GOTO Sarbacane` (un logiciel d'envoi en masse de courriers), elle refuse le message. La seconde expression contrôle le sujet du message : s'il indique

DÉCOUVERTE Tables regexp

Le fichier /etc/postfix/regexp_table peut servir de modèle pour créer les fichiers /etc/postfix/header_checks et /etc/postfix/body_checks. Il contient de nombreux commentaires explicatifs.

une notification de virus sans intérêt, elle accepte le message mais le supprime immédiatement.

L'emploi de ces filtres est à double tranchant, car il est facile de les faire trop génériques et de perdre des courriers légitimes. Dans ce cas, non seulement les messages seront perdus, mais leurs expéditeurs recevront des messages d'erreur inopportuns — souvent agaçants.

Intégration d'un antivirus

Avec les nombreux virus circulant en pièce jointe des courriers électroniques, il est important de placer un antivirus à l'entrée du réseau de l'entreprise car, même après une campagne de sensibilisation sur ce sujet, certains utilisateurs cliqueront sur l'icône d'une pièce jointe liée à un message manifestement très suspect.

Installation et configuration de l'antivirus

L'antivirus libre retenu par les administrateurs de Falcot est **`clamav`**. En plus du paquet *clamav*, ils ont installé les paquets *arj*, *unzoo*, *unrar* et *lha*, qui permettent aussi à l'antivirus d'analyser le contenu d'archives dans l'un de ces formats.

L'installation de **`clamav`** déclenche un certain nombre de questions de configuration, mais toutes les valeurs proposées par défaut conviennent.

Pour interfacer cet antivirus au serveur de messagerie, on emploiera le logiciel **`amavisd-new`**, mini-serveur de courrier électronique qui écoute sur le port 10024. Les courriers reçus sont analysés avant d'être renvoyés au serveur de courrier s'ils sont sains. Dans le cas contraire, le message est refusé voire supprimé (ce comportement est paramétrable dans le fichier /etc/amavis/amavisd.conf).

Après l'installation du paquet *amavisd-new*, il faut ajouter l'utilisateur `clamav` au groupe `amavis` pour qu'**`amavisd`** puisse piloter **`clamav`**.

Enfin, il faut personnaliser plusieurs paramètres dans le fichier /etc/amavis/amavisd.conf, et y indiquer le nom de domaine principal et la liste des domaines locaux (y compris virtuels).

```
$mydomain = "falcot.com"
@local_domains_acl = (".$mydomain", ".marqueafalcot.tm.fr", ".falcot.org
   ")
```

Ensuite, il faut préciser la manière de faire suivre les messages, en décommentant les deux lignes adaptées au cas de Postfix.

```
# where to forward checked mail
$forward_method = 'smtp:127.0.0.1:10025';

# where to submit notifications
$notify_method = $forward_method;
```

Le paramètre `$banned_filename_re` refuse par défaut les fichiers joints dotés d'une double extension (comme document.doc.pif, typique d'un nom de pièce attachée contenant un virus). On pourra encore durcir cette règle en interdisant directement tout exécutable.

Les paramètres `*_lovers` limitent les vérifications effectuées en fonction des destinataires. Tous les domaines hébergés sur une machine n'imposent pas systématiquement le même niveau de sécurité à l'entrée et ces variables permettent d'adapter les contrôles de manière à satisfaire tout le monde.

Le comportement par défaut d'**`amavisd-new`** est d'envoyer un message au *Postmaster* (l'administrateur supposé de la messagerie électronique) pour signaler chaque message mis en quarantaine pour cause de virus. Selon le trafic, ces messages peuvent devenir très vite gênants. On supprimera ces notifications en commentant la ligne positionnant la variable `$virus_admin` à l'adresse électronique du *Postmaster*.

Configuration de Postfix avec l'antivirus

Il faut configurer Postfix pour lui faire relayer le courrier à *amavis*, qui lui renverra par un autre canal (en l'occurrence, le port 10025). Les instructions de configuration sont détaillées dans le fichier /usr/share/doc/amavisd-new/README.postfix.gz.

On modifiera le fichier /etc/postfix/master.cf en lui ajoutant les lignes ci-dessous.

EXEMPLE **Lignes à ajouter au fichier `/etc/postfix/master.cf`**

```
smtp-amavis unix - - n - 2 lmtp
    -o lmtp_data_done_timeout=1200
    -o lmtp_send_xforward_command=yes
    -o disable_dns_lookups=yes
127.0.0.1:10025 inet n - n - - smtpd
    -o content_filter=
    -o receive_override_options=no_header_body_checks,
      no_unknown_recipient_checks
    -o local_recipient_maps=
    -o relay_recipient_maps=
    -o smtpd_restriction_classes=
    -o smtpd_client_restrictions=
    -o smtpd_helo_restrictions=
    -o smtpd_sender_restrictions=
    -o smtpd_recipient_restrictions=permit_mynetworks,reject
    -o mynetworks=127.0.0.0/8
    -o strict_rfc821_envelopes=yes
    -o smtpd_error_sleep_time=0
    -o smtpd_soft_error_limit=1001
    -o smtpd_hard_error_limit=1000
```

Tout est maintenant prêt pour recevoir les courriers d'**`amavis`**, mais il faut encore dévier les courriers entrants vers ce dernier. Cela s'effectue en ajoutant la directive *content_filter* :

EXEMPLE **Activation du filtre extérieur sur le contenu**

```
content_filter = smtp-amavis:[127.0.0.1]:10024
```

En cas de problèmes avec l'antivirus, il suffira de commenter cette ligne et d'exécuter la commande **`/etc/init.d/postfix reload`** pour faire prendre en compte cette modification.

Les messages traités par Postfix passent désormais systématiquement par un détecteur-filtre antivirus.

Serveur web (HTTP)

Les administrateurs de Falcot SA ont choisi Apache comme serveur HTTP. Ils ont hésité entre les versions 1.3 et 2.0, mais cette dernière ne leur semblait pas encore assez mûre, notamment en matière de prise en charge de PHP.

Installation d'Apache

L'installation du paquet *apache* provoque de nombreuses questions de configuration.

Le programme demande notamment s'il doit activer la prise en charge de `suExec`. Cette option est utile lorsque des inconnus peuvent exécuter des scripts CGI : les droits sont alors limités à ceux de leur propre compte (au lieu du compte `www-data` employé par le serveur web). Dans le cas de Falcot, aux utilisateurs internes et de confiance, cette option est désactivée.

DÉCOUVERTE **Liste des modules**

La liste complète des modules standards d'Apache se trouve en ligne.

▸ http://httpd.apache.org/docs/mod/index.html

L'installation propose également de sélectionner les modules Apache à charger. Chaque module offre une fonctionnalité spécifique, mais vous n'activerez que les modules utiles. Si vous ne connaissez pas leurs rôles, le plus sage est de se contenter de la liste de modules suggérée par défaut. Il sera toujours possible d'en activer d'autres plus tard avec la commande **`modules-config apache`**.

On vous demandera encore le nom du serveur (`www.falcot.com`), l'adresse électronique du *Webmaster* (`webmaster@falcot.com`, le responsable du site web), le répertoire racine du site web (/var/www par défaut) et le port sur lequel Apache attend les connexions (80 par défaut).

POUR ALLER PLUS LOIN **Prise en charge de SSL**

Pour mettre en place le HTTP sécurisé (HTTPS), il faut installer le paquet *libapache-mod-ssl*, qui propose au cours de l'installation de créer un certificat pour authentifier le serveur sécurisé.
Les changements de configuration nécessaires à Apache étant quelque peu complexes, il faut suivre les instructions données dans le fichier /usr/share/doc/libapache-mod-ssl/README.Debian.

Configuration d'hôtes virtuels

Un hôte virtuel est une identité supplémentaire assumée par le serveur web.

Apache distingue deux types d'hôtes virtuels : ceux qui se basent sur l'adresse IP et ceux qui reposent sur le nom DNS du serveur web. La première méthode nécessite une adresse IP différente pour chaque site tandis que la seconde n'emploie qu'une adresse IP et différencie les sites par le nom d'hôte communiqué par le client HTTP (ce qui ne fonctionne qu'avec la version 1.1 du protocole HTTP, heureusement déjà employée par tous les navigateurs web).

La rareté des adresses IPv4 fait en général privilégier cette deuxième méthode. Elle est cependant impossible si chacun des hôtes virtuels a besoin de HTTPS.

Pour mettre en place des hôtes virtuels basés sur le nom, il faut insérer dans le fichier /etc/apache/httpd.conf les lignes ci-dessous, juste avant la directive `Include /etc/apache/conf.d`.

```
NameVirtualHost *
<VirtualHost *>
ServerName www.falcot.com
ServerAlias falcot.com
DocumentRoot /srv/www/www.falcot.com
</VirtualHost>
```

ATTENTION **Premier hôte virtuel**
Le premier hôte virtuel défini répondra systématiquement aux requêtes concernant des hôtes virtuels inconnus. C'est pourquoi nous avons d'abord défini ici *www.falcot.com*.

Chaque hôte virtuel supplémentaire est ensuite décrit par un fichier placé dans le répertoire /etc/apache/conf.d. Ainsi, la mise en place du domaine *falcot.org* se résume à créer le fichier ci-dessous.

EXEMPLE **Fichier /etc/apache/conf.d/www.falco.org.conf**

```
<VirtualHost *>
ServerName www.falcot.org
ServerAlias falcot.org
DocumentRoot /srv/www/www.falcot.org
</VirtualHost>
```

Le serveur Apache est ici configuré pour n'utiliser qu'un seul fichier de log pour tous les hôtes virtuels (ce qu'on pourrait changer en intégrant des directives `CustomLog` dans les définitions des hôtes virtuels). Il est donc nécessaire de personnaliser le format de ce fichier pour y intégrer le nom de l'hôte virtuel. Pour cela, on ajoutera une ligne définissant le nouveau format de log (directive `LogFormat`), puis on changera le format employé par le fichier principal de log en modifiant la ligne `CustomLog`.

```
# Nouveau format de log avec virtual host (vhost)
LogFormat "%v %h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i
   \"" vhost

# On emploie le format vhost en standard
CustomLog /var/log/apache/access.log vhost
```

Directives courantes

Cette section passe brièvement en revue certaines directives de configuration d'Apache employées assez régulièrement par les administrateurs.

Le fichier de configuration principal contient habituellement plusieurs blocs `Directory` permettant de paramétrer le comportement du serveur en fonction de l'emplacement du fichier servi. À l'intérieur de ce bloc, on trouve généralement les directives `Options` et `AllowOverride`.

EXEMPLE **Bloc Directory**

```
<Directory /var/www>
Options Includes FollowSymlinks
AllowOverride All
DirectoryIndex index.php index.html index.htm
</Directory>
```

La directive `DirectoryIndex` précise la liste des fichiers à essayer pour répondre à une requête sur un répertoire. Le premier fichier existant est appelé pour générer la réponse.

La directive `Options` est suivie d'une liste d'options à activer. L'option `None` désactive toutes les options. Inversement, l'option `All` les active toutes sauf `MultiViews`. Voici les options existantes :

- `ExecCGI` indique qu'il est possible d'exécuter des scripts CGI.
- `FollowSymlinks` indique au serveur qu'il doit suivre les liens symboliques et donc effectuer la requête sur le fichier réel qui en est la cible.
- `SymlinksIfOwnerMatch` a le même rôle mais impose la restriction supplémentaire de ne suivre le lien que si le fichier pointé appartient au même propriétaire.
- `Includes` active les inclusions côté serveur (*Server Side Includes*, ou SSI). Il s'agit de directives directement intégrées dans les pages HTML et exécutées à la volée à chaque requête.
- `Indexes` autorise le serveur à retourner le contenu du dossier si la requête HTTP pointe sur un répertoire dépourvu de fichier d'index (tous les fichiers de la directive `DirectoryIndex` ayant été tentés en vain).
- `MultiViews` active la négociation de contenu, ce qui permet notamment au serveur de renvoyer la page web correspondant à la langue annoncée par le navigateur web.

La directive `AllowOverride` donne toutes les options qu'on peut activer ou désactiver par l'intermédiaire d'un fichier .htaccess. Il est souvent important de contrôler l'option `ExecCGI` pour rester maître des utilisateurs autorisés à exécuter un programme au sein du serveur web (sous l'identifiant `www-data`).

B.A.-BA **Fichier `.htaccess`**

Le fichier .htaccess contient des directives de configuration d'Apache, prises en compte à chaque fois qu'une requête concerne un élément du répertoire où est il stocké. Sa portée embrasse également les fichiers de toute l'arborescence qui en est issue.
La plupart des directives qu'on peut placer dans un bloc `Directory` peuvent également se trouver dans un fichier .htaccess.

Requérir une authentification

Il est parfois nécessaire de restreindre l'accès à une partie d'un site. Les utilisateurs légitimes doivent alors fournir un identifiant et un mot de passe pour accéder à son contenu.

EXEMPLE **Fichier `.htaccess` requérant une authentification**

```
Require valid-user
AuthName "Répertoire privé"
AuthType Basic
AuthUserFile /etc/apache/authfiles/htpasswd-prive
```

Le fichier /etc/apache/authfiles/htpasswd-prive contient la liste des utilisateurs et leur mots de passe ; on le manipule avec la commande **`htpasswd`**. Pour ajouter un utilisateur ou changer un mot de passe, on exécutera la commande suivante :

```
# htpasswd /etc/apache/authfiles/htpasswd-prive utilisateur
New password:
Re-type new password:
Adding password for user utilisateur
```

SÉCURITÉ **Aucune sécurité**
Ce système d'authentification (`Basic`) a une sécurité très faible puisque les mots de passe circulent sans protection (ils sont uniquement codés en *base64* — un simple encodage et non pas un procédé de chiffrement). Il faut noter que les documents protégés par ce mécanisme circulent également de manière non chiffrée. Si la sécurité vous importe, faites appel à SSL pour chiffrer toute la connexion HTTP.

Restrictions d'accès

Les directives `Allow from` (autoriser en provenance de) et `Deny from` (refuser en provenance de), qui s'appliquent à un répertoire et à toute l'arborescence qui en est issue, paramètrent les restrictions d'accès.

La directive `Order` indique dans quel ordre évaluer les directives `Allow from` et `Deny from`. Concrètement, `Order deny,allow` autorise l'accès si aucune des règles `Deny from` ne s'applique. Inversement, `Order allow,deny` refuse l'accès si aucune directive `Allow from` ne l'autorise.

Les directives `Allow from` et `Deny from` peuvent être suivies d'une adresse IP, d'un réseau (exemples : `192.168.0.0/255.255.255.0`,`192.168.0.0/24` et même `192.168.0`), d'un nom de machine ou de domaine, ou du mot clé `all` désignant tout le monde.

EXEMPLE **Interdire par défaut mais autoriser le réseau local**

```
Order deny,allow
Allow from 192.168.0.0/16
Deny from all
```

Analyseur de logs

L'analyseur de logs est un compagnon fréquent du serveur web puisqu'il permet aux administrateurs d'avoir une idée plus précise de l'usage fait de ce service.

Les administrateurs de Falcot SA ont retenu *AWStats* (*Advanced Web Statistics*, ou statistiques web avancées) pour analyser les fichiers de logs d'Apache.

La première étape de la configuration consiste à créer le fichier /etc/awstats/awstats.conf. Pour cela il est recommandé d'adapter le fichier modèle /usr/share/doc/awstats/examples/awstats.model.conf.gz, que les administrateurs de Falcot ont conservé tel quel en modifiant les différents paramètres donnés ci-dessous :

```
LogFile="/var/log/apache/access.log"
LogFormat = "%virtualname %host %other %logname %time1 %methodurl %code %
  bytesd %refererquot %uaquot"
SiteDomain="www.falcot.com"
HostAliases="falcot.com REGEX[^.*\.falcot\.com$]"
DNSLookup=1
DirData="/var/lib/awstats"
DirIcons="/awstats-icon"
DirLang="/usr/share/awstats/lang"
LoadPlugin="tooltips"
```

Tous ces paramètres sont documentés par commentaires dans le fichier modèle. Les paramètres `LogFile` et `LogFormat` indiquent l'emplacement du fichier de log et les informations qu'il contient. Les paramètres `SiteDomain` et `HostAliases` indiquent les différents noms associés au site web principal.

SÉCURITÉ **Accès aux statistiques**
Les statistiques d'*AWStats* sont disponibles sur le site web sans restrictions. On pourra le protéger de manière à ce que seules quelques adresses IP (internes probablement) puissent y accéder. Cela s'effectue en donnant la liste des adresses IP autorisées dans le paramètre `AllowAccessFromWebToFollowingIPAddresses`.

Pour les sites à fort trafic, il est déconseillé de positionner `DNSLookup` à `1` comme dans l'exemple ci-dessus. En revanche, pour les petits sites, ce réglage permet d'avoir des rapports plus lisibles qui emploient les noms complets des machines plutôt que leurs adresses IP.

On activera *AWStats* pour d'autres hôtes virtuels, en créant un fichier spécifique par hôte, comme par exemple /etc/awstats/awstats.www.falcot.org.conf.

EXEMPLE **Fichier de configuration pour un hôte virtuel**

```
Include "/etc/awstats/awstats.conf"
SiteDomain="www.falcot.org"
HostAliases="falcot.org"
```

Pour faire prendre en compte ce nouvel hôte virtuel, il faut modifier le fichier /etc/cron.d/awstats et y ajouter une invocation comme **`/usr/lib/cgi-bin/awstats.pl -config=www.falcot.org -update`**.

EXEMPLE **Fichier `/etc/cron.d/awstats`**

```
0,10,20,30,40,50 * * * * www-data [ -x /usr/lib/cgi-bin/awstats.pl -a -f
   /etc/awstats/awstats.conf -a -r /var/log/apache/access.log ] && /usr/
  lib/cgi-bin/awstats.pl -config=awstats -update >/dev/null && /usr/lib/
  cgi-bin/awstats.pl -config=www.falcot.org -update >/dev/null
```

AWStats emploie de nombreuses icônes stockées dans le répertoire /usr/share/awstats/icon. Pour les rendre disponibles sur le site web, il faut modifier la configuration d'Apache et y ajouter la directive suivante :

ATTENTION **Rotation de logs**

Pour que les statistiques prennent en compte tous les logs, il est impératif qu'*AWStats* soit invoqué juste avant la rotation des fichiers de logs d'Apache. Pour cela, on peut ajouter au fichier /etc/logrotate.d/apache une directive *prerotate*.

```
/var/log/apache/*.log {
  weekly
  missingok
  rotate 52
  compress
  delaycompress
  notifempty
  create 644 root adm
  sharedscripts
  prerotate
    su - www-data -c "/usr/lib/cgi-bin/awstats.pl -config=awstats -update > /dev/null"
    su - www-data -c "/usr/lib/cgi-bin/awstats.pl -config=www.falco.org -update > /dev/null"
  endscript
  postrotate
    /etc/init.d/apache reload > /dev/null
  endscript
}
```

Au passage, il est bon de s'assurer que les fichiers de logs mis en place par **`logrotate`** soient lisibles par tout le monde (et notamment AWStats). Dans l'exemple ci-dessus, c'est effectivement le cas (voir la ligne `create 644 root adm`).

```
Alias /awstats-icon/ /usr/share/awstats/icon/
```

Après quelques minutes (et les premières exécutions du script), le résultat est accessible en ligne :

- http://www.falcot.com/cgi-bin/awstats.pl
- http://www.falcot.org/cgi-bin/awstats.pl

Serveur de fichiers NFS

NFS (*Network File System*) est un protocole qui permet d'accéder à un système de fichiers à distance par le réseau, pris en charge par tous les systèmes Unix. Pour Windows, il faudra employer Samba.

NFS est un outil fort utile mais il ne faut jamais oublier ses limitations, surtout en termes de sécurité : toutes les données circulent en clair sur le réseau (un *sniffer* peut donc les intercepter) ; le serveur restreint les accès en fonction de l'adresse IP du client, ce qui le rend vulnérable au *spoofing* ; enfin, si une machine est autorisée à accéder à un système de fichiers NFS mal configuré, son utilisateur root peut accéder à tous les fichiers du partage (n'appartenant pas à root) puisque le serveur NFS utilise l'identifiant utilisateur que le client lui a communiqué (sans aucune vérification possible ; le protocole est ainsi conçu depuis le début).

DOCUMENTATION **NFS HOWTO**

Le NFS HOWTO contient de nombreuses informations intéressantes, notamment des méthodes pour optimiser les performances de NFS. On y découvre aussi un moyen de sécuriser les transferts NFS à l'aide d'un tunnel SSH (mais cette technique ne permet pas d'employer **`lockd`**).

- http://nfs.sourceforge.net/nfs-howto/

B.A.-BA **RPC**

RPC (*Remote Procedure Call*, ou appel de procédure distante) est un standard Unix pour des services distants. NFS est un service RPC. Les services RPC s'enregistrent dans un annuaire, le *portmapper*. Un client désireux d'effectuer une requête NFS s'adresse au *portmapper* (port 111 en TCP ou UDP) et lui demande où se trouve le serveur NFS. On lui répond généralement en indiquant le port 2049 (port par défaut pour NFS). Tous les services RPC ne disposent pas nécessairement d'un port fixe.

Sécuriser NFS (au mieux)

Étant donné que NFS fait confiance aux informations reçues par le réseau, il convient de s'assurer que seules les machines autorisées à l'employer peuvent se connecter aux différents serveurs RPC qui lui permettent de fonctionner. Le pare-feu doit donc prohiber le *spoofing* pour qu'une machine extérieure ne puisse pas se faire passer pour une machine intérieure, et les différents ports employés doivent être restreints aux machines devant accéder aux partages NFS.

D'autres services RPC sont nécessaires au fonctionnement optimal de NFS, notamment **`rpc.mountd`**, **`rpc.statd`** et **`lockd`**. Malheureusement, ils emploient par défaut un port aléatoire assigné par le **`portmapper`** et il est donc difficile de filtrer le trafic qui leur est destiné. Les administrateurs de Falcot SA ont découvert comment résoudre ce problème.

Les deux premiers services cités ci-dessus sont implémentés par des programmes utilisateur, démarrés respectivement par /etc/init.d/nfs-kernel-server et /etc/init.d/nfs-common. Ils disposent d'options pour forcer le choix des ports employés. Pour employer systématiquement les options adéquates, il faut modifier les fichiers /etc/default/nfs-kernel-server et /etc/default/nfs-common.

EXEMPLE **Fichier /etc/default/nfs-kernel-server**

```
# Number of servers to start up
RPCNFSDCOUNT=8

# Options for rpc.mountd
RPCMOUNTDOPTS="-p 2048"
```

EXEMPLE **Fichier /etc/default/nfs-common**

```
# Options for rpc.statd.
#   Should rpc.statd listen on a specific port?
#   If so, set this variable to a statd argument like: "--port 1000".
STATDOPTS="-p 2046 -o 2047"

# Are you _sure_ that your kernel does or does not need a lockd daemon?
# If so, set this variable to either "yes" or "no".
NEED_LOCKD=
```

Après ces modifications et un redémarrage des services, **`rpc.mountd`** emploie le port 2048 ; **`rpc.statd`** écoute le port 2046 et utilise le port 2047 pour les connections sortantes.

Le service **`lockd`** est géré par un *thread* (processus léger) noyau, fonctionnalité compilée sous forme de module dans les noyaux Debian. Le module dispose également de deux options pour choisir systématiquement le même port : `nlm_udpport` et `nlm_tcpport`. Pour employer ces options automatiquement, il faut créer un fichier /etc/modutils/lockd comme dans l'exemple ci-dessous, puis exécuter **`update-modules`**. Pour un noyau 2.6, il faut créer /etc/modprobe.d/lockd plutôt que /etc/modutils/lockd.

EXEMPLE **Fichier `/etc/modutils/lockd` ou `/etc/modprobe.d/lockd`**

```
options lockd nlm_udpport=2045 nlm_tcpport=2045
```

Avec tous ces paramétrages, il est maintenant possible de contrôler plus finement les accès au service NFS grâce à un pare-feu. Ce sont les ports 111 et 2045 à 2049 (en UDP et en TCP) qui doivent faire l'objet d'attentions particulières.

Serveur NFS

Le serveur NFS est intégré au noyau Linux, Debian le compile dans ses noyaux sous forme de module. Pour l'activer automatiquement à chaque démarrage, il faut installer le paquet *nfs-kernel-server*, qui contient les scripts d'initialisation adéquats.

Le fichier de configuration du serveur NFS, /etc/exports, donne les répertoires exportés à l'extérieur. À chaque partage NFS sont associées des machines qui ont le droit d'y accéder. Un certain nombre d'options permettent de dicter quelques règles d'accès. Le format de ce fichier est très simple :

```
/repertoire/a/partager machine1(option1,option2,...) machine2(...) ...
```

Chaque machine est identifiée par son nom DNS ou son adresse IP. Il est aussi possible de spécifier un ensemble de machines en employant la syntaxe `*.falcot.com` ou en décrivant une plage complète d'adresses IP (exemples : `192.168.0.0/255.255.255.0`, `192.168.0.0/24`).

Par défaut, un partage n'est accessible qu'en lecture seule (option `ro`). L'option `rw` donne un accès en lecture/écriture. Les clients NFS doivent se connecter depuis un port réservé à root (c'est-à-dire inférieur à 1024) à moins que l'option `insecure` (pas sûr) n'ait été employée (l'option `secure` — sûr — est implicite en l'absence de `insecure`, mais on peut quand même la mentionner).

Le serveur ne répond à une requête NFS que lorsque l'opération sur disque a été complétée (option `sync`). L'option `async` (asynchrone) désactive cette fonctionnalité et améliore quelque peu les performances, au détriment de la fiabilité puisqu'il subsiste alors un risque de perte de données en cas de *crash* du serveur (des données acquittées par le serveur NFS n'auront pas été sauvegardées sur le disque avant le *crash*). La valeur par défaut de cette option ayant changé récemment, il est recommandé de toujours mentionner explicitement l'option souhaitée.

Pour ne pas donner un accès root au système de fichiers à n'importe quel client NFS, toutes les requêtes provenant d'un utilisateur root sont transformées en requêtes provenant de l'utilisateur `anonymous`. Cette option (`root_squash`)

ALTERNATIVE **Le serveur nfs-user-server**

nfs-user-server est un serveur NFS fonctionnant comme un serveur traditionnel, à l'aide d'un programme et non pas d'un module noyau. Cette version de NFS est quasiment obsolète depuis que la prise en charge de NFS intégrée au noyau est fiable.

est activée par défaut ; l'option inverse `no_root_squash` ne doit être employée qu'avec parcimonie étant donné les risques qu'elle comporte. Les options `anonuid=`*`uid`* et `anongid=`*`gid`* permettent d'employer un autre utilisateur écran à la place d'`anonymous`.

D'autres options existent encore, que vous découvrirez dans la page de manuel `exports(5)`.

ATTENTION **Première installation**

Le script de démarrage /etc/init.d/nfs-kernel-server ne démarre rien si le fichier /etc/exports ne prévoit aucun partage NFS. C'est pourquoi il faut démarrer le serveur NFS juste après avoir rempli ce fichier pour la première fois :

```
# /etc/init.d/nfs-kernel-server start
```

Client NFS

Comme tous les systèmes de fichiers, il est nécessaire de le monter pour l'intégrer dans l'arborescence du système. Étant donné qu'il s'agit d'un système de fichiers un peu particulier, il a fallu adapter la syntaxe habituelle de la commande **`mount`** et le format du fichier /etc/fstab.

EXEMPLE **Montage manuel avec la commande `mount`**

```
# mount -t nfs -o rw,nosuid arrakis.interne.falco.com:/org/partage /
  partage
```

EXEMPLE **Entrée NFS dans le fichier `/etc/fstab`**

```
arrakis.interne.falco.com:/org/partage /partage nfs rw,nosuid 0 0
```

L'entrée ci-dessus monte automatiquement à chaque démarrage le répertoire NFS /org/partage présent sur le serveur `arrakis` dans le répertoire local /partage. L'accès demandé est en lecture/écriture (`rw` comme *read-write* ; pour un accès en lecture seule il aurait fallu indiquer `ro` comme *read-only*). L'option `nosuid` est une mesure de protection qui supprime tout bit `setuid` ou `setgid` présent sur les programmes contenus dans le partage NFS. Si le répertoire NFS est dédié au stockage de documents, il est recommandé d'employer de plus l'option `noexec` qui empêche l'exécution de programmes par NFS.

La page de manuel `nfs(5)` détaille toutes les options possibles.

Partage Windows avec Samba

Samba est une suite d'outils qui permettent de gérer le protocole SMB (maintenant appelé « CIFS ») sous Linux. Ce dernier est employé par Windows pour accéder aux partages réseau et aux imprimantes partagées.

Samba sait également jouer le rôle de contrôleur de domaine NT. C'est un outil extraordinaire pour assurer une cohabitation parfaite entre les serveurs sous Linux et les machines de bureautique encore sous Windows.

Samba en serveur

Le paquet Debian *samba* contient les deux principaux serveurs de Samba 3 (**smbd** et **nmbd**).

OUTIL **Authentifier à l'aide d'un serveur Windows**

Winbind permet d'utiliser un serveur Windows NT comme serveur d'authentification et s'intègre à PAM et à NSS. Il est ainsi possible de mettre en place des machines Linux où tous les utilisateurs d'un domaine NT disposeront automatiquement d'un compte.
Vous trouverez plus d'informations à ce sujet dans le fichier /usr/share/doc/samba-doc/htmldocs/howto/winbind.html.

Configuration avec debconf

Le paquet met en place une configuration minimale en posant quelques questions au cours de l'installation initiale. Il est possible de reprendre cette étape de la configuration avec la commande **dpkg-reconfigure samba-common samba**.

La première information demandée est le nom du groupe de travail auquel le serveur Samba appartient (dans notre cas, la réponse est FALCOTNET). Une autre question demande s'il faut employer les mots de passe chiffrés ; la réponse est « oui » : c'est nécessaire pour fonctionner avec les clients Windows les plus récents

OUTIL **Administrer Samba avec SWAT**

SWAT (*Samba Web Administration Tool*, outil d'administration web de Samba) est une interface web permettant de configurer le service Samba. Le paquet Debian *swat* n'activant pas l'interface de configuration par défaut, il faut le faire manuellement en exécutant **update-inetd --enable swat**.
SWAT est alors accessible à l'URL http://localhost:901. Pour y accéder, il faut employer le compte root (et le mot de passe administrateur habituel). Attention cependant, SWAT réécrit le fichier smb.conf à sa manière, pensez-donc à en faire une copie préalable si vous ne faites qu'essayer cet outil.
SWAT, très agréable à utiliser, dispose d'un assistant qui permet de définir le rôle du serveur en trois questions. Il est ensuite possible de configurer toutes les options globales ainsi que celles de tous les partages. On peut bien entendu créer de nouveaux partages. Chaque option est accompagnée d'un lien qui renvoie à la documentation correspondante.

DOCUMENTATION **Pour aller plus loin**

Le serveur Samba est extrêmement configurable et peut répondre à de très nombreux cas d'utilisation correspondant à des besoins et des architectures réseau très différents. Le cas traité dans ce livre utilise Samba comme contrôleur de domaine principal, mais il peut très bien n'être qu'un serveur du domaine déléguant l'authentification au contrôleur principal, qui serait un serveur Windows NT ou Windows Server 2003.
La documentation présente dans le paquet *samba-doc* est très bien faite. Je citerai en particulier le document *Samba 3 by example* : /usr/share/doc/samba-doc/htmldocs/guide/index.html. Ce texte traite d'un cas concret, évoluant au fil de la croissance de l'entreprise.

et cela augmente la sécurité (la contre-partie étant l'obligation de gérer les mots de passe des utilisateurs de manière séparée des mots de passe Unix).

Le paquet propose également d'identifier le serveur Wins grâce aux informations fournies par le démon DHCP. Les administrateurs de Falcot ont refusé cette option, puisque leur intention était d'employer Samba pour jouer aussi le rôle de serveur Wins !

Ensuite, l'ordinateur demande de choisir comment les démons sont démarrés : soit par l'intermédiaire de **`inetd`**, soit en tant que démons indépendants. Cette seconde option fut retenue parce que l'emploi d'**`inetd`** ne se justifie que si Samba est utilisé très occasionnellement, ce qui n'est pas le cas chez Falcot.

Enfin, le paquet a proposé de créer un fichier /var/lib/samba/passdb.tdb pour stocker les mots de passe chiffrés, option acceptée parce que ce système est bien plus efficace que le fichier texte standard /etc/samba/smbpasswd.

Configuration manuelle

Modifications à `smb.conf`

Pour adapter le serveur aux besoins de Falcot, il faut modifier d'autres options dans le fichier de configuration de Samba, /etc/samba/smb.conf. Les extraits ci-dessous résument les changements effectués au sein de la section `[global]`.

```
[global]

## Browsing/Identification ###

# Change this to the workgroup/NT-domain name your Samba server will part
    of
   workgroup = FALCOTNET

# server string is the equivalent of the NT Description field
   server string = %h server (Samba %v)

# Windows Internet Name Serving Support Section:
# WINS Support - Tells the NMBD component of Samba to enable its WINS
   Server
   wins support = yes ❶

[...]
```

```
####### Authentication #######

# "security = user" is always a good idea. This will require a Unix
   account
# in this server for every user accessing the server. See
# /usr/share/doc/samba-doc/htmldocs/ServerType.html in the samba-doc
# package for details.
   security = user ❷

# You may wish to use password encryption.  See the section on
# 'encrypt passwords' in the smb.conf(5) manpage before enabling.
   encrypt passwords = true

# If you are using encrypted passwords, Samba will need to know what
# password database type you are using.
   passdb backend = tdbsam guest

[...]

########## Printing ##########

# If you want to automatically load your printer list rather
# than setting them up individually then you'll need this
   load printers = yes ❸

# lpr(ng) printing. You may wish to override the location of the
# printcap file
;   printing = bsd
;   printcap name = /etc/printcap

# CUPS printing.  See also the cupsaddsmb(8) manpage in the
# cupsys-client package.
   printing = cups ❹
   printcap name = cups

[...]

######## File sharing ########

# Name mangling options
;   preserve case = yes
;   short preserve case = yes

unix charset=ISO8859-1 ❺
```

❶ Indique que Samba doit jouer le rôle de serveur de nom Netbios (Wins) pour le réseau local.

❷ C'est la valeur par défaut de ce paramètre. Comme il est central à la configuration de Samba, il est toutefois raisonnable de le renseigner de manière explicite. Chaque utilisateur doit s'authentifier avant de pouvoir accéder au moindre partage.

❸ Demande à Samba de partager automatiquement toutes les imprimantes existantes en local dans la configuration de Cupsys. Il est toujours possible de restreindre les droits sur ces imprimantes en définissant des sections appropriées dans le fichier.

❹ Documente le système d'impression employé, en l'occurrence Cupsys.

❺ Indique le jeu de caractères employé (sous Linux) dans les noms de fichiers. Valeur par défaut : UTF8 (Unicode).

Ajout des utilisateurs

Chaque utilisateur de Samba ayant besoin d'un compte sur le serveur, il faut créer les comptes Unix puis enregistrer chaque utilisateur dans la base de données de Samba. La création des comptes Unix se fait tout à fait normalement (avec la commande **`adduser`** par exemple).

L'ajout d'un utilisateur existant dans la base de données de Samba s'effectue par la commande **`smbpasswd -a utilisateur`**, qui demande le mot de passe interactivement.

On supprime un utilisateur avec la commande **`smbpasswd -x utilisateur`**. Un compte Samba peut n'être que gelé quelque temps avec la commande **`smbpasswd -d utilisateur`**, puis réactivé avec **`smbpasswd -e utilisateur`**.

Transformation en contrôleur de domaines

Cette section indique comment les administrateurs de Falcot sont allés encore plus loin en transformant le serveur Samba en contrôleur de domaines offrant des profils errants (qui permettent aux utilisateurs de retrouver leur bureau quelle que soit la machine sur laquelle ils se connectent).

Tout d'abord, ils ont rajouté des directives supplémentaires dans la section `[global]` du fichier de configuration :

```
domain logons = yes              ❶
preferred master = yes
logon path = \\%L\profiles\%U    ❷
logon script = scripts/logon.bat ❸
```

❶ Active la fonctionnalité de contrôleur de domaine.

❷ Indique l'emplacement des répertoires personnels des utilisateurs. Ceux-ci sont stockés sur un partage dédié afin de pouvoir activer des options spécifiques (en l'occurrence `profile acls`, qui est nécessaire pour la compatibilité avec Windows 2000 et XP).

❸ Indique le script *batch* (non interactif) à exécuter à chaque ouverture de session sur la machine Windows cliente. En l'occurrence, il s'agit de /var/lib/samba/netlogon/scripts/logon.bat. Le script doit être au format DOS (les lignes étant séparées par un retour chariot et un saut de ligne ; il suffit d'exécuter **`unix2dos`** sur le fichier créé depuis Linux pour s'en assurer).

Les commandes les plus couramment employées dans ces scripts permettent de créer automatiquement des lecteurs réseau et de synchroniser l'heure de l'ordinateur.

EXEMPLE **Fichier `logon.bat`**

```
net time \\ARRAKIS /set /yes
net use H: /home
net use U: \\ARRAKIS\utils
```

Deux partages supplémentaires et leurs répertoires associés ont aussi été créés :

```
[netlogon]
comment = Network Logon Service
path = /var/lib/samba/netlogon
guest ok = yes
writable = no
share modes = no

[profiles]
comment = Profile Share
path = /var/lib/samba/profiles
read only = No
profile acls = Yes
```

Il faut également créer les répertoires personnels de tous les utilisateurs (/var/lib/samba/profiles/<*utilisateur*>), chacun d'entre eux devant être propriétaire de son répertoire personnel.

Samba en client

Les fonctionnalités clientes de Samba donnent à une machine Linux l'accès à des partages Windows et à des imprimantes partagées. Les paquets Debian *smbfs* et *smbclient* regroupent les programmes clients nécessaires.

Le programme `smbclient`

Le programme **`smbclient`** interroge tous les serveurs SMB. Il accepte l'option **`-U utilisateur`** pour se connecter au serveur sous une autre identité. **`smbclient //serveur/partage`** accède au partage de manière interactive (comme le client FTP en ligne de commande). **`smbclient -L serveur`** donne la liste des partages disponibles (et visibles).

Monter un partage Windows

Le programme **`smbmount`** permet de monter un partage Windows dans l'arborescence du système Linux.

EXEMPLE Montage d'un partage Windows

```
smbmount //arrakis/partage /partage -o credentials=/usr/local/etc/smb-
  credentials
```

Le fichier /usr/local/etc/smb-credentials ne sera pas lisible par les utilisateurs et respectera le format suivant :

```
username = utilisateur
password = mot_de_passe
```

On peut préciser d'autres options sur la ligne de commande, que la page de manuel `smbmount(1)` détaille. Deux options intéressantes permettent de forcer l'utilisateur (`uid`) et le groupe (`gid`) propriétaire des fichiers accessibles sur le montage afin de ne pas restreindre l'accès à root.

Le programme **`smbumount`** démonte un partage SMB.

ALTERNATIVE **Utiliser `mount` pour un partage Windows**

La commande **`mount`** ne gère pas *smbfs*, mais face à un système de fichiers inconnu elle tente de déléguer la tâche à la commande **`mount.`*`type`***. Le paquet **`smbfs`** fournissant cette dernière, il est possible de monter un partage Windows avec la commande **`mount`** :

```
mount -t smbfs -o credentials=/usr/local/etc/smb-credentials //
  serveur/partage /partage
```

On peut donc aussi préciser un montage **`smbfs`** dans le fichier /etc/fstab :

```
//serveur/partage /partage smbfs credentials=/usr/local/etc/smb-
  credentials
```

Imprimer sur une imprimante partagée

Cupsys est une solution élégante pour imprimer sur une imprimante partagée par une machine Windows depuis un poste Linux. Si le paquet *smbclient* est installé, Cupsys offre la possibilité d'installer automatiquement une imprimante partagée par un poste Windows.

Voici les étapes à suivre :

- Rentrez dans l'interface de configuration de Cupsys : `http://localhost:631/admin`.
- Cliquez sur « Ajouter imprimante » puis saisissez les données de cette imprimante.
- Lors du choix du périphérique de l'imprimante, il faut choisir « *Windows Printer via SAMBA* ».
- L'URI décrivant l'imprimante doit avoir la forme suivante : `smb://`*`utilisateur`*`:`*`motdepasse`*`@`*`serveur`*`/`*`imprimante`*.

Et voilà, l'imprimante est fonctionnelle !

Mandataire HTTP/FTP

Un mandataire HTTP/FTP (ou proxy) est un intermédiaire pour les connexions HTTP et/ou FTP. Son rôle est double :

- Celui de serveur cache : il garde une copie des documents téléchargés pour éviter de les rapatrier plusieurs fois.

- Celui de serveur filtrant s'il est obligatoire et que les connexions sortantes sont par ailleurs bloquées. En tant qu'intermédiaire inévitable, il a en effet la liberté d'effectuer ou non la requête demandée.

Le serveur mandataire employé par Falcot SA est Squid.

Installation

Le paquet Debian *squid* n'est qu'un mandataire modulaire. Pour le transformer en serveur filtrant, il faut lui adjoindre le paquet *squidguard*. Le paquet *squid-cgi* permet d'interroger et d'administrer un mandataire Squid.

Préalablement à l'installation, il faut vérifier que le système est capable d'identifier son nom complet. La commande **`hostname -f`** doit renvoyer un nom long (incluant un nom de domaine). Si ce n'est pas le cas, il faut modifier /etc/hosts pour documenter le nom complet du système (exemple : `arrakis.falcot.com`). N'hésitez pas à faire valider le nom officiel de l'ordinateur avec votre administrateur réseau afin de ne pas créer de conflits inutiles.

Configuration d'un cache

Pour activer la fonctionnalité de serveur cache, il suffit de modifier le fichier de configuration /etc/squid/squid.conf pour autoriser les machines du réseau local à effectuer des requêtes au travers du mandataire. L'exemple ci-dessous montre les modifications effectuées par les administrateurs de Falcot SA.

EXEMPLE **Extrait du fichier `/etc/squid/squid.conf`**

```
# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS

# Example rule allowing access from your local networks. Adapt
# to list your (internal) IP networks from where browsing should
# be allowed
acl our_networks src 192.168.1.0/24 192.168.2.0/24
http_access allow our_networks
http_access allow localhost
# And finally deny all other access to this proxy
http_access deny all
```

Configuration d'un filtre

Le filtrage des requêtes n'est pas effectué par **`squid`** mais par **`squidGuard`**. Il faut donc configurer **`squid`** pour qu'il interagisse avec ce dernier, ce qui s'effectue en ajoutant au fichier /etc/squid/squid.conf la directive ci-dessous :

```
redirect_program /usr/bin/squidGuard -c /etc/squid/squidGuard.conf
```

Il faut également installer le programme CGI /usr/lib/cgi-bin/squidGuard.cgi à partir du fichier d'exemple squidGuard.cgi.gz, que l'on trouve dans le répertoire /usr/share/doc/squidguard/examples/. On modifiera ce script en changeant les variables `$proxy` (nom du serveur mandataire) et `$proxymaster` (courrier électronique de contact de l'administrateur). Les variables `$image` et `$redirect` devront pointer sur des images existantes, symbolisant le refus d'accéder à la page demandée.

La commande **`/etc/init.d/squid reload`** active le filtre. Le paquet *squidguard* n'offrant aucun filtrage par défaut, c'est la responsabilité de l'administrateur de le définir. Pour cela, il doit personnaliser le fichier /etc/squid/squidGuard.conf.

Après chaque modification du fichier de configuration de **`squidGuard`** ou de l'une des listes de domaines ou d'URL qu'il mentionne, il est nécessaire de régénérer la base de données de travail. Cela s'effectue en exécutant la commande **`update-squidguard`**. Le format du fichier de configuration est documenté sur le site web ci-dessous :

- http://www.squidguard.org/config/

Les fichiers fournis par le paquet *chastity-list* constituent de bons exemples pour la configuration de **`squidGuard`**.

OUTIL **Paquet chastity-list**

Le paquet *chastity-list* offre un ensemble de règles prêtes à l'emploi pour filtrer des sites pornographiques, des sites dédiés au piratage, etc. Pour les activer, il suffit de remplacer dans le fichier /etc/squid/squid.conf la ligne habituelle par la ligne suivante :

```
redirect_program /usr/bin/squidGuard -c /etc/chastity/squidGuard-
  chastity.conf
```

Annuaire LDAP

OpenLDAP implémente le protocole LDAP ; ce n'est qu'une base de données adaptée pour gérer des annuaires. Son intérêt est multiple : l'emploi d'un serveur LDAP permet de centraliser la gestion des comptes des utilisateurs et des droits associés. De plus, la base de données LDAP est facile à dupliquer, ce qui permet de mettre en place plusieurs serveurs LDAP synchronisés. En cas de croissance rapide du réseau, il sera aisé de monter en puissance en répartissant la charge sur plusieurs serveurs LDAP.

Les données LDAP sont structurées et hiérarchisées. Les « schémas » définissent les objets que la base peut stocker avec la liste de tous les attributs possibles. La syntaxe qui permet de désigner un objet de la base traduit cette structure, même si elle n'est pas aisée à maîtriser.

Installation

Le paquet *slapd* contient le serveur OpenLDAP. Le paquet *ldap-utils* renferme des utilitaires en ligne de commande pour interagir avec les serveurs LDAP.

L'installation du paquet *slapd* pose plusieurs questions par l'intermédiaire de **debconf**.

- Faut-il ignorer la configuration de **slapd** ? Non bien sûr, nous allons configurer ce service.
- Quel est le nom de domaine ? « falcot.com ».
- Quel est le nom de l'organisation ? « Falcot SA ».
- Il faut saisir un mot de passe administrateur pour la base de données.
- La base doit-elle être supprimée si le paquet **slapd** est supprimé ? Non. Mieux vaut éviter de perdre ces données suite à une mauvaise manipulation.
- Faut-il autoriser LDAPv2 ? Non, ce n'est pas la peine. Tous les outils que nous employons connaissent LDAPv3.

B.A.-BA **Format LDIF**

Un fichier LDIF (*LDAP Data Interchange Format*, ou format d'échange de données de LDAP) est un fichier textuel portable décrivant le contenu (ou une partie de celui-ci) d'une base de données LDAP afin de pouvoir intégrer les données dans n'importe quel autre serveur LDAP.

Une base de données minimale est maintenant configurée, ce qu'on peut vérifier en l'interrogeant directement :

```
$ ldapsearch -x -b dc=falcot,dc=com
# extended LDIF
#
# LDAPv3
# base <dc=falcot,dc=com> with scope sub
# filter: (objectclass=*)
# requesting: ALL
#

# falcot.com
dn: dc=falcot,dc=com
objectClass: top
objectClass: dcObject
objectClass: organization
o: Falcot SA
dc: falcot

# admin, falcot.com
dn: cn=admin,dc=falcot,dc=com
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator

# search result
search: 2
result: 0 Success

# numResponses: 3
# numEntries: 2
```

La requête a renvoyé deux objets : l'organisation dans son ensemble et l'administrateur.

Remplissage de l'annuaire

La base de données vide n'ayant pas grand intérêt, il s'agit maintenant d'y intégrer l'ensemble des annuaires existants, et notamment les utilisateurs, groupes, services et hôtes.

Le paquet Debian *migrationtools* offre un ensemble de scripts qui permettent justement de récupérer les informations depuis les annuaires Unix standards (/etc/passwd, /etc/group, /etc/services, /etc/hosts, etc.) puis de les intégrer dans la base de données LDAP.

Après installation du paquet, il faut éditer le fichier /usr/share/migrationtools/migrate_common.ph pour activer les options `IGNORE_UID_BELOW` et `IGNORE_GID_BELOW` (qu'il suffit de décommenter).

La mise à jour à proprement parler se fait en exécutant la commande **`migrate_all_online.sh`** comme suit :

```
# cd /usr/share/migrationtools
# LDAPADD="/usr/bin/ldapadd -c" ETC_ALIASES=/dev/null ./
  migrate_all_online.sh
```

Le script **`migrate_all_online.sh`** pose plusieurs questions auxquelles il faut répondre correctement pour indiquer la base de données LDAP dans laquelle les données vont être intégrées. Le tableau ci-dessous résume les réponses données dans le cas de Falcot.

Tableau 11–1 Réponses aux questions du script **`migrate_all_online.sh`**

Question	Réponse
X.500 naming context	`dc=falcot,dc=com`
LDAP server hostname	`localhost`
Manager DN	`cn=admin,dc=falcot,dc=com`
Bind credentials	le mot de passe administrateur
Create DUAConfigProfile	*no*

OUTIL **Explorer un annuaire LDAP**

Le programme **gq** (du paquet Debian éponyme) est un outil graphique qui permet d'explorer et de modifier une base de données LDAP. Il est intéressant et permet notamment de mieux se représenter la structure hiérarchique des données LDAP.

La migration du fichier /etc/aliases est volontairement ignorée parce que le schéma standard (installé par Debian) ne comprend pas les structures employées par ce script pour décrire les alias de courrier électronique. S'il est nécessaire d'intégrer cette information dans la base de données LDAP, il faudra ajouter le fichier /etc/ldap/schema/misc.schema comme schéma standard dans le fichier /etc/ldap/slapd.conf.

On peut également noter l'emploi de l'option **`-c`** de la commande **`ldapadd`** lui demandant de ne pas s'interrompre en cas d'erreur. Elle est nécessaire car la conversion du fichier /etc/services génère quelques erreurs que l'on peut ignorer sans soucis.

Utiliser LDAP pour gérer les comptes

Maintenant que la base de données LDAP contient des informations, il est temps de les utiliser. Cette section explique comment paramétrer un système Linux afin que les différents annuaires système emploient la base de données LDAP de manière transparente.

Configuration de NSS

Le système NSS (*Name Service Switch*, ou multiplexeur de service de noms, voir page 118) est un système modulaire pour définir ou récupérer les informations des annuaires système. Pour utiliser LDAP comme une source de données NSS, il faut mettre en place le paquet *libnss-ldap*. Son installation pose plusieurs questions dont les réponses sont résumées dans le tableau 11.2.

Tableau 11–2 Configuration du paquet *libnss-ldap*

Question	Réponse
Nom du serveur LDAP	`ldap.falcot.com`
Nom distinctif de la base de recherche	`dc=falcot,dc=com`
Version de LDAP à utiliser	`3`
La base demande-t-elle un *login* ?	non
Le fichier de configuration doit-il être restreint en lecture ?	non (ce n'est important que si le fichier contient un *login*/mot de passe pour se connecter au serveur LDAP, ce qui n'est pas le cas ici)

Il faut ensuite modifier le fichier /etc/nsswitch.conf pour lui indiquer d'employer le module **`ldap`** fraîchement installé.

EXEMPLE **Fichier `/etc/nsswitch.conf`**

```
# /etc/nsswitch.conf
#
# Example configuration of GNU Name Service Switch functionality.
# If you have the 'glibc-doc' and 'info' packages installed, try:
# 'info libc "Name Service Switch"' for information about this file.

passwd: ldap compat
group: ldap compat
shadow: ldap compat

hosts: files dns ldap
networks: ldap files

protocols: ldap db files
services: ldap db files
ethers: ldap db files
rpc: ldap db files

netgroup: files
```

Le module **ldap**, systématiquement ajouté au début, est donc consulté en premier. Le service `hosts` fait exception puisque pour contacter le serveur LDAP, il faut consulter le DNS au préalable (pour résoudre `ldap.falcot.com`). Sans cette précaution, une requête de résolution de nom de machine consulterait le serveur LDAP, ce qui déclencherait une résolution du nom du serveur LDAP, etc., produisant une boucle infinie. Quant au service `netgroup`, il n'est pas encore pris en charge par LDAP.

Si l'on souhaite que le serveur LDAP soit la référence unique (et ne pas prendre en compte les fichiers locaux employés par le module *files*), il est possible de configurer chaque service avec la syntaxe :

service: ldap [NOTFOUND=return] files.

Si l'entrée demandée n'existe pas dans le serveur LDAP, la réponse sera « n'existe pas » même si la ressource existe dans l'un des fichiers locaux, qui ne seront employés que lorsque le service LDAP sera hors d'usage.

Configuration de PAM

La configuration de PAM (voir l'encadré « EN COULISSES » page 111) proposée dans cette section permettra aux applications d'effectuer les authentifications nécessaires à partir des données de la base LDAP.

ATTENTION **Impossible de s'identifier**

Le changement de la configuration PAM standard employée par les divers programmes est une opération sensible. En cas de mauvaise manipulation, il peut être impossible de s'authentifier, donc de se connecter. Pensez donc à garder un shell root ouvert en parallèle pour pouvoir corriger vos erreurs le cas échéant.

Il faut installer le module LDAP pour PAM, qui se trouve dans le paquet Debian *libpam-ldap*. Son installation pose des questions similaires à celles de *libnss-ldap*. Les réponses sont résumées dans le tableau 11.3.

Après installation et configuration du module *libpam-ldap*, il reste à adapter la configuration PAM par défaut en modifiant les fichiers /etc/pam.d/common-auth, /etc/pam.d/common-password, et /etc/pam.d/common-account comme dans les exemples suivants.

EXEMPLE **Fichier /etc/pam.d/common-auth**

```
#
# /etc/pam.d/common-auth - authentication settings common to all services
#
# This file is included from other service-specific PAM config files,
# and should contain a list of the authentication modules that define
# the central authentication scheme for use on the system
# (e.g., /etc/shadow, LDAP, Kerberos, etc.).  The default is to use the
# traditional Unix authentication mechanisms.
#
auth    sufficient      pam_ldap.so
auth    required        pam_unix.so try_first_pass nullok_secure
```

Tableau 11–3 Configuration de *libpam-ldap*

Question	Réponse
Nom du serveur LDAP	`ldap.falcot.com`
Nom distinctif de la base de recherche	`dc=falcot,dc=com`
Version de LDAP à utiliser	`3`
Faut-il faire de l'administrateur local ("root") un administrateur de la base LDAP ?	oui (cette option permet d'utiliser la commande **`passwd`** habituelle pour changer les mots de passe stockés dans la base LDAP)
La base demande-t-elle un *login* ?	non
Root login account	`cn=admin,dc=falcot,dc=com`
Root login password	le mot de passe administrateur de la base LDAP
Local crypt to use when changing password	`crypt`

EXEMPLE **Fichier `/etc/pam.d/common-password`**

```
#
# /etc/pam.d/common-password - password-related modules common to all
# services
#
# This file is included from other service-specific PAM config files,
# and should contain a list of modules that define  the services to be
#used to change user passwords.  The default is pam_unix

password sufficient pam_ldap.so

# The "nullok" option allows users to change an empty password, else
# empty passwords are treated as locked accounts.
#
# (Add ‘md5’ after the module name to enable MD5 passwords)
#
# The "obscure" option replaces the old ‘OBSCURE_CHECKS_ENAB’ option in
# login.defs. Also the "min" and "max" options enforce the length of the
# new password.

password   required   pam_unix.so nullok obscure min=4 max=8 md5

# Alternate strength checking for password. Note that this
# requires the libpam-cracklib package to be installed.
# You will need to comment out the password line above and
# uncomment the next two in order to use this.
# (Replaces the ‘OBSCURE_CHECKS_ENAB’, ‘CRACKLIB_DICTPATH’)
#
# password required       pam_cracklib.so retry=3 minlen=6 difok=3
# password required       pam_unix.so use_authtok nullok md5
```

EXEMPLE **Fichier `/etc/pam.d/common-account`**

```
#
# /etc/pam.d/common-account - authorization settings common to all
# services
#
# This file is included from other service-specific PAM config files,
# and should contain a list of the authorization modules that define
```

```
# the central access policy for use on the system.  The default is to
# only deny service to users whose accounts are expired in /etc/shadow.
#
account sufficient      pam_ldap.so
account required        pam_unix.so try_first_pass
```

ATTENTION **Services mal configurés**

Les différents fichiers /etc/pam.d/common-* sont prévus pour être employés de manière standard par tous les services (grâce à une directive `@include`), mais ce n'est malheureusement pas encore le cas de tous. **sudo** est une exception à la règle, et continuera à utiliser pam_unix.so même après les modifications indiquées dans ce chapitre. Dans ce cas, les services deviennent non fonctionnels puisque la base *shadow* renvoyée par le module *libnss-ldap* ne mentionne aucun mot de passe chiffré (la connexion au serveur LDAP étant anonyme).
Il faut donc reconfigurer ces services manuellement (en modifiant le fichier /etc/pam.d/<*service*>) pour employer également le module pam_ldap.so.
L'autre solution est de positionner la variable `rootbinddn` dans /etc/libnss-ldap.conf, ce qui permet au moins aux processus disposant des droits root de récupérer les mots de passe chiffrés via NSS.

Sécuriser les échanges de données LDAP

LDAP est par défaut transporté en clair sur le réseau, ce qui signifie que les mots de passe chiffrés circulent sans précaution particulière. Repérables, ils peuvent donc subir une attaque de type dictionnaire. Pour éviter ce désagrément, il convient d'employer une couche supplémentaire de chiffrement, et cette section détaille comment procéder.

Configuration côté serveur

La première étape consiste à créer une paire de clés publique et privée pour LDAP. Pour cela, il faut installer le paquet *openssl*. On peut ensuite exécuter la commande **`/usr/lib/ssl/misc/CA.pl -newcert`**, qui pose plusieurs questions banales (lieu, nom de l'organisation, etc.). Il est impératif de répondre à la question « *Common Name* » le nom complet du serveur LDAP ; en l'occurrence il s'agit donc de `ldap.falcot.com`.

La commande précédente a généré un certificat complet dans le fichier newreq.pem. Il faut maintenant séparer la clé publique de la clé privée avec la commande **`openssl rsa -in newreq.pem -out newkey.pem`** puis en supprimant dans le fichier newreq.pem le bloc `RSA PRIVATE KEY`.

Il reste à installer ces clés dans un emplacement standard :

```
# mv newkey.pem /etc/ssl/private/ldap-key.pem
# chmod 0600 /etc/ssl/private/ldap-key.pem
# mv newreq.pem /etc/ssl/certs/ldap-cert.pem
```

Indiquons à **slapd** qu'il doit employer ces clés dans le cadre du chiffrement. Pour cela, il faut ajouter les directives ci-dessous au fichier /etc/ldap/slapd.conf :

EXEMPLE **Configuration de slapd pour la prise en charge du chiffrement**

```
# TLS support
TLSCipherSuite HIGH
TLSCertificateFile /etc/ssl/certs/ldap-cert.pem
TLSCertificateKeyFile /etc/ssl/private/ldap-key.pem
```

La dernière étape pour activer la mise en place le chiffrement est de modifier la variable SLAPD_SERVICES du fichier /etc/default/slapd. Notons au passage que pour éviter tout risque, on désactive la possibilité de LDAP non sécurisé.

EXEMPLE **Fichier /etc/default/slapd**

```
# Default location of the slapd.conf file
SLAPD_CONF=

# System account to run the slapd server under. If empty the server
# will run as root.
SLAPD_USER=

# System group to run the slapd server under. If empty the server will
# run in the primary group of its user.
SLAPD_GROUP=

# Path to the pid file of the slapd server. If not set the init.d script
# will try to figure it out from $SLAPD_CONF (/etc/ldap/slapd.conf)
SLAPD_PIDFILE=

# Configure if the slurpd daemon should be started. Possible values:
# - yes:   Always start slurpd
# - no:    Never start slurpd
# - auto:  Start slurpd if a replica option is found in slapd.conf
# (default)
SLURPD_START=auto

# slapd normally serves ldap only on all TCP-ports 389. slapd can also
# service requests on TCP-port 636 (ldaps) and requests via unix
# sockets.
# Example usage:
SLAPD_SERVICES="ldaps:/// ldapi:///"

# Additional options to pass to slapd and slurpd
SLAPD_OPTIONS=""
SLURPD_OPTIONS=""
```

Configuration côté client

Côté client, il faut modifier la configuration des modules *libpam-ldap* et *libnss-ldap* en ajoutant la directive `ssl on` aux deux fichiers de configuration /etc/pam_ldap.conf et /etc/libnss-ldap.conf.

Les clients LDAP doivent également pouvoir authentifier le serveur en connaissant sa clé publique. C'est pourquoi il est nécessaire d'en installer une copie (par exemple dans le fichier /etc/ssl/certs/ldap-cert.pem) et de la référencer depuis le fichier /etc/ldap/ldap.conf pour indiquer son existence.

EXEMPLE **Fichier /etc/ldap/ldap.conf**

```
# $OpenLDAP: pkg/ldap/libraries/libldap/ldap.conf,v 1.9 2000/09/04
# 19:57:01 kurt Exp $
#
# LDAP Defaults
#

# See ldap.conf(5) for details
# This file should be world readable but not world writable.

BASE   dc=falcot,dc=com
URI    ldaps://ldap.falcot.com

#SIZELIMIT      12
#TIMELIMIT      15
#DEREF          never

TLS_CACERT /etc/ssl/certs/ldap-cert.pem
```

Le tour d'horizon des logiciels serveurs proposé par ce chapitre est loin d'être exhaustif mais il recouvre toutefois la réalité des services réseau les plus employés. Passons sans plus tarder au pendant des serveurs dans un réseau : les stations clientes — en l'occurrence les machines « bureautiques ».



12



Station de travail

Les divers déploiements concernant les serveurs maintenant achevés, les administrateurs peuvent se charger des stations de travail individuelles et créer une configuration type.

SOMMAIRE

- Configuration de XFree86
- Personnalisation de l'interface graphique
- Bureaux graphiques
- Outils
- L'émulation Windows : Wine, VMWare, VNC, QEMU...

MOTS-CLEFS

- Station de travail
- Bureau graphique
- Bureautique
- XFree86

Configuration de XFree86

La phase de configuration initiale de l'interface graphique est toujours un peu délicate ; il arrive fréquemment qu'une carte vidéo très récente ne fonctionne pas avec la version de XFree86 livrée dans la version stable de Debian. Par ailleurs, il n'est pas toujours facile de choisir le bon pilote.

Rappelons que XFree86 est la brique logicielle de base qui permet aux applications graphiques d'afficher leur fenêtre sur l'écran. Il inclut le pilote de la carte graphique qui permet d'en tirer le meilleur parti, mais aussi une interface standardisée (*X11R6*) pour les fonctionnalités mises à disposition des applications graphiques.

Détection automatique

La configuration de XFree86 se gère par une interface **`debconf`** associée au paquet *xserver-xfree86*. Il s'agit du serveur X générique exploité par les versions 4.x de XFree86. Ce serveur modulaire dispose d'une collection de pilotes pour gérer les différents modèles de carte vidéo.

> COMPLÉMENTS **Autres serveurs X**
>
> Dans quelques situations très rares, le serveur X à employer n'est pas `xserver-xfree86` ; c'est notamment le cas si le serveur X de la carte vidéo est fourni par le fabricant lui-même ou pour les cartes vidéo tellement anciennes que le seul pilote encore disponible est un serveur X datant de la version 3.x de XFree86. L'emploi de ces serveurs peut provoquer d'autres problèmes, notamment au niveau de la syntaxe du fichier de configuration de XFree86.

Lors de sa première exécution, le script **`debconf`** essaie de deviner les valeurs adéquates pour les différents paramètres de configuration, mais par la suite, et comme tous les scripts **`debconf`**, il propose la dernière valeur sélectionnée. Pour déterminer ces valeurs, il exploite les utilitaires **`discover`**, **`mdetect`** et **`read-edid`** (ce dernier existe uniquement sur architecture i386). Si ces programmes n'étaient pas disponibles lors de la première invocation du script, les valeurs sélectionnées par défaut seront des valeurs passe-partout aux performances limitées (cas du pilote VESA pour la carte graphique) ou un choix arbitraire de la configuration la plus probable (souris PS/2 de base). C'est pourquoi il peut être intéressant d'exécuter manuellement les programmes d'autodétection pour repérer les valeurs les plus probables à utiliser si après la première configuration l'interface graphique ne démarre pas.

La valeur la plus importante est sans nul doute celle qui indique le pilote vidéo à utiliser. On l'obtient aisément avec la commande **`discover --data-path=xfree86/server/device/driver display`**. On pourra aussi détecter automatiquement la souris avec **`mdetect -x`**. Pour une meilleure détection des souris PS/2 et série, il est recommandé de déplacer l'engin pendant l'exécution du programme **`mdetect`**. Enfin, les caractéristiques de l'écran seront

données par le biais du protocole DDC (s'il est géré par votre carte graphique et par votre écran) ; vérifiez-le en exécutant **`get-edid | parse-edid`**. Si votre sous-ensemble vidéo ne comprend pas DDC, vous en serez averti par des messages d'erreur sur la console (voir exemple ci-après).

```
# discover --data-path=xfree86/server/device/driver display
ati
# mdetect -x
/dev/psaux
PS/2
# get-edid | parse-edid >config-ecran.txt
[ ... ]
parse-edid: EDID checksum failed - data is corrupt. Continuing anyway.
parse-edid: first bytes don't match EDID version 1 header
parse-edid: do not trust output (if any).
```

Script de configuration

Avant de débuter la configuration, il est bon d'avoir à proximité — en plus des informations auto-détectées — les manuels de la carte vidéo et de l'écran pour y contrôler les résolutions et les fréquences de rafraîchissement maximales possibles.

COMPLÉMENTS **Pilote propriétaire**

Certains fabricants de cartes graphiques (comme nVidia) refusent de donner les spécifications nécessaires à la création de bons pilotes libres. En revanche, ils fournissent des pilotes propriétaires qui permettent malgré tout d'employer leur matériel. Cette politique est à combattre car le pilote fourni — s'il existe — est souvent de moins bonne qualité, et surtout ne suit pas les mises à jour de XFree86, ce qui peut vous empêcher d'utiliser la dernière version disponible. Je ne peux que vous encourager à boycotter de tels fabricants et à vous tourner vers des concurrents plus coopératifs.
Si vous êtes malgré tout le malheureux propriétaire de l'une de ces cartes, la documentation ci-dessous devrait vous permettre de tirer profit de votre matériel :
▸ http://guide.andesi.org/html/dnvidia.html

Pour reconfigurer l'interface graphique après l'installation initiale, il convient d'exécuter **`dpkg-reconfigure xserver-xfree86`**, qui produit une foule de questions et qui finit par régénérer le fichier de configuration /etc/X11/XF86Config-4. La plupart des valeurs par défaut conviennent, mais certaines méritent que l'on s'y attarde. Concernant le choix du serveur X à employer, on répondra `xserver-xfree86`. Pour le pilote de la carte vidéo, le choix le plus sage est de reprendre le pilote détecté par la commande **`discover`**. À défaut d'un pilote adéquat, on peut employer *vesa* qui convient pour la quasi-totalité des cartes vidéo du monde du PC. En règle générale, on trouve un pilote par constructeur ou *chipset* vidéo (le composant électronique central de la carte vidéo) : *ati* pour ATI, *mga* pour Matrox, *nv* pour nVidia, etc.

Configuration du clavier

Les questions portant sur le clavier proposent pour la plupart des réponses par défaut convenables. Pensez à préciser `pc105` si votre clavier possède les touches « Windows ». N'oubliez pas non plus d'indiquer qu'il s'agit d'un clavier azerty en précisant la disposition de touches (*keyboard layout* en anglais) `fr`.

Configuration de la souris

Les questions portant sur la souris devraient reprendre les réponses suggérées par **`mdetect`**. Le port de la souris est souvent */dev/psaux* ; son type est fréquemment *PS/2*. Les utilisateurs d'un noyau 2.6.x peuvent systématiquement répondre */dev/input/mice* et ImPS/2 à cette question. L'émulation du troisième bouton est recommandée pour les souris n'en comptant que deux. De même, la prise en charge de la molette peut être activé sans crainte même si la souris réelle n'en a pas.

COMPLÉMENTS **Souris USB**

Le fichier de configuration de XFree86 généré inclut systématiquement /dev/input/mice comme périphérique de souris supplémentaire (et optionnel). Cela prend en charge les souris USB.

Configuration de l'écran

Les dernières questions importantes concernent l'écran. Il faut préciser s'il s'agit ou non d'un écran LCD. Les utilisateurs d'écrans plats et de portables doivent répondre « oui » à cette question. Trois modes différents décrivent ensuite les caractéristiques de l'écran : « simple », « moyen » et « expert ». Le mode simple demande uniquement la taille réelle de l'écran et n'est disponible que pour les écrans cathodiques — les réglages déduits seront probablement sous-optimaux pour les écrans de bonne qualité. Le mode moyen permet de choisir dans une liste de résolutions et de fréquences de rafraîchissement celles qui conviennent au mieux pour l'écran. Le mode expert permet de spécifier séparément les intervalles de fréquences de rafraîchissement horizontale et verticale. Le mode « moyen » est le meilleur compromis entre simplicité de saisie et optimisation de la configuration de l'écran. Il faut ensuite choisir le couple résolution/fréquence de rafraîchissement le plus proche de la configuration maximum décrite dans le manuel de l'écran (il est désormais très rare que la carte vidéo soit le facteur limitant mais cela peut encore arriver ; c'est pourquoi il est bon de vérifier que la carte vidéo supportera également ce mode). Enfin, il faut indiquer les résolutions que le serveur X pourra utiliser. En spécifiant une résolution plus grande que la résolution maximale de l'écran, on obtient un écran virtuel dont seule une partie sera affichée. Il est en général recommandé de choisir la résolution maximum de l'écran ainsi que les résolutions inférieures prises en charge. Cela permet à certaines applications de changer la résolution de l'interface graphique à la volée

et d'exploiter une résolution plus adaptée. C'est le cas de certains jeux vidéo ou des programmes qui permettent de regarder la télévision en plein écran.

Personnalisation de l'interface graphique

Choix d'un gestionnaire d'écran (*display manager*)

L'interface graphique n'est qu'un espace d'affichage... Si on se contente d'y exécuter le serveur X, l'écran restera désespérément vide. C'est pourquoi on installe habituellement un gestionnaire d'écran (*display manager*) affichant un écran d'authentification de l'utilisateur et exécutant ensuite son bureau graphique habituel. Les principaux gestionnaires d'écrans sont *gdm* (*GNOME Display Manager*), *kdm* (*KDE Display Manager*) et *xdm* (*X Display Manager*). Les administrateurs de Falcot SA ont retenu **gdm** puisqu'il s'associe logiquement à GNOME, le bureau graphique retenu. Le fichier /etc/gdm/gdm.conf compte de nombreuses options de configuration, mais toutes sont bien commentées. La plupart portent des valeurs par défaut qui conviennent relativement bien au cas des PC de bureau de Falcot SA.

Les quelques changements effectués permettent d'améliorer l'aspect visuel de l'écran d'accueil et donnent la possibilité à l'utilisateur d'éteindre et de redémarrer la machine sans connaître le mot de passe administrateur. Ces changements se traduisent dans le fichier de configuration par les entrées suivantes :

```
Greeter=/usr/bin/gdmgreeter
SystemMenu=true
SecureSystemMenu=false
Welcome=Bienvenue sur %s
Use24Clock=true
UseCirclesInEntry=true
GraphicalTheme=happygnome
```

Choix d'un gestionnaire de fenêtres

Chaque bureau graphique étant accompagné de son propre gestionnaire de fenêtres, le choix du premier implique habituellement celui du second. GNOME emploie ainsi **metacity** tandis que KDE exploite **kwm** (*KDE Window Manager*). De même, XFce (présenté dans une prochaine section) dispose de **XFwm**. La philosophie Unix autorise toujours d'employer le gestionnaire de fenêtres de son choix, mais suivre les recommandations permet de profiter au mieux des efforts d'intégration effectués par chacun des projets.

Il se peut cependant que certains ordinateurs trop anciens supportent mal la lourdeur des bureaux graphiques ; dans ce cas, une configuration plus légère

B.A.-BA **Gestionnaire de fenêtres**

Fidèle à la tradition Unix de ne faire qu'une chose mais de la faire bien, le gestionnaire de fenêtres affiche les cadres des fenêtres des différentes applications en cours de fonctionnement, ce qui inclut les bordures et la barre de titre. Il offre donc également les fonctionnalités de réduction, restauration, maximisation et masquage des fenêtres. La plupart des gestionnaires de fenêtres gèrent également un menu qui s'obtient en cliquant d'une certaine manière sur le bureau, et qui permet de quitter la session, de démarrer de nouvelles applications, et parfois de changer de gestionnaire de fenêtres.

SPÉCIFICITÉ DEBIAN **Les choix (alternatives)**

La charte Debian définit un certain nombre de commandes standards capables d'effectuer une action prédéfinie. Ainsi, la commande **x-window-manager** invoque un gestionnaire de fenêtres. Au lieu d'affecter cette commande à un gestionnaire de fenêtres présélectionné, Debian permet à l'administrateur de l'associer au gestionnaire de son choix.
Chaque gestionnaire de fenêtres s'enregistre comme un choix valable pour **x-window-manager** et fournit une priorité associée. Celle-ci permet de sélectionner automatiquement le meilleur gestionnaire de fenêtres installé en l'absence d'un choix explicite de l'administrateur.
C'est le script **update-alternatives** qui est utilisé à la fois par les paquets pour s'enregistrer comme un choix et par l'administrateur pour modifier le logiciel sur lequel la commande symbolique pointe (**update-alternatives --config *commande-symbolique***). Chaque commande symbolique pointe en réalité vers un lien symbolique contenu dans le répertoire /etc/alternatives, modifié par la commande **update-alternatives** au gré des mises à jour et des requêtes de l'administrateur. Si un paquet fournissant un choix est désinstallé, c'est le choix de priorité suivante qui le remplace.
Toutes les commandes symboliques existantes ne sont pas explicitées par la charte Debian, et certains responsables de paquets Debian ont délibérément choisi d'employer ce mécanisme dans d'autres cas moins standards où il apportait une souplesse appréciable (citons par exemple **x-www-browser**, **www-browser**, **cc**, **c++**, **awk**, etc.).

peut être envisagée. Parmi les gestionnaires de fenêtres correspondant à cette description, citons WindowMaker (paquet **wmaker**), Afterstep, *fvwm2*, *icewm* ou encore *blackbox*. Dans ce cas, il peut être intéressant d'indiquer au système quel gestionnaire de fenêtres privilégier. Pour cela, il est possible de modifier le choix **x-window-manager** grâce à la commande **update-alternatives --config x-window-manager**.

Gestion des menus

Les bureaux modernes et de nombreux gestionnaire de fenêtres disposent de menus donnant la liste des applications accessibles à l'utilisateur. Pour avoir des menus à jour correspondant aux applications réellement disponibles, Debian a créé une base centrale où chaque nouvelle application s'enregistre. Chaque nouveau paquet installé s'ajoute dans cette base et ordonne au système de mettre à jour les différents menus. Cette infrastructure est offerte par le paquet **menu**.

Chaque paquet disposant d'une application à insérer dans le système de menus dépose un fichier dans le répertoire /usr/lib/menu. Ce fichier décrit les capacités de l'application (graphique ou non, etc.) et l'emplacement qui lui convient le mieux dans la hiérarchie. Le script de post-installation du même paquet appellera **update-menus** qui se chargera de mettre à jour tous les fichiers nécessaires — mais cette commande ne peut pas connaître tous les types de menus disponibles parmi les applications installées. Chaque paquet intégrant un tel menu dans l'une de ses applications doit donc fournir un fichier exécutable qui recevra en entrée les différents éléments composant le menu, à charge pour lui de transformer ces informations en éléments exploitables par l'application contenant le menu. Ces filtres sont installés dans le répertoire /etc/menu-methods.

POUR ALLER PLUS LOIN **Standardisation des menus**

Debian dispose de son propre système de menus, mais aussi bien GNOME que KDE ont développé leur propre solution pour gérer leurs menus respectifs. Les deux projets semblent décidés à standardiser le format de ces menus — ou plutôt des fichiers .desktop représentant les éléments des menus — sous l'égide du projet FreeDesktop.org.

▸ http://www.freedesktop.org/

Les développeurs Debian suivent ce projet de près, un support préalable est déjà intégré aux outils Debian et il est à peu près certain que le format définitif sera pleinement supporté.

L'administrateur peut également intervenir dans le processus pour influencer les menus générés. La première de ses prérogatives est de pouvoir supprimer un élément du menu même si le logiciel correspondant est installé. Il suffit pour cela qu'il place dans /etc/menu/ un fichier vide portant le nom du paquet dont il souhaite supprimer les entrées. Il peut également réorganiser le menu en changeant le nom de certaines de ses sections ou en en regroupant quelques-unes. Il dispose pour cela du fichier /etc/menu-methods/translate_menus (qui contient des exemples dans ses commentaires). Enfin, il peut ajouter des éléments au menu pour donner un accès à des programmes installés manuellement ou pour exécuter une commande de son choix (comme démarrer un navigateur web sur une page particulière). Ces éléments supplémentaires sont décrits par des fichiers /etc/menu/local.<*element*>, qui suivent le même format que tous les autres fichiers disponibles dans le répertoire /usr/lib/menu.

Bureaux graphiques

Le domaine des bureaux graphiques connaît deux grandes familles de logiciels : GNOME et KDE, tous deux très populaires. C'est un phénomène que l'on ne retrouve pas dans tous les domaines du logiciel libre ; les concurrents d'Apache ne sont ainsi que des serveurs web marginaux.

Cette diversité a une origine historique, KDE fut le premier projet de bureau graphique mais son choix de la bibliothèque graphique Qt ne convenait pas à tous. À l'époque, Qt n'était pas encore un logiciel libre et GNOME a rapidement démarré en optant pour la bibliothèque graphique GTK+. Depuis, les projets évoluent en parallèle, Qt est depuis devenu libre, mais ces deux projets n'ont pas fusionné.

Ils collaborent cependant : par l'intermédiaire de FreeDesktop.org, ils ont défini des normes favorisant l'interopérabilité entre les différentes applications.

Je ne m'aventure pas à répondre à l'épineuse question du choix du bureau graphique : ce chapitre passe rapidement en revue les différentes possibilités, et fournit des éléments de réflexion sur le sujet. Il est toujours préférable d'essayer les différentes possibilités avant d'en adopter une.

GNOME

Debian Sarge contient la version 2.8 de GNOME, qui s'installe simplement par la commande **`apt-get install gnome-desktop-environment`**.

GNOME est intéressant de par ses efforts dans le domaine de l'ergonomie et l'accessibilité. Des professionnels du *design* ont en effet rédigé des normes pour aider les développeurs à créer des interfaces graphiques satisfaisantes. Le projet est en outre encouragé par des grands acteurs de l'informatique comme Hewlett-Packard, Sun et Novell, sans oublier des distributions Linux. Enfin, un grand nombre de langages de programmation sont exploitables pour développer des applications s'intégrant à GNOME.

La réalisation de toute cette infrastructure a pris beaucoup de temps au projet GNOME, qui donne parfois l'impression d'une maturité moins aboutie que celle de KDE. L'ergonomie et l'accessibilité n'ont fait que récemment l'objet d'une attention particulière, et on n'en perçoit les bénéfices que depuis les dernières versions de l'environnement.

Pour les administrateurs, GNOME semble être mieux préparé à des déploiements massifs. La configuration des applications est gérée par *GConf*, une sorte de base de registres interrogeable et modifiable par l'utilitaire en ligne de commande **`gconftool-2`**. L'administrateur peut donc modifier la configuration des utilisateurs par un simple script. Le site web ci-dessous regroupe toutes les informations qui peuvent intéresser un administrateur en charge de stations employant GNOME :

- http://www.gnome.org/learn/admin-guide/latest/

KDE

La version 3.3 de KDE, intégrée à Debian Sarge, s'installe facilement avec la commande **`apt-get install kde`**.

KDE a évolué rapidement en suivant une approche très pragmatique ; ses auteurs ont très rapidement obtenu d'excellents résultats, ce qui leur a permis de mettre en place une importante base d'utilisateurs... contribuant elle-même à la qualité du projet. Globalement, KDE est un bureau graphique parfaitement mûr, disposant d'une très large palette d'applications.

Au rang des limitations, signalons que la bibliothèque Qt n'est libre que sous Unix et que sa version Windows reste soumise à des contraintes — KDE ne compte donc aucune application libre qui puisse être portée sous Windows. Cette limitation ne devrait cependant pas tarder à disparaître puisque les versions 4.x de Qt devraient également être disponibles sous licence GPL sous Windows. Notons enfin que le langage C++ est obligatoire pour développer une application KDE.

Xfce et autres

Xfce est un bureau graphique simple et allégé qui convient parfaitement aux ordinateurs limités en ressources. Il s'installe avec la commande **`apt-get install xfce4`**.

Contrairement à GNOME et KDE, Xfce ne fournit pas une importante famille d'applications. Il se contente d'intégrer quelques utilitaires indispensables :

- un gestionnaire de fichiers ;
- un gestionnaire de fenêtres ;
- un panneau démarreur d'applications ;

La page suivante résume quelques informations intéressantes pour un usage de Xfce en entreprise :

▸ http://www.xfce.org/index.php?page=documentation&lang=fr#corporate

Outils

Courrier électronique

Evolution

C'est le logiciel de messagerie de GNOME, qu'on installe avec la commande **`apt-get install evolution`**. En plus du courrier électronique, il gère un agenda, un carnet d'adresses et une liste de tâches, et dispose d'un puissant système d'indexation des messages. Il est possible de créer des dossiers virtuels correspondant à des requêtes sur l'ensemble des messages archivés. C'est-à-dire que tous les messages sont stockés de la même façon, mais que l'affichage des courriers électroniques recrée une simili-organisation par dossier — chacun de ces dossiers contenant tous les messages répondant à un ou plusieurs critères de filtrage.

COMMUNAUTÉ **Les paquets populaires**

En installant le paquet Debian *popularity-contest* vous pouvez participer à un sondage (automatique) qui permet au projet Debian de recenser les paquets les plus populaires. Un script lancé hebdomadairement par **`cron`** envoie par courrier électronique et de façon aussi anonyme que possible la liste des paquets installés ainsi que la date de dernier accès aux fichiers contenus dans chacun. Ce système permet de savoir quels paquets sont installés et lesquels sont réellement utilisés.
Ces informations sont extrêmement utiles au projet Debian, et lui permettent notamment de savoir quels paquets intégrer sur le premier cédérom d'installation. Il lui est aussi possible de vérifier si un paquet est employé ou non avant de décider de le supprimer de la distribution. C'est pourquoi je vous invite fortement à installer le paquet *popularity-contest* et à participer au sondage.
Les statistiques ainsi collectées sont publiées quotidiennement.
▸ http://popcon.debian.org
Ces statistiques peuvent éventuellement vous aider à choisir entre deux paquets qui vous semblent équivalents. En prenant le plus populaire, vous multipliez vos chances de faire un bon choix.

Une extension du logiciel permet de l'intégrer à une messagerie Microsoft Exchange : il s'agit de Ximian Connector. Le paquet Debian correspondant est *evolution-exchange*.

KMail

On installe le logiciel de messagerie de KDE en exécutant **`apt-get install kmail`**. Il se contente de gérer le courrier électronique mais fait partie d'un ensemble logiciel appelé « KDE-PIM » (*PIM* signifiant *Personal Information Manager*, ou gestionnaire des informations personnelles) qui regroupe évidemment les fonctionnalités de carnet d'adresses, d'agenda, etc. Kmail dispose de toutes les fonctionnalités requises pour être un excellent client de messagerie.

Thunderbird

Ce logiciel de messagerie, disponible dans le paquet Debian *mozilla-thunderbird*, fait partie de la suite logicielle du projet Mozilla. Sa localisation en français nécessite la présence du paquet *mozilla-thunderbird-locale-fr* ; l'extension *mozilla-thunderbird-enigmail* prend complètement en charge le chiffrement et la signature des messages.

Ce logiciel fait partie des meilleurs clients de messagerie électronique. Gageons qu'il connaîtra un beau succès — à l'instar de Mozilla Firefox.

Navigateurs web

Epiphany, le navigateur web développé par GNOME, utilise le moteur d'affichage Gecko développé pour Mozilla. On le trouve dans le paquet Debian *epiphany-browser*.

Konqueror, le navigateur web de KDE, utilise le moteur de rendu KHTML propre à cet environnement de bureau. Même s'il n'égale pas Gecko en termes de fonctionnalités et de respect des standards, KHTML est de très bonne facture et a été choisi par Apple pour réaliser son navigateur Safari. Son paquet Debian se nomme *konqueror*.

Les personnes satisfaites par aucun des deux navigateurs mentionnés ci-dessus pourront se tourner vers *Mozilla Firefox*. Ce navigateur, qu'on trouve dans le paquet Debian *mozilla-firefox*, allie la puissance de Gecko à une interface légère et extensible.

CULTURE **Mozilla**

Netscape Navigator était le navigateur emblématique du début du Web mais l'arrivée de *Microsoft Internet Explorer* l'a progressivement marginalisé. Face à cet échec, la société Netscape a décidé de « libérer » ses codes sources (en les publiant sous une licence libre) pour tenter de lui donner une seconde vie. C'était le début du projet Mozilla. Après de nombreuses années de développement, le résultat est plus que satisfaisant : le projet Mozilla est à l'origine du moteur de rendu HTML (nommé Gecko) le plus compatible avec les normes. Le navigateur Mozilla (paquet Debian *mozilla-browser*) regagne une petite partie du terrain perdu par Netscape. Par ailleurs, il a encadré la naissance de nombreux navigateurs alternatifs, dont certains — comme Mozilla Firefox — connaissent un franc succès et une croissance importante du nombre de leurs utilisateurs.

Développement

Outils pour GTK+ sur GNOME

Anjuta (du paquet Debian éponyme) est un environnement de développement permettant de créer des applications GTK+ pour GNOME. Glade (du paquet Debian éponyme) est un logiciel capable de créer des interfaces utilisateur avec GTK+ sous GNOME et de les enregistrer dans un fichier (au format XML). La bibliothèque partagée *libglade* permet alors de recréer dynamiquement les interfaces sauvegardées — fonctionnalité intéressante par exemple pour des greffons (*plugins*) ayant besoin d'une boîte de dialogue.

Anjuta2, projet beaucoup plus ambitieux, a pour objectif de conjuguer de façon modulaire toutes les fonctionnalités nécessaires à un environnement de développement intégré. Ce projet n'en est qu'à ses débuts et il ne faut rien en attendre avant encore de nombreux mois.

Outils pour Qt sur KDE

KDevelop (du paquet Debian *kdevelop3*) est un environnement de développement pour KDE. *Qt Designer* (du paquet Debian *kde-designer*) est un logiciel facilitant la conception d'interface graphique pour Qt sur KDE.

Les prochaines versions de ces logiciels promettent une meilleure intégration de l'un à l'autre grâce à la technologie de composants *KParts*.

Travail collaboratif

Travail en groupe : *groupware*

PHPGroupware est une application web regroupant de nombreuses fonctionnalités de travail collaboratif :

- messagerie électronique
- messagerie instantanée

- carnet d'adresses
- Wiki
- système de gestion de contenus
- base de connaissances
- annuaire de liens
- système de *chat* (discussion à plusieurs)
- gestion de projet (*workflow*)

▸ http://www.phpgroupware.org

ALTERNATIVE **eGroupware**

eGroupware est une solution qui mérite d'être prise en compte.

▸ http://www.egroupware.org

Le paquet *egroupware* installe tous les modules disponibles. Si vous voulez n'installer que certains modules, installez d'abord le paquet *egroupware-core* puis installez les modules retenus (exemple : *egroupware-calendar* et *egoupware-email*). La commande **`apt-cache search egroupware-`** permet de découvrir tous les modules existants.

DÉCOUVERTE **Vidéoconférence avec GnomeMeeting**

GnomeMeeting est l'application phare du monde de la vidéoconférence sous Linux. Logiciel stable et fonctionnel, il s'emploie facilement et sans restrictions sur un réseau local, mais il est beaucoup plus difficile de faire fonctionner le service à travers un pare-feu qui ne gère pas explicitement le protocole de téléconférence H323 et ses subtilités.

Si l'on souhaite placer un seul client GnomeMeeting derrière le pare-feu, on peut se contenter de *forwarder* (faire suivre) quelques ports sur la machine dédiée à la vidéoconférence : le port 1720 en TCP (port d'écoute des connexions entrantes), les ports 30000-30010 en TCP (pour le contrôle des connexions ouvertes) et les ports 5000-5013 en UDP (pour les transmissions audio et vidéo, et l'enregistrement dans un proxy H323).

Si l'on souhaite placer plusieurs clients GnomeMeeting derrière le pare-feu, les choses se compliquent sérieusement. Il faut alors installer un « proxy H323 » (paquet *gatekeeper*), dont la configuration n'est pas triviale.

Messagerie instantanée

Pour mettre en place une messagerie instantanée interne à l'entreprise, l'emploi de Jabber s'impose : son protocole est standardisé, et il ne démérite pas en termes de fonctionnalités. D'autre part, la possibilité d'y chiffrer les échanges est appréciable. Enfin, il est possible de placer des passerelles entre un serveur Jabber et les autres réseaux de messagerie instantanée (ICQ, GAIM, Yahoo, MSN, etc.).

ALTERNATIVE **Internet Relay Chat**

L'IRC peut remplacer Jabber. Ce service gravite autour de la notion de canal (dont les noms débutent systématiquement par le signe dièse #) : chaque canal regroupe un ensemble de personnes autour d'un thème ou d'une habitude de groupe. Au besoin, des personnes peuvent converser en privé. Son protocole, plus ancien, n'offre pas la possibilité de sécuriser les échanges.

Les clients IRC, plus difficiles à maîtriser, offrent beaucoup de fonctionnalités peu utiles en entreprise. Les opérateurs sont des utilisateurs dotés du pouvoir d'exclure voire de bannir les utilisateurs indésirables pour préserver le calme au sein du canal.

Configuration du serveur

La mise en place du serveur Jabber demandant quelques efforts, elle est détaillée ci-dessous.

La première étape consiste à installer les paquets nécessaires en exécutant la commande **`apt-get install jabber jabber-muc jabber-jud`**. Le module *jabber-muc* permet les discussions à plusieurs (*Multi User Chat*) tandis que *jabber-jud* propose un annuaire des utilisateurs (*Jabber User Directory*).

Le premier fichier de configuration à modifier est /etc/jabber/jabber.xml. Toutes les références à `localhost` doivent y être remplacées par le nom officiel du serveur (en l'occurrence, il s'agit de `jabber.falcot.com`). On personnalisera les informations de *vCard* (« carte de visite ») du serveur et modifiera ses messages textuels standards — notamment la notification d'enregistrement et le message de bienvenue.

On corrigera le nom de machine (variable `JABBER_HOSTNAME`) dans le fichier /etc/jabber/jabber.cfg.

L'intégration des modules *jabber-muc* et *jabber-jud* relève de la même démarche, documentée dans les fichiers /usr/share/doc/jabber-muc/README.Debian et /usr/share/doc/jabber-jud/README.Debian. Il s'agit d'ajouter une référence et un bloc *service* au fichier jabber.xml. Il faut ensuite activer le module en modifiant les fichiers /etc/default/jabber-muc et /etc/default/jabber-jud et en positionnant la variable `ENABLED` à `1`. Enfin, il faut personnaliser les fichiers /etc/jabber/jabber-muc.xml et /etc/jabber/jabber-jud.xml de manière concordante (les mots de passe indiqués dans les balises `<secret>` et les identifiants de service doivent correspondre au contenu du fichier jabber.xml).

EXEMPLE Extraits du fichier `jabber.xml`

```
<host><jabberd:cmdline flag="h">jabber.falcot.com</jabberd:cmdline></host
  >
[...]
<vCard>
 <FN>Falcot Jabber Server</FN>
 <DESC>Serveur Jabber interne (Falcot SA)</DESC>
 <URL>http://www.falcot.com/</URL>
</vCard>
[...]
<register notify="yes">
  <instructions>Choisissez un identifiant (pas
   trop fantaisiste) et un mot de passe pour vous enregistrer
   sur ce serveur.</instructions>
  <name/>
  <email/>
</register>
[...]
<welcome>
  <subject>Bienvenue!</subject>
  <body>Bienvenue sur le serveur Jabber. Consultez
   http://intranet.falcot.com/jabber/ pour en savoir
   plus.</body>
</welcome>
[...]
<browse>
[...]
 <service type="jud" jid="jud.jabber.falcot.com" name="Falcot Jabber User
    Directory">
  <ns>jabber:iq:search</ns>
  <ns>jabber:iq:register</ns>
 </service>
[...]
</browse>
[...]
<!-- OK, we've finished defining the Jabber Session Manager. -->

<service id="muclinker">
  <host>conference.jabber.falcot.com</host>
  <accept>
```

```
    <ip>127.0.0.1</ip>
    <port>31518</port>
    <secret>secret</secret>
  </accept>
</service>

<service id="jud">
 <host>jud.jabber.falcot.com</host>
 <accept>
   <ip>127.0.0.1</ip>
   <port>5559</port>
   <secret>someSecret</secret>
   <timeout>30</timeout>
 </accept>
</service>
```

ATTENTION **Les JID sont uniques**

Lorsque l'on configure les différents modules Jabber sur la même machine que le serveur Jabber, on est tenté de donner le même nom aux différents services (après tout, le numéro de port devrait permettre de les différencier). Cela ne fonctionne pas, et chaque service doit recevoir un JID différent qui soit un nom DNS valide (et pointant sur la machine qui héberge le serveur Jabber). C'est pourquoi *conference.jabber.falcot.com*, *jud.jabber.falcot.com* et *jabber.falcot.com* pointent tous trois sur la même machine.

Clients Jabber

GNOME dispose de Gossip (du paquet Debian éponyme), il s'agit d'un client très simple d'emploi qui s'intègre dans la zone de notification du tableau de bord (présent par défaut en haut de l'écran si vous utilisez GNOME).

KDE dispose quant à lui de Kopete (paquet Debian éponyme).

Travail collaboratif avec GForge

GForge est un outil de développement collaboratif. Historiquement, il dérive de Sourceforge, service d'hébergement en ligne de projets logiciels libres. Il en garde l'approche basée sur le mode de développement du logiciel libre, et a continué à évoluer après que le code de Sourceforge a été rendu propriétaire (i.e. les détenteurs des droits — VA Software — ont décidé de ne plus le diffuser sous une licence libre). Il fournit donc également quelques fonctionnalités mieux adaptées à un mode de fonctionnement plus traditionnel, ainsi qu'à des activités qui ne relèvent pas du développement pur.

GForge est en réalité une agglomération d'un ensemble d'outils permettant de gérer, suivre et animer des projets. Ces outils relèvent de trois grandes catégories :

- *communication* : forums de discussion sur le Web, gestionnaire de listes de diffusion par messagerie électronique, systèmes de nouvelles permettant à un projet de publier des « brèves » ;

- *suivi* : gestionnaire de tâches permettant le contrôle de leur progrès et leur ordonnancement, « pisteurs » (*tracker*) pour le suivi des bogues, des correctifs et des demandes d'amélioration, sondages ;
- *partage* : gestionnaire de documentation permettant de centraliser les documentations pour un projet, mise à disposition de fichiers génériques, espace web dédié à chaque projet.

À cela, s'ajoute l'intégration de CVS (*Concurrent Versions System*, système de gestion de sources, ou de gestion de configuration ou de suivi de versions — les appellations sont nombreuses). Ce programme conserve un historique des différentes versions par lesquelles est passé chaque fichier (il s'agit fréquemment de codes sources de programmes), conserve une trace de chaque changement, et fusionne les modifications apportées indépendamment par plusieurs développeurs lorsqu'ils travaillent en même temps sur la même partie d'un projet.

La plupart de ces outils sont accessibles (voire gérés) par une interface web, avec un système de gestion de permissions assez fin, et des notifications par courrier électronique pour certains événements.

Suites bureautiques

Les logiciels de bureautique furent longtemps les parents pauvres du logiciel libre : les utilisateurs demandent des équivalents aux outils de Microsoft (Word ou Excel), mais ces logiciels sont si riches fonctionnellement qu'il était difficile de développer des équivalents. La création du projet OpenOffice.org (grâce à la libération du code de StarOffice par Sun) a comblé cette lacune. Les efforts respectifs de GNOME (GNOME Office) et KDE (KOffice) se poursuivent et parviennent — peut-être grâce à l'existence d'une saine concurrence — à des résultats intéressants. Ainsi Gnumeric (le tableur de GNOME) surpasse même OpenOffice.org dans certains domaines (la précision des calculs notamment). En revanche, du côté des traitements de texte, celui de la suite OpenOffice.org reste encore la référence.

Par ailleurs, il est important pour les utilisateurs de pouvoir exploiter les documents Word et Excel qu'ils reçoivent et qui peuplent leurs archives. Même si tous ont des filtres de prise en charge des logiciels de Microsoft, seul OpenOffice.org atteint un niveau suffisant.

Le paquet Debian d'OpenOffice.org se nomme *openoffice.org*, celui de KOffice *koffice*, et celui de GNOME *gnome-office*. Pour bénéficier d'une francisation complète d'OpenOffice.org, il faudra en outre installer les paquets *openoffice.org-l10n-fr*, *openoffice.org-help-fr*, et *openoffice.org-spellcheck-fr-fr* (ce dernier est un paquet virtuel fourni par *myspell-fr-gut*).

L'émulation Windows : Wine, VMWare, VNC, QEMU...

Quels que soient les efforts fournis sur tous les plans précédents, on trouve toujours tel ou tel outil particulier sans équivalent connu sous Linux ou dont il est absolument nécessaire d'employer la version originale. C'est pourquoi les systèmes d'émulation de Windows sont intéressants, et c'est précisément le rôle du logiciel Wine.

▸ http://www.winehq.com

COMPLÉMENT **CrossOver Office**

La société CodeWeavers a amélioré Wine en créant *CrossOver Office* — ce dernier permettrait d'employer Microsoft Office sans soucis grâce à une palette plus large de fonctionnalités émulées. Certaines de ses améliorations sont réintégrées à Wine.

▸ http://www.codeweavers.com/site/products/

Mais il serait regrettable de se limiter à son étude, alors même que cette problématique peut être résolue de manière élégante avec d'autres outils comme une machine virtuelle ou bien encore VNC, tous deux présentés dans les encadrés ci-contre.

Rappelons au passage qu'une émulation permet d'exécuter un programme développé pour un autre système en imitant celui-ci grâce à un logiciel (qui en simule donc les fonctionnalités du mieux qu'il peut à partir des possibilités du système hôte sur lequel il s'exécute).

Le plus simple pour utiliser Wine est de disposer d'une partition comportant déjà une installation de Microsoft Windows (ce qui est le cas pour toute machine sur laquelle Linux est installé en double amorçage avec ce système Windows). Cette partition doit utiliser le système de fichiers FAT et non pas NTFS, car Linux ne peut que lire ce dernier (et pas y écrire de façon sûre et fiable — ce qui est pourtant nécessaire pour le fonctionnement optimal de la plupart des applications Windows). Si la machine ne dispose pas d'une version fonctionnelle de Windows, Wine fonctionnera un peu moins bien, et sera capable d'encadrer moins de programmes.

Il faut au préalable s'assurer que la partition Windows est montée (par exemple sous /windows) et accessible en lecture/écriture à l'utilisateur qui emploie **wine**. L'entrée de fichier fstab ci-dessous donne cet accès en écriture à tous les utilisateurs :

```
/dev/hda1 /windows fat defaults,uid=1000,gid=100,umask=002,nls=iso8859-
  1 0 0
```

Installons tous les paquets nécessaires :

```
# apt-get install wine winesetuptk msttcorefonts libwine-print wine-
  utils wine-doc
```

Il reste ensuite à l'utilisateur à exécuter **winesetup** et accepter les réglages par défaut. Il pourra ensuite exécuter les programmes Windows en exécutant simplement la commande **wine *`/windows/.../program.exe`***.

Wine fonctionne relativement bien avec Windows 95/98/Me et les applications de cette génération, mais pose encore quelques problèmes pour Windows NT/2000/XP.

Avant de compter sur Wine ou des solutions similaires, il faut savoir que rien ne remplace le test réel d'une version d'un logiciel : lui seul assure un fonctionnement satisfaisant avec une solution d'émulation.

ALTERNATIVE **Les émulateurs VMWare, Bochs, QEMU**

VMWare propose une machine virtuelle compatible PC où on peut démarrer Microsoft Windows (ou tout autre système d'exploitation sur architecture Intel). Il s'agit cependant d'un logiciel propriétaire et commercial — il n'est donc pas présent dans Debian.

▸ http://www.vmware.com

Une machine virtuelle, contrairement aux émulateurs traditionnels, est un logiciel émulant du matériel et non pas un système d'exploitation. Cela augmente la fiabilité de l'émulation puisque le problème de fonctionnalités non émulées n'existe presque plus, mais cela a un coût non négligeable en termes de performances.

Le succès de *VMWare* parmi les utilisateurs de Linux a conduit à l'apparition de solutions libres : citons Bochs et QEMU. D'un point de vue technique, les 3 outils ne fonctionnent pas exactement selon le même principe mais le résultat concret est le même. Leur niveau de maturité diffère fortement aussi… les performances ne sont pas systématiquement au rendez-vous, étant donné qu'il s'agit d'émulation de machines.

ALTERNATIVE **Windows Terminal Server ou VNC**

Une dernière solution est à considérer : les applications Windows à conserver peuvent être installées sur un serveur central *Windows Terminal Server* et exécutées à distance par des machines Linux employant *rdesktop*. Ce programme est un client Linux qui comprend le protocole RDP (*Remote Desktop Protocol*, protocole de bureau distant) employé par *Windows NT/2000 Terminal Server* pour déporter des bureaux graphiques.

Le logiciel VNC offre des fonctions similaires et fonctionne en outre avec de nombreux systèmes d'exploitation. Les clients et serveurs VNC pour Linux sont traités dans la section « Connexion à distance » du chapitre 9.

13



Conception d'un paquet Debian

Manipuler régulièrement des paquets Debian provoque tôt ou tard le besoin de créer le sien propre ou d'en modifier un. Ce chapitre essaie de répondre à vos interrogations en la matière et fournit des éléments pour tirer le meilleur parti de l'infrastructure offerte par Debian. Qui sait, en mettant ainsi le pied à l'étrier, peut-être irez-vous plus loin et deviendrez-vous développeur Debian !

SOMMAIRE

- Recompiler un paquet depuis ses sources
  - Récupérer les sources
  - Effectuer les modifications
  - Démarrer la recompilation
- Construire son premier paquet
  - Méta-paquet ou faux paquet
  - Simple archive de fichiers
- Créer une archive de paquets pour APT
- Devenir mainteneur de paquet
  - Apprendre à faire des paquets
  - Processus d'acceptation

MOTS-CLEFS

- Rétroportage
- Recompilation
- Paquet source
- Archive
- Méta-paquet
- Développeur Debian
- Mainteneur

Recompiler un paquet depuis ses sources

Plusieurs éléments peuvent justifier la recompilation d'un paquet depuis ses sources. L'administrateur peut avoir besoin d'une fonctionnalité du logiciel qui implique de recompiler le programme en activant une option particulière ou souhaiter en installer une version plus récente que celle fournie dans sa version de Debian (dans ce cas, il recompilera un paquet plus récent récupéré dans la version *testing* ou *unstable* pour qu'il fonctionne parfaitement dans sa distribution *stable*, opération appelée le « rétroportage »).

Récupérer les sources

Pour recompiler un paquet Debian, il faut commencer par rapatrier son code source. Le moyen le plus simple est d'employer la commande **`apt-get source` *`nom-paquet-source`***, qui nécessite la présence d'une ligne de type `deb-src` dans le fichier `/etc/apt/sources.list` et l'exécution préalable de la commande **`apt-get update`**. C'est déjà le cas si vous avez suivi les instructions du chapitre portant sur la configuration d'APT (voir page 82). Notez cependant que vous téléchargerez les paquetages sources du paquet disponible dans la version de Debian désignée par la ligne `deb-src` de ce fichier de configuration. Si vous souhaitez en rapatrier une version particulière, il vous faudra peut-être la télécharger manuellement depuis un miroir Debian ou depuis le site web : récupérer deux ou trois fichiers (d'extensions `*.dsc` (*Debian Source Control*), `*.tar.gz`, et `*.diff.gz` — ce dernier n'existe que si l'archive `.tar.gz` contient en fait l'extension `.orig.tar.gz`) puis exécuter la commande **`dpkg-source -x` *`fichier.dsc`***.

Effectuer les modifications

Les sources du paquet maintenant disponibles dans un répertoire portant le nom du paquet source et sa version (ex : *samba-3.0.2*), nous pouvons nous y rendre pour y effectuer nos modifications.

La première modification à apporter est de changer le numéro de version du paquet pour distinguer les paquets recompilés des paquets originaux fournis par Debian. Supposons que la version actuelle soit `3.0.2-2`; nous pouvons créer une version `3.0.2-2.falcot1`, ce qui désigne clairement l'origine du paquet. De cette manière, la version du paquet est supérieure à celle fournie par Debian et le paquet s'installera facilement en tant que mise à jour du paquet original. Pour effectuer ce changement, il est préférable d'utiliser le programme **`dch`** (*Debian CHangelog*) du paquet *devscripts* en saisissant **`dch -v 3.0.2-2.falcot1`**. Cette commande démarre un éditeur de texte (**`sensible-editor`** — votre éditeur favori si vous l'avez précisé dans la variable d'environnement `VISUAL` ou `EDITOR`, un éditeur par défaut dans les autres cas) où l'on pourra documenter

les différences apportées par cette recompilation. On peut constater que **dch** a bien modifié le fichier debian/changelog.

Si une modification des options de compilation s'avère nécessaire, il faudra modifier le fichier debian/rules, qui pilote les différentes étapes de la compilation du paquet. Vous repérerez facilement les lignes concernant la configuration initiale (`./configure ...`) ou déclenchant la compilation (`$(MAKE) ...` ou `make ...`). En les adaptant convenablement, il est possible d'obtenir l'effet souhaité.

Il convient parfois de s'occuper du fichier debian/control, qui renferme la description des paquets générés. Il peut être intéressant de la modifier pour qu'elle reflète les changements apportés. Par ailleurs, ce fichier contient aussi des champs `Build-Depends` qui donnent la liste des dépendances de génération du paquet. Celles-ci se rapportent souvent à des versions de paquets contenus dans la distribution d'origine du paquet source, qui ne sont peut-être pas disponibles dans la version utilisée pour la recompilation. Il n'existe pas de moyen automatique pour savoir si une dépendance est réelle ou si elle a été créée pour garantir que la compilation s'effectue bien avec les dernières versions d'une bibliothèque (c'est le seul moyen disponible pour forcer un *autobuilder* à recompiler le paquet avec une version prédéfinie d'un paquet — c'est pourquoi les mainteneurs Debian utilisent fréquemment ce procédé).

N'hésitez donc pas à modifier ces dépendances pour les assouplir si vous savez qu'elles sont trop strictes. La lecture d'éventuels fichiers documentant le mode de compilation du logiciel (souvent nommés INSTALL) vous sera sans doute utile pour retrouver les bonnes dépendances. Idéalement, il faudrait satisfaire toutes les dépendances avec les paquets disponibles dans la version utilisée pour la recompilation. Sans cela, on entre dans un processus récursif où il faut préalablement rétroporter les paquets donnés dans les champs `Build-Depends` avant de pouvoir compléter le rétroportage souhaité. Certains paquets, qui n'ont pas besoin d'être rétroportés, peuvent être installés tels quels pour les besoins de la recompilation (c'est souvent le cas de *debhelper*). Cependant, le processus peut se compliquer rapidement si l'on n'y prend garde, aussi faut-il éviter autant que possible tout rétroportage non strictement nécessaire. Avant de vous lancer dans une telle opération, vérifiez aussi que personne d'autre ne l'a menée avant vous (en particulier sur le site `apt-get.org`).

ASTUCE **Installer les `Build-Depends`**

`apt-get` permet d'installer rapidement tous les paquets cités dans le ou les champs `Build-Depends` d'un paquet source disponible dans une distribution donnée sur une ligne `deb-src` du fichier /etc/apt/sources.list. Il suffit pour cela d'exécuter la commande **`apt-get build-dep`** ***`paquet-source`***.

Démarrer la recompilation

Toutes les modifications souhaitables étant apportées sur les sources, il faut maintenant régénérer le paquet binaire correspondant (fichier .deb). Ce processus nécessite théoriquement les droits root à différentes étapes, mais pour des raisons de sécurité l'utilitaire nommé **`fakeroot`** permet de s'en passer — nous y recourrons donc. Tout ce processus de création est contrôlé par le programme **`dpkg-buildpackage`**.

DÉCOUVERTE **pbuilder**

Le programme **pbuilder** (du paquet éponyme) permet de recompiler un paquet Debian dans un environnement *chrooté* : il crée un répertoire temporaire contenant un système minimal nécessaire à le reconstruction du paquet (en se basant sur les informations contenues dans le champ *Build-Depends*). Grâce à la commande **chroot**, ce répertoire sert ensuite de racine (/) lors du processus de recompilation.
Cette technique permet de compiler le paquet dans un environnement non dégradé (notamment par les manipulations des utilisateurs), de détecter rapidement les manques éventuels dans les dépendances de compilation (qui échouera si un élément essentiel n'est pas documenté) et de compiler un paquet pour une version de Debian différente de celle employée par le système (la machine peut utiliser *stable* pour le fonctionnement quotidien et **pbuilder** peut employer *unstable* pour la recompilation).

EXEMPLE **Recompilation d'un paquet**

```
$ dpkg-buildpackage -rfakeroot -us -uc
[...]
```

La commande précédente peut échouer si les champs `Build-Depends` n'ont pas été corrigés ou si les dépendances correspondantes n'ont pas été installées. Dans ce cas, on peut outrepasser cette vérification en ajoutant le paramètre **-d** à l'invocation de **dpkg-buildpackage**. En ignorant volontairement ces dépendances, on s'expose cependant à ce que la compilation échoue plus tard. Pis, il se peut que le paquet compile correctement mais que son fonctionnement soit altéré car certains programmes désactivent automatiquement des fonctionnalités s'ils détectent l'absence d'une bibliothèque lors de la compilation.

Les développeurs Debian utilisent plus volontiers un programme comme **debuild** qui fera suivre l'appel de **dpkg-buildpackage** par l'exécution d'un programme chargé de vérifier que le paquet généré est conforme à la charte Debian. Par ailleurs, ce script nettoie l'environnement pour que les variables d'environnement locales n'affectent pas la compilation du paquet. **debuild** fait partie de la série d'outils du paquet *devscripts*, qui partagent une certaine cohérence et une configuration commune simplifiant le travail des mainteneurs.

Construire son premier paquet

Méta-paquet ou faux paquet

Faux paquet et méta-paquet se concrétisent tous deux par un paquet vide qui n'existe que pour les effets de ses informations d'en-têtes sur la chaîne logicielle de gestion des paquets.

Le faux paquet existe pour tromper **dpkg** et **apt** en leur faisant croire que le paquet correspondant est installé alors qu'il ne s'agit que d'une coquille vide. Cela permet de satisfaire les dépendances lorsque le logiciel en question a été installé manuellement. Cette méthode fonctionne, mais il faut l'éviter autant que possible ; rien ne garantit en effet que le logiciel installé manuellement constitue un remplaçant parfait du paquet concerné, et certains autres paquets, qui en dépendent, pourraient donc ne pas fonctionner.

Le méta-paquet existe en tant que collection de paquets par le biais de ses dépendances, que son installation installera donc toutes.

Pour créer ces deux types de paquets, on peut recourir aux programmes **equivs-control** et **equivs-build** (du paquet Debian *equivs*). La commande **equivs-control *fichier*** crée un fichier contenant des en-têtes de paquet Debian qu'on modifiera pour indiquer le nom du paquet souhaité, son numéro de version, le nom du mainteneur, ses dépendances, sa description. Tous les autres champs dépourvus de valeur par défaut sont optionnels et peuvent être

supprimés. Les champs `Copyright`, `Changelog`, `Readme` et `Extra-Files` ne sont pas standards pour un paquet Debian. Propres à **`equivs-build`**, ils disparaîtront des en-têtes réels du paquet généré.

EXEMPLE **Fichier d'en-têtes d'un faux paquet libxml-libxml-perl**

```
Section: perl
Priority: optional
Standards-Version: 3.5.10

Package: libxml-libxml-perl
Version: 1.57-1
Maintainer: Raphael Hertzog <hertzog@debian.org>
Depends: libxml2 (>= 2.6.6)
Architecture: all
Description: Fake package - module manually installed in site_perl
 This is a fake package to let the packaging system
 believe that this Debian package is installed.
 .
 In fact, the package is not installed since a newer version
 of the module has been manually compiled & installed in the
 site_perl directory.
```

L'étape suivante consiste à générer le paquet Debian en invoquant la commande **`equivs-build`** ***`fichier`***. Le tour est joué : le paquet est disponible dans le répertoire courant et vous pouvez désormais le manipuler comme tous les autres paquets Debian.

Simple archive de fichiers

Les administrateurs de Falcot SA souhaitent créer un paquet Debian pour déployer facilement un ensemble de documents sur un grand nombre de machines. Après avoir étudié le guide du nouveau mainteneur, l'administrateur en charge de cette tâche se lance dans la création de son premier paquet.

▸ http://www.debian.org/doc/maint-guide/index.fr.html

Il commence par créer un répertoire `falcot-data-1.0`, qui abritera le paquet source qu'il a choisi de réaliser. Ce paquet se nommera donc *falcot-data* et portera le numéro de version *1.0*. L'administrateur place ensuite les fichiers des documents qu'il souhaite distribuer dans un sous-répertoire `data`. Il invoque la commande **`dh_make`** (du paquet *dh-make*) pour ajouter les fichiers requis par le processus de génération d'un paquet (tous contenus dans un sous-répertoire `debian`) :

```
$ cd falcot-data-1.0
$ dh_make

Type of package: single binary, multiple binary, library, or kernel
  module?
 [s/m/l/k] s

Maintainer name : Raphael Hertzog
Email-Address   : hertzog@debian.org
Date            : Thu,  4 Mar 2004 18:55:01 +0100
Package Name    : falcot-data
Version         : 1.0
Type of Package : Single
```

```
Hit <enter> to confirm:
Currently there is no top level Makefile. This may require additional
tuning.  Done. Please edit the files in the debian/ subdirectory now.
You should also check that the falcot-data Makefiles install into
$DESTDIR and not in / .
```

Le type de paquet *single binary* indique que ce paquet source ne générera qu'un seul paquet binaire.

multiple-binary est à employer pour un paquet source générant plusieurs paquets binaires. Le type *library* est un cas particulier pour les bibliothèques partagées qui doivent suivre des règles de mise en paquet très strictes. Il en est de même pour *kernel module*, réservé aux paquets contenant des modules noyau.

Le programme **dh_make** a créé un sous-répertoire debian contenant de nombreux fichiers. Certains sont nécessaires : c'est notamment le cas des fichiers rules, control, changelog et copyright. Les fichiers d'extension .ex sont des fichiers d'exemples qu'on peut modifier et rebaptiser (en supprimant simplement cette extension) si cela s'avère utile. Dans le cas contraire, il convient de les supprimer. Le fichier compat doit être conservé car il est nécessaire au bon fonctionnement des programmes de l'ensemble appelé *debhelper*, dont les noms commencent par le préfixe **dh_** et qui sont employés à diverses étapes de la création de paquet.

Il faut mentionner dans le fichier copyright les auteurs des documents inclus dans le paquet et la licence logicielle associée. En l'occurrence, il s'agit de documents internes dont l'usage est limité à la société Falcot. Le fichier changelog par défaut convient relativement bien, et l'administrateur s'est contenté d'écrire une explication un peu plus longue que *Initial release* (version initiale) et de modifier la distribution *unstable* en *internal*. Le fichier control a lui aussi changé : la section a désormais pour valeur *misc* et le champ `Architecture` est passé de `any` (n'importe laquelle) à `all` (toutes) puisque le paquet abrite des documents et non des programmes binaires : il est donc exploitable sur toutes les architectures. Le champ `Depends` a été changé en `mozilla-browser | www-browser` pour garantir la présence d'un navigateur web capable de consulter les documents ainsi diffusés.

EXEMPLE Le fichier `control`

```
Source: falcot-data
Section: misc
Priority: optional
Maintainer: Raphael Hertzog <hertzog@debian.org>
Build-Depends: debhelper (>= 4.0.0)
Standards-Version: 3.6.0

Package: falcot-data
Architecture: all
Depends: mozilla-browser | www-browser
Description: Documentation interne de Falcot SA
 Ce paquet fournit plusieurs documents décrivant
 la structure interne de Falcot SA. Cela comprend:
  - l'organigramme
  - les contacts pour chaque département
 .
 Ces documents NE DOIVENT PAS sortir de la société.
```

ASTUCE Nom et adresse électronique du mainteneur

La plupart des programmes qui recherchent votre nom et adresse électronique de responsable de paquet utilisent les valeurs contenues dans les variables d'environnement DEBFULLNAME et DEBEMAIL ou EMAIL. En les définissant une fois pour toutes, vous vous éviterez de devoir les saisir à de multiples reprises. Si votre shell habituel est **bash**, il suffit pour cela d'ajouter les deux lignes suivantes dans vos fichiers ~/.bashrc et ~/.bash_profile (en remplaçant évidemment ces valeurs par celles qui vous correspondent !) :

```
export EMAIL="hertzog@debian.org"
export DEBFULLNAME="Raphael Hertzog"
```

```
 Ils sont réservés à un USAGE INTERNE.
```

EXEMPLE **Le fichier `changelog`**

```
falcot-data (1.0-1) internal; urgency=low

  * Initial Release.
  * Commençons avec peu de documents:
    - la structure interne de la société
    - les contacts de chaque département

 -- Raphael Hertzog <hertzog@debian.org>  Thu,  4 Mar 2004 18:55:01 +0100
```

EXEMPLE **Le fichier `copyright`**

```
This package was debianized by Raphael Hertzog <hertzog@debian.org> on
Thu,  4 Mar 2004 18:55:01 +0100.

Upstream Author(s): Falcot SA

Copyright:

Copyright 2004 Falcot SA - all rights reserved.
```

Le fichier rules contient normalement un ensemble de règles employées pour configurer, compiler et installer le logiciel dans un sous-répertoire dédié (portant le nom du paquet binaire généré). Le contenu de ce sous-répertoire est ensuite intégré au paquet Debian comme s'il était la racine du système de fichiers. Dans le cas qui nous concerne, les fichiers seront installés dans le répertoire debian/falcot-data/usr/share/falcot-data pour que les documents ainsi diffusés soient disponibles sous /usr/share/falcot-data dans le paquet généré. Le fichier rules est de type makefile avec quelques cibles standardisées (notamment `clean` et `binary`, respectivement pour nettoyer et produire le binaire). Dans le cas qui nous concerne, les cibles `build`, `install` et `clean` ont été modifiées. Les références à `$(MAKE)` ont été supprimées puisqu'il n'y aucun logiciel à compiler. La cible `install` a reçu les commandes permettant de créer le répertoire /usr/share/falcot-data et d'y copier la documentation. Voici le contenu final de ces cibles :

```
build: build-stamp

build-stamp: configure-stamp
        dh_testdir
        # Add here commands to compile the package.
        touch build-stamp

clean:
        dh_testdir
        dh_testroot
        rm -f build-stamp configure-stamp
        # Add here commands to clean up after the build process.
        dh_clean

install: build
        dh_testdir
        dh_testroot
        dh_clean -k
        dh_installdirs
        # Add here commands to install the package into debian/falcot-
          data.
        mkdir -p $(CURDIR)/debian/falcot-data/usr/share/falcot-data
        cp -a data/* $(CURDIR)/debian/falcot-data/usr/share/falcot-data
```

B.A.-BA **Fichier `Makefile`**

Un fichier makefile est un script détaillant au programme **`make`** les règles nécessaires pour reconstruire des fichiers issus d'un réseau de dépendances (un programme, fruit de la compilation de fichiers sources, en est un exemple). Le fichier makefile contient la liste de ces règles en respectant le format suivant :

```
cible: sources
        commandes
```

Cette règle peut se traduire ainsi : si l'un des fichiers de `sources` est plus récent que le fichier `cible`, il faut exécuter les `commandes` pour régénérer la cible à partir des sources.
Attention, un caractère de tabulation doit impérativement précéder toutes les commandes. Sachez aussi que si la ligne de commande débute par un signe moins (`-`), la commande peut échouer sans que tout le processus avorte.

À ce stade, il est déjà possible de créer le paquet. Nous allons toutefois y ajouter une dernière touche. Les administrateurs souhaitent que ces documents soient facilement accessibles depuis les menus Aide (ou *Help*) des bureaux graphiques. Ils décident donc de créer une entrée dans le système de menus Debian. Pour cela, ils modifient le fichier debian/menu.ex et l'enregistrent sans l'extension.

EXEMPLE **Le fichier menu**

```
?package(falcot-data):needs=X11|wm section=Help\
  title="Documentation interne à Falcot SA" \
  command="/usr/bin/x-www-browser /usr/share/falcot-data/index.html"
?package(falcot-data):needs=text section=Help\
  title="Documentation interne à Falcot SA" \
  command="/usr/bin/www-browser /usr/share/falcot-data/index.html"
```

Le champ `needs` positionné à `X11|wm` indique que cette entrée de menu n'a de sens que dans l'interface graphique. Elle sera donc intégrée uniquement dans les menus des applications graphiques (ou X11) et les gestionnaires de fenêtres (`wm` est en effet l'abréviation de *window manager*). Le champ `section` précise l'emplacement de l'entrée dans le menu. Dans notre cas, elle sera intégré au sous-menu d'aide *Help*. Le champ `title` (titre) est le texte que les utilisateurs verront dans le menu. Enfin, le champ `command` décrit la commande à exécuter lorsqu'un utilisateur sélectionne cet élément de menu.

La deuxième entrée est le pendant de la première, mais adaptée au mode texte d'une console Linux.

CHARTE DEBIAN **L'organisation des menus**

L'organisation des menus Debian suit une structure précise, documentée dans le texte suivant :
▸ http://www.debian.org/doc/packaging-manuals/menu-policy/
Il est recommandé de choisir une section listée dans ce document pour remplir le champ `section` d'un fichier menu.

Après rédaction du fichier menu, il s'agit de l'installer au bon endroit. Nous déléguerons cette tâche au programme **`dh_installmenu`** en décommentant la ligne correspondante dans la cible `binary-arch` du fichier rules.

Le paquet source est prêt ! Il ne reste plus qu'à générer le paquet binaire avec la commande déjà employée pour des recompilations de paquets : on se place dans le répertoire falcot-data-1.0 et on exécute **`dpkg-buildpackage -rfakeroot -us -uc`**.

Créer une archive de paquets pour APT

Les administrateurs de Falcot SA maintiennent désormais un certain nombre de paquets Debian modifiés ou créés par eux et qui leur servent à diffuser des données et programmes internes.

Pour faciliter leur déploiement, ils souhaitent les intégrer dans une archive de paquets directement utilisable par APT. Pour des raisons évidentes de maintenance, ils désirent y séparer les paquets internes des paquets officiels recompilés. Les entrées qui correspondraient à cette situation dans un fichier /etc/apt/sources.list seraient les suivantes :

```
deb http://packages.falcot.com/ updates/
deb http://packages.falcot.com/ internal/
```

Les administrateurs configurent donc un hôte virtuel sur leur serveur HTTP interne. La racine de l'espace web associé est /srv/vhosts/packages/. Pour gérer ces archives, ils ont décidé d'employer le programme **mini-dinstall** (du paquet éponyme). Celui-ci scrute un répertoire d'arrivée incoming (en l'occurrence, il s'agira de /srv/vhosts/packages/mini-dinstall/incoming) pour y récupérer tout paquet Debian déposé et l'installer dans une archive Debian (dont le répertoire est /srv/vhosts/packages). Ce programme fonctionne en traitant les fichiers .changes créés lors de la génération d'un paquet Debian. Un tel fichier contient en effet la liste de tous les autres fichiers associés à cette version du paquet (.deb, .dsc, .diff.gz, .orig.tar.gz) et permet donc à **mini-dinstall** de savoir quels fichiers installer. Accessoirement, ce fichier reprend le nom de la distribution de destination (c'est souvent *unstable*) indiquée en tête du fichier debian/changelog, information utilisée par **mini-dinstall** pour décider de l'emplacement d'installation du paquet. C'est la raison pour laquelle les administrateurs doivent systématiquement modifier ce champ avant la génération d'un paquet et y placer `internal` ou `updates`, selon l'emplacement souhaité. **mini-dinstall** génère alors les fichiers indispensables au bon fonctionnement d'APT, comme par exemple Packages.gz.

ALTERNATIVE **apt-ftparchive**

Si l'emploi de **mini-dinstall** semble trop complexe par rapport à vos besoins de création d'une archive Debian, il est possible d'utiliser directement le programme **apt-ftparchive**. Ce dernier inspecte le contenu d'un répertoire et affiche sur sa sortie standard le contenu du fichier Packages correspondant. Pour reprendre le cas de Falcot SA, les administrateurs pourraient directement déposer les paquets dans /srv/vhosts/packages/updates/ ou /srv/vhosts/packages/internal/ et exécuter les commandes suivantes pour créer les fichiers Packages.gz :

```
$ cd /srv/vhosts/packages
$ apt-ftparchive packages updates >updates/Packages
$ gzip updates/Packages
$ apt-ftparchive packages internal >internal/Packages
$ gzip internal/Packages
```

La commande **apt-ftparchive sources** permet de créer de manière similaire les fichiers Sources.gz.

La configuration de **mini-dinstall** nécessite de mettre en place un fichier ~/.mini-dinstall.conf, que les administrateurs de Falcot SA ont renseigné comme suit :

```
[DEFAULT]
archive_style = flat
archivedir = /srv/vhosts/packages

verify_sigs = 0
mail_to = admin@falcot.com

generate_release = 1
release_origin = Falcot SA
release_codename = stable
```

```
[updates]
release_label = Recompiled Debian Packages

[internal]
release_label = Internal Packages
```

Il est intéressant d'y remarquer la décision de générer des fichiers Release pour chacune des archives. Cela permettra éventuellement de gérer les priorités d'installation des paquets à l'aide du fichier de configuration /etc/apt/preferences (voir le chapitre sur la configuration d'APT).

L'exécution de **mini-dinstall** démarre en fait le démon en arrière-plan. Tant qu'il fonctionne, il vérifiera toutes les demi-heures si un nouveau paquet est disponible dans le répertoire incoming, le placera dans l'archive, et régénérera les différents fichiers Packages.gz et Sources.gz. Si la présence d'un démon constitue un problème, il est possible de l'invoquer en mode non interactif (ou *batch*), à l'aide de l'option **-b**), à chaque fois qu'un paquet aura été déposé dans le répertoire incoming. Découvrez les autres possibilités offertes par **mini-dinstall** en consultant sa page de manuel `mini-dinstall(1)`.

SÉCURITÉ **mini-dinstall et droits**

mini-dinstall étant prévu pour fonctionner dans un compte utilisateur, il ne serait pas raisonnable de l'employer avec le compte root. La solution la plus simple est de tout configurer au sein du compte utilisateur de l'administrateur qui a la responsabilité de créer les paquets Debian. Étant donné que lui seul a le droit de déposer des fichiers dans le répertoire incoming, il n'est pas nécessaire d'authentifier l'origine de chaque paquet à installer : on peut considérer que l'administrateur l'aura fait préalablement. Cela justifie le paramètre `verify_sigs = 0` (pas de vérification des signatures). Toutefois, si le contenu des paquets est très sensible, il est possible de revenir sur ce choix et d'avoir un trousseau de clés publiques identifiant les personnes habilitées à créer des paquets (le paramètre `extra_keyrings` existe à cette fin) ; **mini-dinstall** vérifiera la provenance de chaque paquet déposé en analysant la signature intégrée au fichier .changes.

Devenir mainteneur de paquet

Apprendre à faire des paquets

Construire un paquet Debian de qualité n'est pas chose facile, et on ne s'improvise pas responsable de paquet. C'est une activité qui s'apprend par la pratique et par la théorie, et qui ne se limite pas à compiler et installer un logiciel. Elle implique surtout de maîtriser les problèmes, conflits et interactions qui se produiront avec les milliers d'autres paquets logiciels.

Les règles

Un paquet Debian est conforme aux règles précises édictées dans la charte Debian. Chaque responsable de paquet se doit de les connaître. Il ne s'agit pas de les réciter par cœur, mais de savoir qu'elles existent et de s'y référer lorsque l'on n'est pas sûr de son choix. Tout mainteneur Debian officiel a déjà commis des erreurs en ignorant l'existence d'une règle, mais ce n'est pas dramatique : un utilisateur avancé de ses paquets finit tôt ou tard par signaler cette négligence sous la forme d'un rapport de bogue.

▸ http://www.debian.org/doc/debian-policy/

Les procédures

Debian n'est pas une collection de paquets réalisés individuellement. Le travail de chacun s'inscrit dans un projet collectif et à ce titre on ne peut être développeur Debian et ignorer le fonctionnement global de la distribution. Tôt ou tard, chaque développeur doit interagir avec d'autres volontaires. La référence du développeur Debian (paquet *developers-reference-fr*) reprend tout ce que chaque développeur doit savoir pour interagir au mieux avec les différentes équipes du projet et profiter au maximum des ressources mises à disposition. Ce document précise également un certain nombre de devoirs que chaque développeur se doit de remplir.

▸ http://www.debian.org/doc/developers-reference/

Les outils

Toute une panoplie d'outils aide les responsables de paquets dans leur travail. Ce chapitre les décrit rapidement sans détailler leur emploi, car ils sont tous bien documentés.

Les programmes **`lintian`** et **`linda`**

Ces deux premiers programmes font partie des outils les plus importants : ce sont les vérificateurs de paquets Debian. Chacun dispose d'une batterie de tests créés en fonction de la charte Debian. Ils permettent de trouver rapidement et automatiquement de nombreuses erreurs et donc de les corriger avant de publier les paquets. Ces deux programmes sont globalement assez redondants, leur différence majeure étant que le premier est écrit en Perl alors que le second est en Python. Deux vérifications valant mieux qu'une, ce n'est pas gênant de les employer tous les deux.

Ces outils ne fournissent qu'une aide, et il arrive qu'ils se trompent (la charte Debian évolue parfois, ces programmes sont alors momentanément en retard). Par ailleurs, ces logiciels ne sont pas exhaustifs : qu'ils ne signalent aucune erreur ne signifie pas qu'un paquet est parfait, tout au plus qu'il évite les erreurs les plus communes.

devscripts

Le paquet *devscripts* contient de nombreux programmes couvrant bien des aspects du travail d'un développeur Debian :

- **`debuild`** permet de générer un paquet (**`dpkg-buildpackage`**) et de vérifier dans la foulée s'il est conforme à la charte Debian (**`lintian`** ou **`linda`**).
- **`debclean`** nettoie un paquet source après la génération d'un paquet binaire.
- **`dch`** permet d'éditer facilement un fichier `debian/changelog` dans un paquet source.

- **uscan** vérifie si l'auteur amont a publié une nouvelle version de son logiciel. Ce programme nécessite un fichier debian/watch décrivant l'emplacement de publication de ces archives.
- **debi** permet d'installer (**dpkg -i**) le paquet Debian qui vient d'être généré (sans devoir saisir son nom complet).
- **debc** permet de consulter le contenu (**dpkg -c**) du paquet qui vient d'être généré (sans devoir saisir son nom complet).
- **bts** manipule le système de suivi de bogues depuis la ligne de commande ; ce programme génère automatiquement les courriers électroniques adéquats.
- **debrelease** envoie la nouvelle version du paquet sur un serveur distant sans devoir saisir le nom complet du fichier .changes concerné.
- **debsign** signe les fichiers .dsc et .changes.
- **uupdate** crée automatiquement une nouvelle révision du paquet lors de la publication d'une nouvelle version amont.

debhelper et *dh-make*

debhelper est un ensemble de scripts facilitant la création d'un paquet conforme à la charte Debian, et invoqués depuis debian/rules. Il a conquis de très nombreux développeurs Debian puisque la majorité des paquets officiels l'utilisent. Tous les scripts sont préfixés par **dh_**.

ALTERNATIVE **CDBS**

cdbs offre une autre approche de la réalisation des paquets Debian, entièrement basée sur un système d'héritage entre fichiers Makefile.

Le script **dh_make** (du paquet *dh-make*) intègre les fichiers nécessaires à la génération d'un paquet Debian dans un répertoire contenant les sources d'un logiciel. Les fichiers qu'il ajoute utilisent *debhelper* de manière standard, comme son nom le laisse supposer.

dupload et **dput**

dupload et **dput** permettent d'envoyer une nouvelle version d'un paquet Debian sur un serveur local ou distant. Cela permet aux développeurs d'envoyer leur paquet sur le serveur principal de Debian (`ftp-master.debian.org`) pour qu'il soit intégré à l'archive et distribué par les miroirs. Ces commandes prennent en paramètre un fichier .changes et en déduisent les autres fichiers à envoyer.

Processus d'acceptation

Ne devient pas développeur Debian qui veut. Différentes étapes jalonnent le processus d'acceptation, qui se veut autant un parcours initiatique qu'une sélection. Ce processus est formalisé et chacun peut suivre sa progression sur le site web des nouveaux mainteneurs (`nm` est l'abréviation de *New Maintainer*).

▸ http://nm.debian.org

Prérequis

Il est demandé à tous les candidats de maîtriser un minimum l'anglais. C'est nécessaire à tous les niveaux : dans un premier temps pour communiquer avec l'examinateur, mais c'est aussi la langue de prédilection pour une grande partie de la documentation. De plus, les utilisateurs de vos paquets communiqueront avec vous en anglais pour vous signaler des bogues, et il faudra être capable de leur répondre.

Le deuxième prérequis porte sur la motivation. Il faut être pleinement conscient que la démarche qui consiste à devenir développeur Debian ne vaut le coup d'être effectuée que si vous savez par avance que Debian restera un sujet d'intérêt pendant de nombreux mois. En effet, la procédure en elle-même dure plusieurs mois et Debian a besoin de mainteneurs qui s'inscrivent dans la durée, car chaque paquet a besoin d'un mainteneur en permanence (et pas seulement lorsqu'il est créé).

Inscription

La première étape (réelle) consiste à trouver un « sponsor », ou « avocat » (*advocate*) ; c'est un développeur officiel qui affirme « je pense que l'acceptation de *X* serait une bonne chose pour Debian ». Cela implique normalement que le candidat ait déjà été actif au sein de la communauté et que quelqu'un ait apprécié son travail. Si le candidat est timide et n'affiche pas en public le fruit de son travail, il peut tenter de convaincre individuellement un développeur Debian officiel de le soutenir en lui présentant ses travaux en privé.

En parallèle, le candidat doit se générer une paire de clés DSA/ElGamal avec GnuPG, qu'il doit faire signer par au moins un développeur Debian officiel. La signature certifie l'authenticité du nom présent sur la clé. En effet, lors d'une séance de signature de clés, il est d'usage de présenter des papiers d'identité et les identifiants de ses clés pour officialiser la correspondance entre la personne physique et les clés. Cette signature nécessite donc une rencontre réelle ; si vous n'avez pas encore eu l'occasion de croiser un développeur Debian lors d'une manifestation de logiciels libres, il est possible de solliciter expressément les développeurs en demandant qui serait dans la région concernée par le biais de la liste de diffusion debian-devel-french@lists.debian.org.

Une fois l'inscription sur `nm.debian.org` validée par le sponsor, un *Application Manager* (gestionnaire d'application) sera assigné au candidat. C'est la personne qui va le suivre dans ses démarches et réaliser les différentes vérifications prévues dans le processus.

La première vérification est celle de l'identité. Si vous avez une clé signée par un développeur Debian cette étape est facile. Dans le cas contraire, l'*Application Manager* essaiera de guider le candidat dans sa recherche de développeurs Debian à proximité de chez lui pour qu'une rencontre et une signature de clés puissent être arrangées. Au tout début, lorsque le nombre de développeurs était très

restreint, il était possible de s'identifier à l'aide d'une capture numérique (*scan*) des papiers d'identité.

Acceptation des principes

Ces formalités administratives sont suivies de considérations philosophiques. Il est question de s'assurer que le candidat comprend le contrat social et les principes du logiciel libre. En effet, il n'est pas possible de rejoindre Debian si l'on ne partage pas les valeurs qui unissent les développeurs actuels, exprimées dans les deux textes fondateurs.

En plus de cela, il est souhaité que chaque personne qui rejoint les rangs de Debian connaisse déjà son fonctionnement et sache interagir comme il se doit pour résoudre les problèmes qu'elle rencontrera au fil du temps. Toutes ces informations sont généralement documentées dans les divers manuels ciblant les nouveaux mainteneurs, mais aussi et surtout dans le guide de référence du développeur Debian. Une lecture attentive de ce document devrait suffire pour répondre aux questions de l'examinateur. Si les réponses ne sont pas satisfaisantes, il le fera savoir et invitera le candidat à se documenter davantage avant de retenter sa chance. Si la documentation ne semble pas répondre à la question, c'est qu'un peu de pratique au sein de Debian permet de découvrir la réponse par soi-même (éventuellement en discutant avec d'autres développeurs Debian). Ce mécanisme entraîne les gens dans les rouages de Debian avant de pouvoir totalement prendre part au projet. C'est une politique volontaire, et les gens qui arrivent finalement à rejoindre le projet s'intègrent comme une pièce supplémentaire d'un puzzle extensible à l'infini.

Cette étape est couramment désignée par le terme de *Philosophy & Procedures* (*P&P*) dans le jargon des personnes impliquées dans le processus d'acceptation de nouveaux mainteneurs.

Vérification des compétences

Chaque demande pour devenir développeur Debian officiel doit être justifiée. On ne peut en effet devenir membre que si l'on peut démontrer que ce statut est légitime et qu'il permettra de faciliter le travail du candidat. La justification habituelle est que le statut de développeur Debian facilite la maintenance d'un paquet Debian, mais elle n'est pas universelle. Certains développeurs rejoignent le projet pour contribuer à un portage sur une architecture, d'autres pour contribuer à la documentation, etc.

Cette étape est donc l'occasion pour chaque candidat d'affirmer ce qu'il a l'intention de réaliser dans le cadre de Debian et de montrer ce qu'il a déjà fait dans ce sens. Debian privilégie en effet le pragmatisme et il ne suffit pas de dire quelque chose pour le faire prendre en compte : il faut montrer sa capacité à faire ce qui a été annoncé. En général, lorsqu'il s'agit de mise en paquet, il faudra montrer une première version du paquet et trouver un parrain (parmi les développeurs

officiels) qui contrôle sa réalisation technique et l'envoie sur le serveur principal de Debian.

Enfin, l'examinateur vérifiera les compétences techniques du candidat en matière de mise en paquet grâce à un questionnaire assez étoffé. L'erreur n'est pas permise, mais le temps pour répondre n'est pas limité, toute la documentation est disponible, et il est possible d'essayer plusieurs fois en cas d'erreur. Le questionnaire ne se veut pas discriminatoire mais a pour seul objectif de garantir un niveau minimum de connaissances aux nouveaux contributeurs.

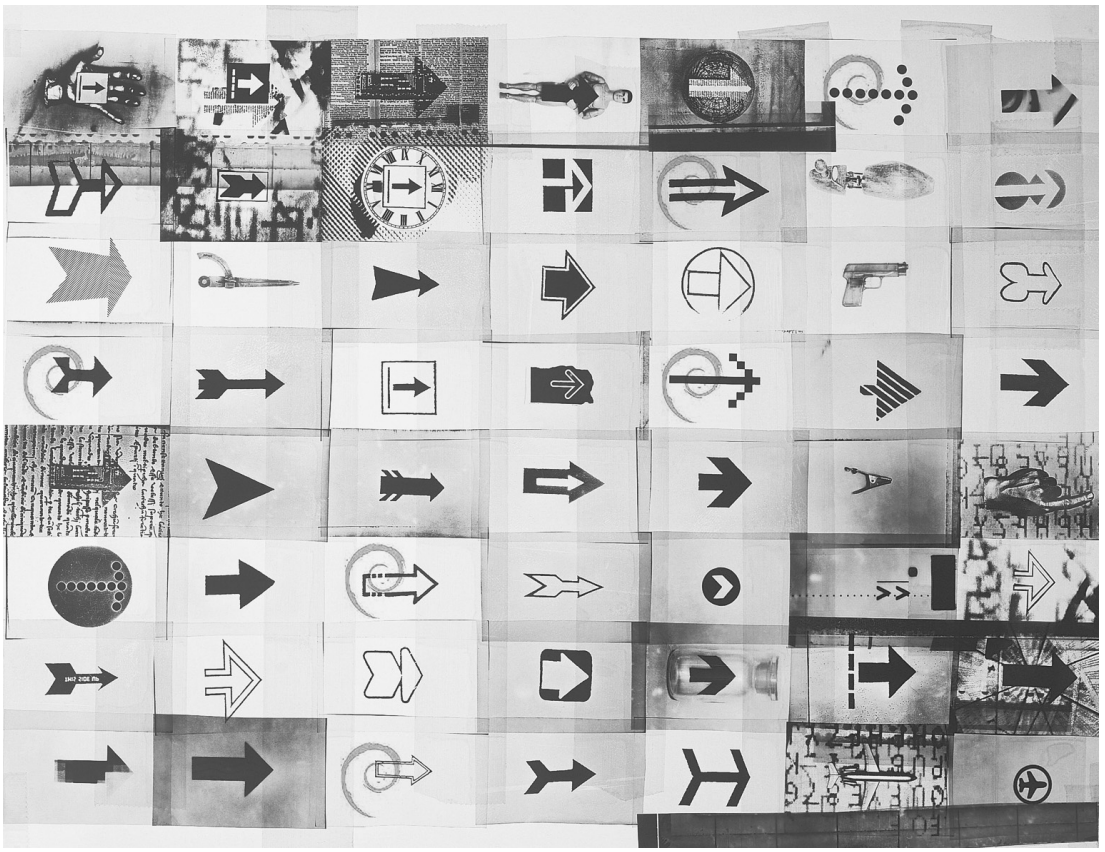
Cette étape se nomme *Tasks & Skills* dans le jargon des examinateurs.

Approbation finale

La toute dernière étape est une étape de validation du parcours par le DAM (*Debian Account Manager*, ou gestionnaire des comptes Debian). Il consulte les informations fournies à propos du candidat par l'examinateur et prend la décision de lui créer ou non un compte sur les serveurs Debian. Parfois, il temporisera cette création dans l'attente d'informations supplémentaires s'il le juge nécessaire. Les refus sont assez rares si l'examinateur a bien fait son travail d'encadrement, mais ils se produisent parfois. Ils ne sont jamais définitifs, et le candidat est libre de retenter sa chance ultérieurement.

La décision du DAM est souveraine et quasiment incontestable. C'est pourquoi le responsable concerné, il s'agit aujourd'hui de James Troup, est souvent critiqué. Par ailleurs, cette étape représente un goulet d'étranglement dans le processus et il n'est pas rare d'y attendre plusieurs mois (pendant ce temps, il est hautement recommandé de continuer à contribuer par l'intermédiaire d'un parrain).

14



Conclusion : l'avenir de Debian

L'histoire de Falcot SA s'arrête, pour le moment, avec ce dernier chapitre. Mais celle de Debian continue et l'avenir nous réserve à coup sûr de nombreuses et agréables surprises.

SOMMAIRE

- Développements à venir
- Avenir de Debian
- Avenir de ce livre

MOTS-CLEFS

- Avenir
- Améliorations
- Opinions

Développements à venir

Quelques semaines à quelques mois avant la sortie d'une nouvelle version, le *Release Manager* choisit le nom de code de la prochaine. Alors que la version 3.1 de Debian paraîtra sous peu, les développeurs s'affairent déjà à la préparation de la version suivante : nom de code *etch*...

Il n'existe pas de liste des changements prévus et Debian ne s'engage jamais quant aux objectifs techniques de la version suivante. Mais quelques axes de développement existent et on a toutes les raisons de croire qu'ils se concrétiseront dans cette nouvelle version.

Ainsi, une nouvelle version d'APT sera capable d'authentifier les paquets téléchargés. L'architecture *amd64* — des nouveaux processeurs 64 bits du fondeur AMD — sera enfin officiellement prise en charge et de nombreuses adaptations de la structure de la distribution faciliteront la cohabitation de programmes 32 et 64 bits.

Bien entendu, tous les principaux logiciels auront connu une mise à jour majeure.

Avenir de Debian

En dehors de ces développements internes, il est probable que de nouvelles distributions fondées sur Debian verront le jour grâce à la popularisation de *debian-installer* et à sa facilité d'adaptation. Par ailleurs de nouveaux sous-projets spécifiques naîtront, élargissant toujours le spectre des domaines couverts par Debian.

La communauté des utilisateurs Debian se sera étoffée, et de nouveaux contributeurs rejoindront le projet... dont vous serez peut-être !

Force est de constater que le projet Debian est plus vigoureux que jamais, et qu'il est désormais bien lancé vers son objectif de distribution universelle. *World domination* (ou domination mondiale), dit-on en plaisantant dans les rangs de Debian.

Malgré son ancienneté et sa taille déjà importante Debian continue de croître et d'évoluer dans de nombreuses directions — certaines sont d'ailleurs inattendues. Les contributeurs ne manquent jamais d'idées, et les discussions sur les listes de développement — même si parfois elles ressemblent à des chamailleries — ne cessent d'alimenter la machine. Certains comparent même Debian à un trou noir : sa densité est telle qu'elle attire systématiquement tout nouveau projet libre.

Au-delà du fait que Debian semble satisfaire une majorité de ses utilisateurs il y a une tendance de fond : les gens commencent à se rendre compte qu'en collaborant — plutôt que de faire sa cuisine dans son coin — il est possible d'obtenir un résultat meilleur pour tous. C'est bien la logique suivie par toutes les distributions qui se greffent à Debian, en formant des sous-projets.

Le projet Debian n'est donc pas près de disparaître...

Avenir de ce livre

Je souhaite que ce livre, même s'il n'est pas publié sous une licence œuvre libre, puisse évoluer dans l'esprit du logiciel libre. C'est pourquoi je vous invite à y contribuer en me faisant part de vos remarques, de vos suggestions et de vos critiques. Pour cela, vous pouvez m'écrire directement à hertzog@debian.org. Le site web ci-dessous regroupera l'ensemble des informations portant sur son évolution.

▸ http://www.ouaza.com/livre/admin-debian/

J'ai essayé d'intégrer tout ce que mon expérience chez Debian m'a fait découvrir, afin que tout un chacun puisse utiliser cette distribution et en tirer le meilleur profit le plus rapidement possible. Je souhaite que ce livre contribue à la démystification et à la popularisation de Debian. N'hésitez donc pas à le recommander !

Pour conclure, voici une note personnelle. J'ai mis près d'un an à rédiger ce livre en parallèle à une activité professionnelle normale. Mon souhait actuel est de pouvoir me consacrer à Debian et aux logiciels libres à plein temps. J'espère que ce livre pourra me servir de tremplin pour bondir dans cette direction.

À bientôt !

Annexe


Bitstream Vera
Courrier entrant pour rhertzog@(none) - Mozilla Thunderbird
Affichage Aller à Messages
Adresses Decrypt Répondre
Voir : Tous
Sujet
Compte personnel
Courrier entrant
Dossiers locaux
Affichage Insertion Format
Corbeille
Fichier Edition Affichage Messages Enigmail Outils Aide
Relever Écrire Adresses Répondre Rép. à tous Transférer Supprimer
Dossiers
Voir : Tous
Le sujet ou l'expéditeur contient :
Effacer
Compte personnel
Sujet
Expéditeur
Date

Distributions dérivées

De nombreuses distributions dérivent de Debian et emploient ses outils. Chacune présente des particularités intéressantes, et comblera peut-être vos attentes !

SOMMAIRE

- Ubuntu Linux
- Knoppix
- Mepis Linux
- Xandros
- Libranet
- Linspire

MOTS-CLEFS

- Live CD
- Spécificités
- Choix particuliers

Ubuntu Linux

Ubuntu Linux est une des plus récentes distributions dérivées de Debian, mais son entrée sur la scène du logiciel libre a été très médiatique. Et pour cause : la société Canonical Ltd. qui a créé cette distribution a embauché une vingtaine de développeurs Debian en affichant l'ambitieux objectif de faire une distribution pour le grand public, et de publier une nouvelle version tous les 6 mois. Ils promettent par ailleurs de maintenir chaque version pendant 18 mois.

Pour parvenir à leurs objectifs, ils se concentrent sur un nombre restreint de logiciels, et s'appuient essentiellement sur GNOME. Tout est internationalisé et disponible dans un grand nombre de langues, dont le français.

La première version de leur distribution, numérotée « 4.10 », porte le nom de code « Warty Warthog ». Ce numéro de version symbolise simplement une date : 4.10 représente le mois d'octobre 2004. La deuxième version, prévue pour le mois d'avril 2005 et intitulée « Hoary Hedgehog », sera numérotée 5.04.

L'implication forte de développeurs Debian dans ce projet garantit que la plupart des améliorations et corrections réalisées dans le cadre d'Ubuntu seront réintégrées dans Debian même — le développement d'Ubuntu contribue donc directement à l'amélioration de Debian. C'est la raison pour laquelle je souhaite bonne chance à cette distribution, qui mérite de se faire sa place dans le monde des distributions Linux.

▸ http://www.ubuntulinux.org/

Knoppix

La distribution Knoppix n'a presque plus besoin d'être présentée. Elle a popularisé le concept de *LiveCD* : il s'agit d'un cédérom amorçable qui démarre directement un système Linux fonctionnel et prêt à l'emploi, sans nécessiter de disque dur — tout système déjà présent sur la machine sera donc laissé intact. L'autodétection des périphériques permet à cette distribution de fonctionner avec presque toutes les configurations matérielles. Le cédérom contient près de 2 Go de logiciels compressés.

Si vous cumulez ce cédérom avec une clé USB, vous pourrez emmener vos fichiers avec vous et travailler sur n'importe quel ordinateur sans laisser de trace — rappelons que la distribution n'utilise pas du tout le disque dur. Knoppix est essentiellement fondé sur KDE, mais de nombreuses dérivées proposent d'autres combinaisons de logiciels. Citons notamment Morphix, qui s'appuie sur une structure modulaire permettant d'offrir plusieurs LiveCD — chacun avec sa propre sélection de logiciels.

Signalons en outre que la distribution offre malgré tout un installateur : vous pourrez ainsi essayer Knoppix en tant que *LiveCD* puis, une fois convaincu, l'installer sur le disque dur pour obtenir de meilleures performances.

- http://www.knoppix-fr.org/
- http://www.morphix.org/

COMMUNAUTÉ **Gnoppix et Ubuntu**

Ubuntu propose également un *LiveCD* : pour cela ils ont fait appel au développeur de Gnoppix — un *LiveCD* fondé sur GNOME.

- http://www.gnoppix.org/

Mepis Linux

Mepis Linux est une distribution commerciale très similaire à Knoppix. Proposant un système Linux prêt à l'emploi depuis un *LiveCD*, cette distribution intègre un certain nombre de logiciels qui ne sont pas libres : les pilotes pour les cartes graphiques nVidia, Macromedia Flash pour les animations intégrées à de nombreux sites web, RealPlayer, le Java de Sun, etc. L'objectif est d'offrir un système 100 % fonctionnel dès l'installation. Mepis est internationalisée et gère la langue française.

- http://www.mepis.com/
- http://www.mepis.org/

Xandros

Xandros Linux est une distribution commerciale traditionnelle, qui dispose d'un installateur graphique ultra-simplifié en 4 étapes. Elle cible principalement les utilisateurs anglophones, puisque seul l'anglais est disponible dans l'installateur et dans le manuel inclus dans la version « boîte ». Comme tout système Linux, il est toutefois toujours possible de configurer le système en français après l'installation.

Il existe plusieurs versions de la distribution, adaptées à des usages différents : la version familiale propose un système standard adapté pour un usage bureautique et ludique ; la version entreprise propose en plus des outils de gestion de parc de machines, etc.

- http://www.xandros.com/

Libranet

Libranet est une distribution commerciale canadienne pour poste de travail (bureautique). La plus récente version est vendue tandis que la précédente est téléchargeable gratuitement. L'installateur simplifié configure automatiquement la gestion du matériel. Libranet propose un logiciel graphique (*adminmenu*) facilitant la configuration de nombreux aspects d'un système Linux. La distribution n'est pas officiellement internationalisée et l'anglais est de rigueur.

- http://www.libranet.com/

Linspire

Cette distribution commerciale vise le grand public, la société éditrice passe des accords de distribution OEM importants aux États-Unis (notamment avec le distributeur Walmart). Son dirigeant principal, Michael Robertson, est connu pour ses prises de position très tranchées. Le premier nom de la distribution était d'ailleurs « Lindows », et une longue guerre juridique les a opposés à Microsoft parce que ce nom était trop proche de celui de leur système d'exploitation Windows.

Cette distribution est elle aussi destinée à un public anglophone.

▸ http://www.linspire.com/



Glossaire

`/etc/init.d/`*`service`*` restart` Voir « redémarrage des services ».

alternatives Voir « choix (*alternatives*) ».

APT Ensemble logiciel facilitant les modifications globales sur le système : installation ou suppression d'un paquet en gérant les dépendances, mise à jour du système, consultation de la liste des paquets disponibles, etc.

`apt-get.org` Site regroupant des sources non officielles de paquets Debian.
Voir « ressources non officielles ».

`ar` Programme **`ar`**, qui permet de manipuler les archives du même nom : **`ar t`** ***`archive`*** fournit la liste des fichiers contenus dans l'archive, **`ar x`** ***`archive`*** extrait les fichiers de l'archive dans le répertoire courant, **`ar d`** ***`archive fichier`*** supprime un fichier de l'archive, etc. Sa page de manuel documente ses nombreuses autres opérations. **`ar`** est un outil très rudimentaire, qu'un administrateur Unix emploie rarement. Mais il emploie régulièrement de **`tar`**, programme de gestion d'archives et de fichiers plus évolué. Grâce à **`ar`** il est facile de restaurer **`dpkg`** en cas de suppression involontaire. Il suffit de télécharger son paquet Debian et d'extraire le contenu de son archive `data.tar.gz` dans la racine du système (/) :

```
# ar x dpkg_1.10.18_i386.deb
# tar -C / -p -zxf data.tar.gz
```

artistique, licence Licence de logiciel libre qui associe la possibilité d'intégration du code dans une application propriétaire et l'obligation de publication des changements de toute version modifiée et distribuée. Elle est utilisée par le langage de programmation Perl, conjointement avec la GNU GPL.
▸ http://www.opensource.org/licenses/artistic-license.php
Voir « licences libres ».

auteur amont Traduction littérale de *upstream author*, ce terme désigne le ou les auteurs/développeurs d'un logiciel, qui l'écrivent et le font évoluer.
De manière générale, Debian encourage l'implication des responsables de paquets dans le développement « amont » (de simples utilisateurs d'un logiciel ils deviennent alors contributeurs).
Voir « développeur Debian ».

autobuilders Machines du projet Debian consacrées à la production de paquets binaires.
Voir « **`buildd`** ».

binaire, paquet Voir « paquet binaire ».

bo Voir « noms de code ».

bogue Description d'un problème de fonctionnement d'un paquet. À chaque bogue, Debian associe un niveau de gravité dans son système de suivi. Les niveaux les plus élevés sont traités en priorité et doivent tous être éliminés dans une version « stable » de la distribution.
Voir « sévérité d'un bogue ».
Voir « système de suivi de bogues ».

Bruce Perens Voir « Perens, Bruce ».

BSD, licence L'une des licences de logiciel libre les plus répandues. Elle n'est pas *copyleft* ; en particulier, il est possible d'en diffuser des versions modifiées sans en fournir le code source. De grands noms de l'informatique, par exemple Microsoft et Apple, ont profité de cette possibilité.
▸ http://www.opensource.org/licenses/bsd-license.php
Voir « licences libres ».

BTS (*Bug Tracking System*) Voir « système de suivi de bogues ».

build daemon buildd Voir « **buildd** ».

Build-Depends Champ permettant la mise en place d'un système minimal pour la reconstruction d'un paquet.
Voir « **pbuilder** ».

buildd Abréviation de *build daemon*. Ce logiciel recompile automatiquement les nouvelles versions des paquets Debian sur l'architecture qui l'accueille (la compilation croisée — *crosscompiling* — n'étant pas toujours satisfaisante).
Ainsi pour produire des binaires destinés à l'architecture `sparc`, le projet dispose de machines `sparc` (en l'occurrence de marque Sun). Le programme *buildd* y fonctionne en permanence afin de créer des paquets binaires pour `sparc` à partir des paquets sources expédiés par les développeurs Debian.
Voir « *autobuilders* ».

buzz Voir « noms de code ».

charte Debian Document de référence définissant les règles techniques à respecter pour la mise en paquet des logiciels. Connues sous le nom anglophone de *Debian policy*, ces normes sont élaborées de manière collaborative sur la liste de diffusion debian-policy@lists.debian.org. Voir l'encadré page 11 pour les détails sur le processus d'élaboration.

choix (*alternatives*) Mécanisme permettant à l'administrateur de choisir un programme de prédilection parmi plusieurs logiciels offrant les mêmes fonctionnalités. Ces derniers « fournissent » (au sens du champ d'en-tête *Provides*) souvent le paquet virtuel correspondant.
La charte Debian définit un certain nombre de commandes capables d'effectuer une action prédéfinie. Ainsi, la commande **x-window-manager** invoque un gestionnaire de

fenêtres. Au lieu d'affecter cette commande à un gestionnaire de fenêtres présélectionné, Debian permet à l'administrateur de l'associer au gestionnaire de son choix.

Chaque gestionnaire de fenêtres s'enregistre comme un choix valable pour **`x-window-manager`** et fournit une priorité associée. Celle-ci permet de sélectionner automatiquement le meilleur gestionnaire de fenêtres installé en l'absence d'un choix explicite de l'administrateur.

C'est le script **`update-alternatives`** qui est à la base de ce mécanisme, voir l'encadré page 226 pour plus de détails à son propos.

Voir « paquet virtuel ».

Voir « *Provides* ».

chroot Technique qui limite volontairement l'espace du disque dur accessible à un répertoire particulier, qui devient la nouvelle « racine » du système. Parfois utilisée pour des raisons de sécurité (éviter qu'un programme ou utilisateur douteux n'explore ou ne modifie d'autres fichiers du disque), elle permet encore de contrôler la configuration minimale requise pour une compilation : si l'on ne place pas dans cette zone contrôlée tous les éléments nécessaires, le processus échouera au lieu d'y faire appel à notre insu.

Voir « environnement *chrooté* ».

Colin Watson Co-gestionnaire de version avec Steve Langasek depuis août 2004.

Voir « *Release Manager* ».

comité technique Groupe restreint de personnes aux compétences reconnues qui arbitre les débats d'ordre technique opposant des mainteneurs. Ce comité est défini par la constitution.

Voir « constitution ».

Common Unix Printing System (CUPS) Voir « CUPS (*Common Unix Printing System*) ».

Componentized Linux Nouvelle approche d'une distribution Linux. On n'y assemble plus des paquets, mais des collections cohérentes de paquets, plus faciles à gérer. Chacune progresse indépendamment des autres, facilitant ainsi l'évolution globale du système.

Voir « Murdock, Ian ».

config Voir « scripts de configuration ».

configuration de paquets Lors de l'installation d'un nouveau paquet sur un système Debian, il est souvent nécessaire de faire des choix de configuration (qui peuvent se limiter à la validation des valeurs proposées par défaut).

Voir « **`debconf`** ».

Conflicts Champ d'en-tête déclarant un conflit entre deux paquets Debian.

Voir « conflit ».

conflit Situation dans laquelle un paquet ne peut pas cohabiter avec un autre paquet. Notion détaillée au chapitre 5.

constitution — Texte formel décrivant les pouvoirs, devoirs et interactions des différentes entités du projet Debian : le *leader*, le secrétaire, les développeurs et le comité technique.
▸ http://www.debian.org/devel/constitution
Voir « *leader* ».
Voir « comité technique ».

contrat social — Texte fondateur définissant l'objet de Debian et ses engagements envers ses utilisateurs. Ce texte se trouve page 6.
Voir « *Debian Free Software Guidelines* ».

contrib, archive — Collection secondaire de paquets Debian contenant les logiciels libres ne fonctionnant pas sans certains éléments non libres. Il peut s'agir de programmes qui dépendent 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. Ce fut le cas de la suite bureautique OpenOffice.org, qui avait besoin d'un environnement Java propriétaire.

copyleft — Ce terme, traduit littéralement par « gauche d'auteur », est une astucieuse et révélatrice distorsion de *copyright*.
Une personne non détentrice du *copyright* d'une œuvre sous *copyleft* peut la redistribuer si elle fournit aussi à autrui les mêmes droits que ceux dont elle bénéficia (notamment la possibilité d'en exiger le code source).
Voir « licences libres ».

CUPS (*Common Unix Printing System*) — Abréviation de *Common Unix Printing System* (système d'impression commun sous Unix) et marque déposée de la société *Easy Software Products*, à l'origine de **cupsys**, un gestionnaire d'impression.

DAM (*Debian Account Manager*) — Voir « *Debian Account Manager* ».

debconf — Logiciel qui fut créé pour résoudre un problème récurrent chez Debian. Tous les paquets Debian incapables de fonctionner sans un minimum de configuration posaient des questions à l'utilisateur via des moyens techniques rendant leur rationalisation difficile, par exemple des appels à **echo** et **read** placés dans les scripts shell postinst. Mais cela impliquait également, lors d'une grosse installation ou mise à jour, de rester à côté de son ordinateur pour renseigner ces requêtes qui pouvaient se produire à tout moment. Ces interactions manuelles ont désormais presque totalement disparu au profit de l'outil **debconf**.
debconf offre de nombreuses caractéristiques intéressantes : il contraint le développeur à spécifier les interactions avec l'utilisateur, il permet de localiser les différentes chaînes de caractères affichées (toutes les traductions sont stockées dans le fichier templates décrivant les interactions), il dispose de différents modules d'affichage pour présenter les questions à l'utilisateur (modes texte, graphique, non interactif), et il permet de créer une base centrale de réponses pour partager la même configuration entre plusieurs ordinateurs... Mais la plus importante est qu'il est maintenant possible de présenter toutes les questions d'un bloc à l'utilisateur avant de démarrer l'installation ou mise à jour correspondante.

debhelper Ensemble de scripts facilitant la réalisation de paquets conformes à la charte Debian. Joey Hess en est l'auteur principal.
Voir « charte Debian ».

Debian Account Manager (DAM) Terme anglais que l'on peut traduire par « Responsable des comptes Debian », c'est-à-dire la personne chargée d'accepter ou de refuser en dernier recours l'intégration d'un volontaire au sein de la communauté des développeurs Debian.
Voir « nouveaux mainteneurs ».

Debian Bug Tracking System Debian BTS Voir « BTS (*Bug Tracking System*) ».

Debian Developer's Reference Voir « référence du développeur Debian ».

Debian Free Software Guidelines (DFSG) Principes du logiciel libre selon Debian. Ce texte fondateur définit les conditions que doit respecter la licence d'un logiciel pour qu'il soit « libre » selon Debian et puisse donc être intégré dans la distribution. Voir page 7 pour le détail de ces conditions.

Debian Quality Assurance Debian QA Voir « système de suivi de paquets ».

Debian Weekly News (DWN) Journal électronique hebdomadaire présentant les nouvelles de Debian. Voir « Schulze, Martin ».

debian-installer Le plus récent programme d'installation de Debian. D'une structure modulaire, il est employé afin de créer des installateurs répondant à tous types de besoins.

debian-policy Paquet contenant la charte Debian. Toute demande de modification s'exprime sous la forme d'un rapport de bogue sur ce paquet.
Voir « charte Debian ».

`debian.net` Le domaine *debian.net* ne constitue pas une ressource officielle du projet Debian. Chaque développeur Debian a la possibilité d'employer ce nom de domaine pour l'usage de son choix. On y trouve des services officieux (parfois des sites personnels) hébergés sur une machine n'appartenant pas au projet et mis en place par des développeurs Debian, voire des prototypes attendant d'être migrés sur *debian.org*. Deux raisons peuvent expliquer cet état de fait : soit personne ne souhaite faire l'effort nécessaire à sa transformation en service officiel (hébergé dans le domaine *debian.org*), soit le service est trop controversé pour être officialisé.
Le Wiki (`wiki.debian.net`), site collaboratif où même de simples visiteurs peuvent faire des suggestions depuis un navigateur, relève probablement du premier cas.

debian.org Site officiel du projet Debian.
Voir « `debian.net` ».

debsums Outil intéressant du point de vue de la sécurité puisqu'il permet de trouver facilement quels fichiers installés ont été modifiés (suite par exemple à des interventions malignes). Mais il convient de nuancer fortement cette affirmation : d'abord, tous les paquets Debian ne fournissent pas les empreintes nécessaires au fonctionnement de ce programme (quand elles existent, elles se trouvent dans un fichier `md5sums`).

D'autre part, les fichiers `md5sums` sont stockés sur le disque dur : un intrus consciencieux modifiera ces fichiers pour leur faire refléter les nouvelles sommes de contrôle des fichiers sur lesquels il sera intervenu.

`debsums` n'est pas le seul détecteur de modifications. Le programme *AIDE* (paquet Debian *aide*), par exemple, détecte de manière fiable toute modification.

Voir le chapitre 5.

dépendance — Déclaration qui précise, dans le jargon des paquets Debian, qu'un autre paquet est nécessaire au bon fonctionnement du paquet concerné.

Elle indique que le paquet déclaré doit être décompacté et configuré avant que le paquet déclarant ne soit lui-même configuré.

Cette notion est détaillée au chapitre 5.

Depends — Champ d'en-tête d'un paquet déclarant une dépendance.

Voir « dépendance ».

Voir « *Pre-Depends* ».

déporter les logs — C'est une bonne idée que d'enregistrer les logs les plus importants sur une machine séparée (voire dédiée), car cela compliquera la tâche d'un éventuel intrus soucieux de supprimer les traces de son passage (sauf à compromettre également cet autre serveur). Par ailleurs, en cas de problème majeur (tel qu'un plantage noyau), disposer de logs sur une autre machine augmente les chances de retrouver le déroulement des événements.

développeur Debian — Participant au projet Debian qui a la charge de transformer un logiciel existant en paquet (les désignations « mainteneur Debian » ou « responsable de paquet Debian » existent aussi). Généralement, il n'intervient pas sur le code source du logiciel lui-même, contrairement à l'auteur amont. Bien souvent, la ligne de démarcation n'est cependant pas aussi nette : le mainteneur Debian écrit parfois un correctif qui profite à tous les utilisateurs du logiciel.

Voir « auteur amont ».

Voir « responsable de paquet ».

DFSG (*Debian Free Software Guidelines*) — Voir « *Debian Free Software Guidelines* ».

distribution, version — Debian définit plusieurs « versions » de distributions. Chacune rassemble des paquets présentant un niveau de maturité (stabilité) donné.

Voir « *release* ».

documentation — La documentation de chaque paquet est stockée dans `/usr/share/doc/<`*paquet*`>`. Ce répertoire contient souvent un fichier `README.Debian` décrivant les aménagements spécifiques à Debian réalisés par le mainteneur. Il est donc sage de lire ce fichier avant toute configuration.

Voir le chapitre 1.

`dpkg` — Programme qui permet de manipuler des fichiers `.deb`, notamment de les extraire, analyser, décompacter, etc.

`dselect` Programme standard pour sélectionner les paquets à installer, avant **`aptitude`**, doté d'un interface graphique associée à **`dpkg`**. Difficile d'emploi pour les débutants, il est donc déconseillé.

DWN (*Debian Weekly News*) Journal électronique hebdomadaire présentant les nouvelles de Debian. Voir « Schulze, Martin ».

Enhances Champ décrivant des dépendances non obligatoires mais « améliorantes ». Il décrit une suggestion mais se trouve dans le paquet suggéré et non dans celui qui profite de la suggestion. Il offre moyen d'ajouter une suggestion sans devoir modifier le paquet concerné. Ainsi, tous les *add-ons* (ajouts), *plugins* (greffons) et autres extensions d'un logiciel pourront ensuite prendre place dans la liste des suggestions liées au logiciel. Ce dernier champ — récemment créé — est encore largement ignoré par des programmes comme **`apt-get`** ou **`synaptic`**. L'objectif est cependant qu'une suggestion faite par le biais d'un champ *Enhances* apparaisse à l'utilisateur en complément des suggestions traditionnelles — réalisées avec le champ *Suggests*.

environnement *chrooté* Dans le cadre de la construction de paquets Debian, répertoire temporaire contenant un système minimal nécessaire à le reconstruction d'un paquet (en se basant sur les informations contenues dans le champ *Build-Depends*). Grâce à la commande **`chroot`**, ce répertoire sert ensuite de racine (/) lors du processus de recompilation.
Voir « **`pbuilder`** ».

etch Voir « noms de code ».

étiquettes Voir « système de suivi de bogues ».

experimental Distribution spéciale contenant des paquets Debian préliminaires susceptibles de souffrir de gros défauts. Elle est essentiellement employée pour distribuer des paquets de logiciels en développement (pré-versions, alpha, bêta, *release candidate*, etc.).

Free Software Foundation FSF Voir « FSF (*Free Software Foundation*) ».

freeze Période pendant laquelle l'évolution du contenu de la distribution *testing* est bloquée (« gel ») : plus aucune mise à jour automatique n'a lieu. Seuls les *Release Managers* sont alors habilités à y changer des paquets, selon leurs propres critères. L'objectif est d'éviter l'apparition de nouveaux bogues par l'introduction de nouvelles versions ; seules les mises à jour urgentes et bien examinées sont acceptées lorsqu'elles corrigent des bogues fâcheux.
Voir « *Release Manager* ».

FSF (*Free Software Foundation*) Association, responsable de nombreux programmes et projets de logiciels libres et à l'origine des licences les plus répandues : la GNU GPL et ses variantes.
Son projet GNU, démarré au début des années 1980, et qui visait à produire un système d'exploitation compatible Unix entièrement libre, est désormais une réalité avec les distributions GNU/Linux.
Voir « licences libres ».

ftpmasters Responsables de l'archive centrale contenant les paquets Debian envoyés par les mainteneurs. Ils vérifient les nouveaux paquets avant de les intégrer dans la distribution et gèrent les outils qui mettent ensuite à jour les paquets de manière automatique.

gel Voir « *freeze* ».

Genibel, Igor Auteur d'une interface web similaire au système de suivi de paquets mais articulé autour des mainteneurs (plutôt que des paquets eux-mêmes) qui donne à chacun d'eux un synoptique de l'état de tous les paquets Debian placés sous sa responsabilité.
▸ http://qa.debian.org/developer.php

gestionnaire de version Voir « *Release Manager* ».

GNU GPL L'une des licences de logiciel libre les plus répandues. Elle repose sur le principe du *copyleft*.
Utilisée et promue par la FSF (*Free Software Foundation*, ou fondation du logiciel libre), cette licence est la plus courante. Elle a pour particularité de s'appliquer à toute œuvre dérivée et redistribuée : un programme intégrant ou utilisant du code GPL ne peut être diffusé (par un tiers) que selon ses termes. Elle interdit donc toute récupération dans une application propriétaire. Ceci pose de gros problèmes pour le réemploi de code GPL dans des logiciels libres incompatibles avec cette licence. Ainsi, il est parfois impossible de lier une bibliothèque diffusée sous GPL à un programme placé sous une autre licence libre. En revanche, cette licence est très solide en droit américain : les juristes de la FSF ont participé à sa rédaction, et elle a souvent contraint des contrevenants à trouver un accord amiable avec la FSF afin d'éviter un procès.
▸ http://www.gnu.org/copyleft/gpl.html
Voir « licences libres ».

gravité d'un bogue Voir « sévérité d'un bogue ».

hamm Voir « noms de code ».

Ian Murdock Voir « Murdock, Ian ».

Igor Genibel Voir « Genibel, Igor ».

impression Voir « CUPS (*Common Unix Printing System*) ».

init.d Répertoire abritant les scripts d'arrêt, de démarrage et de redémarrage des services.
Voir « redémarrage des services ».

installateur Voir « **`debian-installer`** ».

`invoke-rc.d` Programme auquel les scripts de configuration doivent recourir pour appeler les scripts d'initialisation des services. Il n'exécutera que les commandes nécessaires (un service stoppé ne peut pas être redémarré, ni arrêté à nouveau, etc.). Attention, contrairement à l'usage, le suffixe .d est ici employé dans un nom de programme et non de répertoire.
Voir « redémarrage des services ».

Langasek, Steve
: Co-gestionnaire de version avec Colin Watson depuis août 2004.
Voir « *Release Manager* ».

leader
: Développeur Debian élu par ses pairs pour les diriger et les représenter durant une année. L'élection du *leader* est toujours une période d'intense discussion. Les points de vue de cet élu sont implicitement approuvés par la majorité des membres du projet Debian.
Voir « constitution ».

licences libres
: La GNU GPL, la licence BSD et la licence artistique respectent toutes trois les principes du logiciel libre définis par Debian. Elles sont pourtant très différentes.
Retrouvez le texte complet de ces licences dans /usr/share/common-licenses/ sur tout système Debian.
Voir « DFSG (*Debian Free Software Guidelines*) ».
Voir « GNU GPL ».
Voir « BSD, licence ».
Voir « artistique, licence ».

`linda`
: Logiciel de vérification automatique d'un paquet Debian.
Voir « **`lintian`** ».

`lintian`
: Logiciel de vérification automatique d'un paquet Debian. Il permet aux mainteneurs de paquets de débusquer les erreurs les plus flagrantes, notamment les manquements à la charte Debian.
Voir « **`linda`** ».

listmasters
: Responsables des listes de diffusion, ils en gèrent tous les aspects : création/suppression des listes, traitement des retours, administration du serveur de courrier, maintien des filtres *antispam*, etc.

log
: Fichier-journal consignant certains événements du système.
Voir « déporter les logs ».

main, archive
: Collection principale de paquets Debian, répondant tous aux principes du logiciel libre définis par Debian.

mainteneur
: Terme calqué sur l'anglais *maintainer* et que j'utilise parfois en lieu et place de la traduction officielle, « responsable de paquet ».
Voir « responsable de paquet ».

Martin Schulze
: Voir « Schulze, Martin ».

`mentors.debian.net`
: Site regroupant des paquets réalisés par des prétendants au statut de développeur Debian officiel ou par des volontaires souhaitant créer des paquets Debian sans passer par le processus d'intégration.
Voir « ressources non officielles ».

menus — L'organisation des menus Debian suit une structure précise, documentée dans le texte suivant :

▸ http://www.debian.org/doc/packaging-manuals/menu-policy/

Il est recommandé de choisir une section listée dans ce document pour remplir le champ `section` d'un fichier `menu`.

méta-paquet — Paquet réel, doté de fichiers `.deb`), dont le seul intérêt est d'exprimer des dépendances (son installation déploiera les paquets correspondants).

Voir « paquet virtuel ».

modifications, préserver — Voir « préserver la configuration existante ».

Murdock, Ian — Fondateur du projet Debian, il en fut le premier leader, de 1993 à 1996. Après avoir passé la main à Bruce Perens, il s'est fait plus discret. Il est ensuite revenu sur le devant de la scène du logiciel libre en créant la société Progeny, visant à commercialiser une distribution dérivée de Debian. Ce fut un échec commercial, au développement depuis abandonné. Mais les contributions de Progeny subsistent : citons *Progeny Graphical Installer (PGI)* (installateur graphique) ou *discover* (détection automatique du matériel). Plus récemment, Progeny, devenue une SSII spécialiste des logiciels libres, a adapté l'installateur automatique de Red Hat (anaconda) pour permettre son utilisation avec Debian. Son dernier projet, *Componentized Linux*, consiste en une nouvelle approche d'une distribution Linux à base de collections de paquets cohérents.

Voir « *Componentized Linux* ».

new maintainers — Voir « nouveaux mainteneurs ».

noms de code — Avant d'avoir un numéro, chaque version de Debian est connue par un nom de code : la tradition veut que ces noms proviennent des personnages de *Toy Story* — film d'animation produit par Pixar, l'employeur de Bruce Perens à l'époque où il était leader Debian.

Se sont ainsi succédé les noms de codes suivants : *rex* (version 1.1), *buzz* (1.2), *bo* (1.3), *hamm* (2.0), *slink* (2.1), *potato* (2.2), *woody* (3.0), *sarge* (3.1). *etch* est le nom de code pour la prochaine version tandis que *sid* restera éternellement associé à *unstable* ; dans le film, il s'agit de l'enfant des voisins, incorrigible brise-tout — gare à vous donc si vous approchez *unstable* de trop près ! Par ailleurs, *sid* est l'acronyme de *Still In Development* (encore et toujours en cours de développement).

non-free, archive — Collection spéciale de paquets Debian, qui contient des logiciels ne répondant pas (totalement) aux principes du logiciel libre selon Debian 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 ces logiciels — mais Debian recommande toujours d'employer de préférence un logiciel libre.

nouveaux mainteneurs — Candidats qui souhaitent rejoindre les rangs des développeurs Debian. Ils doivent satisfaire aux conditions d'une procédure d'acceptation de plus en plus exigeante dont l'objectif est de garantir leur capacité à bien s'intégrer et à fournir des paquets Debian conformes.

La procédure se conclut par la revue de la candidature par un très petit nombre de personnes, les « Responsables des comptes Debian » (ou *DAM — Debian Account Managers*). Ceux-ci sont donc particulièrement exposés aux critiques, puisqu'ils acceptent ou refusent en dernier recours l'intégration d'un volontaire au sein de la communauté des développeurs Debian. Dans la pratique, il s'agit parfois de retarder l'acceptation d'une personne afin qu'elle prenne le temps de mieux explorer le fonctionnement du projet. On peut en effet contribuer à Debian avant d'y être accepté comme développeur officiel grâce à un mécanisme de parrainage.

Voir « DAM (*Debian Account Manager*) ».

paquet binaire
: Paquet Debian contenant des fichiers fonctionnels directement utilisables (programmes, documentation) par les utilisateurs de la distribution Debian.

Voir « paquet Debian ».

paquet Debian
: Archive qui renferme un ensemble de fichiers permettant d'installer un logiciel. Dans le cas général, il s'agit d'un fichier d'extension `.deb`, qu'on manipule avec le programme **`dpkg`**. Un paquet sera qualifié de *binaire* s'il contient des fichiers fonctionnels directement utilisables (programmes, documentation) ou de *source* s'il abrite les codes sources du logiciel et les instructions nécessaires à la fabrication du paquet binaire.

Voir « conflit ».

Voir « **`debconf`** ».

Voir « mainteneur ».

Voir « popularité, paquet ».

Voir « système de suivi de paquets ».

paquet source
: Paquet Debian qui abrite les codes sources du logiciel et les instructions nécessaires à la fabrication du paquet binaire.

Voir « paquet Debian ».

Voir « source de paquets ».

paquet virtuel
: Paquet qui n'existe pas physiquement mais fournit un moyen d'identifier des paquets réels sur la base d'un critère logique commun (service fourni, compatibilité avec un programme standard ou un paquet pré-existant, etc.).

Pour que les paquets virtuels soient utiles, il faut que tout le monde s'entende sur leur nom. C'est pourquoi ils sont standardisés par la charte Debian. La liste comprend entre autres *mail-transport-agent* pour les serveurs de courrier électronique, *c-compiler* pour les compilateurs C, *www-browser* pour les navigateurs web, *httpd* pour les serveurs web, *ftp-server* pour les serveurs FTP, *x-terminal-emulator* pour les émulateurs de terminal en mode graphique (**`xterm`**) et *x-window-manager* pour les gestionnaires de fenêtres.

La liste complète se trouve sur le Web :

▸ http://www.debian.org/doc/packaging-manuals/virtual-package-names-list.txt

Voir « méta-paquet ».

`pbuilder`
: Programme (du paquet éponyme) qui permet de recompiler un paquet Debian dans un environnement *chrooté* : il crée un répertoire temporaire contenant un système minimal nécessaire à le reconstruction du paquet (en se basant sur les informations contenues

dans le champ *Build-Depends*). Grâce à la commande **`chroot`**, ce répertoire sert ensuite de racine (/) lors du processus de recompilation.

Cette technique permet de compiler le paquet dans un environnement non dégradé (notamment par les manipulations des utilisateurs), de détecter rapidement les manques éventuels dans les dépendances de compilation (qui échouera si un élément essentiel n'est pas documenté) et de compiler un paquet pour une version de Debian différente de celle employée par le système (la machine peut utiliser *stable* pour le fonctionnement quotidien et **`pbuilder`** peut employer *unstable* pour la recompilation).

Perens, Bruce — Deuxième leader du projet Debian, juste après Ian Murdock. Il fut très controversé pour ses méthodes dynamiques et assez dirigistes. Il n'en reste pas moins un contributeur important, à qui Debian doit notamment la rédaction des fameux « principes du logiciel libre selon Debian » (ou DFSG pour *Debian Free Software Guidelines*), idée originelle d'Ean Schuessler. Par la suite, Bruce en dérivera la célèbre « définition de l'Open Source » en y gommant toutes les références à Debian.

▸ http://www.opensource.org

Son départ du projet fut quelque peu mouvementé mais Bruce est resté assez fortement attaché à Debian puisqu'il continue de promouvoir cette distribution dans les sphères politiques et économiques. Il intervient encore régulièrement sur les listes de diffusion pour donner son avis et présenter ses dernières initiatives en faveur de Debian. Dernier point anecdotique, c'est à lui que l'on doit l'inspiration des « noms de code » des différentes versions de Debian

Voir « noms de code ».

PGI (*Progeny Graphical Installer*) — Voir « Murdock, Ian ».

Pixar — Voir « noms de code ».

popularité, paquet — En installant le paquet Debian *popularity-contest* vous pouvez participer à un sondage (automatique) grâce auquel le projet Debian recense les paquets les plus populaires. Ces informations fournies anonymement et collectées hebdomadairement permettent entre autres au projet de placer les paquets les plus employés sur les premiers cédéroms ou de vérifier l'impact d'une suppression de paquet.

Les statistiques ainsi collectées sont publiées quotidiennement. Elles peuvent éventuellement vous aider lors d'un choix de logiciel. En prenant le plus populaire vous ferez probablement un meilleur choix.

▸ http://popcon.debian.org

popularity-contest — Voir « popularité, paquet ».

Postfix — Serveur de messagerie électronique sur protocole SMTP.

Voir « Venema, Wietse ».

postinst — Voir « scripts de configuration ».

postrm — Voir « scripts de configuration ».

potato Voir « noms de code ».

pré-dépendance Voir « *Pre-Depends* ».

Pre-Depends Déclarations du champ « *Pre-Depends* » de l'en-tête du paquet, qui complètent les dépendances normales ; leur syntaxe est identique. Une pré-dépendance stipule que le paquet concerné doit être décompacté et configuré avant même d'exécuter le script de pré-installation du paquet la déclarant, c'est-à-dire avant son installation proprement dite.

Une pré-dépendance est très contraignante pour **apt**, qui doit ordonnancer la liste des paquets à installer. Elles ne sont donc utilisées qu'en cas de stricte nécessité. Il est même recommandé de consulter l'avis des (autres) développeurs, via debian-devel@lists.debian.org, avant d'ajouter une pré-dépendance. En règle générale on trouve une autre solution.

Voir « dépendance ».

preinst Voir « scripts de configuration ».

prerm Voir « scripts de configuration ».

préserver la configuration existante La charte Debian demandant expressément de tout faire pour préserver au maximum les changements manuels apportés aux fichiers de configuration, de plus en plus de scripts modifiant ces derniers prennent des précautions. Le principe général est simple : le script n'effectue des modifications que s'il connaît l'état du fichier de configuration, vérification effectuée par comparaison de la somme de contrôle du fichier avec celle de la version automatiquement produite. Si elles correspondent, le script s'autorise à modifier le fichier de configuration. Dans le cas contraire, il considère qu'un tiers le modifia et demande quelle action il doit effectuer (installer le nouveau fichier, conserver l'ancien, ou tenter d'intégrer les nouvelles modifications au fichier existant). Ce principe de précaution fut longtemps propre à Debian, mais les autres distributions l'adoptent peu à peu.

Le programme **ucf** (du paquet Debian éponyme) offre des facilités pour gérer cela.

Progeny Graphical Installer (PGI) Voir « Murdock, Ian ».

Provides Champ d'en-tête indiquant qu'un paquet binaire est une instance possible du paquet virtuel cité. On dit qu'il « fournit » le paquet virtuel.

pseudo-paquet Le système de suivi de bogues, conçu pour rassembler les rapports de bogues relatifs à un paquet Debian donné, sembla très pratique pour gérer d'autres cas de figure : liste de problèmes à résoudre ou de tâches à mener indépendamment de tout lien avec un paquet Debian. Les « pseudo-paquets » permettent ainsi à certaines équipes d'utiliser le système de suivi de bogues sans y associer de véritable paquet. Tout le monde peut ainsi leur signaler des éléments à traiter. Le *Bug Tracking System* dispose ainsi d'une entrée *ftp.debian.org* pour signaler les problèmes de l'archive de paquets ou simplement y demander la suppression d'un paquet. Le pseudo-paquet *www.debian.org* correspond ainsi aux soucis liés au site web de Debian et *lists.debian.org* à ceux des listes de diffusion.

purge — Action du programme **dpkg** supprimant tous les fichiers installés correspondants à un paquet Debian donné. Lorsqu'un paquet Debian est désinstallé, les fichiers de configuration sont conservés afin de faciliter une éventuelle réinstallation. De même, les données gérées par un démon (par exemple le contenu de l'annuaire d'un serveur LDAP, ou le contenu de la base de données pour un serveur SQL) sont habituellement conservées.

Pour supprimer toute donnée associée au paquet, il faut procéder à sa « purge » avec la commande **dpkg -P *paquet*** ou **apt-get remove --purge *paquet***.

Quality Assurance QA — Voir « système de suivi de paquets ».

rapport de bogue — Courrier électronique envoyé par un utilisateur pour révéler un problème détecté dans un paquet Debian.

Voir « système de suivi de bogues ».

Recommends — Champ d'en-tête décrivant des dépendances non obligatoires mais « recommandées ». Ce sont les plus importantes, et elles améliorent considérablement les fonctionnalités offertes par le paquet sans pour autant être indispensables à son fonctionnement.

Il faut systématiquement installer les paquets « recommandés » sauf si vous savez précisément pourquoi vous n'en avez pas besoin.

recompiler un paquet — Opération consistant à construire de nouveau un paquet à partir de ses sources.

Voir « **buildd** ».

Voir « **pbuilder** ».

redémarrage des services — Les scripts de configuration des paquets Debian redémarrent parfois certains services pour assurer leur disponibilité ou leur faire prendre en compte certaines nouvelles options. La commande de redémarrage d'un service **/etc/init.d/*service* restart** ne prend pas en compte le niveau d'exécution, suppose (à tort) que le service est actuellement employé, et peut donc le redémarrer à mauvais escient.

référence du développeur Debian — Document destiné aux développeurs Debian présentant un certain nombre de procédures et d'outils existants. Ce guide regroupe des recommandations importantes qui facilitent la collaboration entre les mainteneurs.

▸ http://www.debian.org/doc/developers-reference/

Voir « charte Debian ».

release — Le terme « *release* » désigne chez Debian une version particulière d'une distribution (ex : « *the unstable release* » signifie « la version instable »). Il désigne aussi l'annonce publique de toute nouvelle version stable.

Release Manager — *Release Manager* (gestionnaire de version) est un titre important, associé à de lourdes responsabilités. Son porteur doit en effet gérer la sortie de la nouvelle version stable de Debian et définir le processus d'évolution de *testing* tant qu'elle ne répond pas aux critères de qualité de *stable*. Il définit également un calendrier prévisionnel (jamais respecté).

Colin Watson et Steve Langasek partagent cette responsabilité depuis août 2004. Anthony Towns l'avait auparavant assumée plusieurs années, après avoir programmé les scripts de gestion de *testing*.

Voir « *freeze* ».

Voir « *Stable Release Manager* ».

Replaces Champ d'en-tête indiquant que le paquet contient des fichiers également présents dans un autre paquet, mais qu'il a légitimement le droit de les remplacer.

responsable de paquet Volontaire ayant choisi d'intégrer un paquet à Debian et de l'y faire évoluer. Le terme anglais correspondant est *maintainer* ; c'est pourquoi j'emploie souvent le mot « mainteneur », plus concis et tout aussi explicite. On parle aussi de « développeur Debian ».

Voir « mainteneur ».

ressources non officielles Collections de paquets Debian n'appartenant pas officiellement au projet. 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 pré-versions de leur paquet en ligne. Un site web fut mis en place pour les trouver facilement. 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

Installer un paquet revient à donner les droits *root* à son concepteur, car il décide du contenu de scripts d'initialisation qui sont exécutés sous cette identité. Les paquets officiels Debian sont réalisés par des volontaires cooptés et examinés capables de sceller leurs paquets pour offrir à tous un moyen d'en vérifier l'origine et l'intégrité.

Défiez-vous a priori d'un paquet dont l'origine est incertaine, ce qui est le cas s'il est publié hors des serveurs officiels du projet Debian. Évaluez le degré de confiance que vous accordez au concepteur et vérifiez l'intégrité du paquet.

▸ http://mentors.debian.net

Voir « `mentors.debian.net` ».

rex Voir « noms de code ».

sarge Voir « noms de code ».

Schulze, Martin Mainteneur amont de **`sysklogd`** et **`klogd`**. Ce développeur Debian de la première heure a de nombreuses responsabilités au sein du projet. Outre ses fonctions de mainteneur de paquets, il est membre des équipes *debian-admin* et *sécurité*, a longtemps participé au processus d'acceptation des nouveaux mainteneurs, et assure la fonction de rédacteur en chef du journal électronique hebdomadaire *Debian Weekly News*. Il gère également les mises à jour périodiques de la version stable de Debian.

Voir « *Stable Release Manager* ».

scripts de configuration
: Scripts exécutés automatiquement à différentes étapes de l'installation ou de la suppression d'un paquet Debian. Voir le chapitre 5.

service, redémarrage
: Voir « redémarrage des services ».

sévérité d'un bogue
: Importance relative d'un bogue. La « sévérité » (gravité ; *severity* en anglais) d'un bogue décrit de manière formelle la gravité du problème signalé. Tous n'ont en effet pas la même importance : une faute de frappe dans un manuel n'a rien de comparable à une faille de sécurité dans un logiciel serveur.
Debian utilise une échelle étendue de sévérités permettant d'exprimer assez finement la gravité d'un bogue. Elle définit par ailleurs très précisément chacun de ces niveaux afin de faciliter le choix de l'un ou l'autre.
▸ http://www.debian.org/Bugs/Developer.fr.html#severities

sid
: Voir « noms de code ».

Simple Mail Transfer Protocol (SMTP)
: Protocole de routage de courriers électroniques le plus répandu.
Voir « Venema, Wietse ».

slink
: Voir « noms de code ».

Software in the Public Interest (SPI)
: Voir « SPI (*Software in the Public Interest*) ».

source de paquets
: Le terme *source* est source d'ambiguïté. Il ne faut pas confondre un paquet source — paquet contenant le code source d'un programme — et une source de paquets — emplacement (site web, serveur FTP, cédérom, répertoire local, etc.) contenant des paquets.

sous-projet
: Regroupement de volontaires intéressés par l'adaptation de Debian à des besoins spécifiques (à chaque public sa Debian : éducation, médecine, création multimédia, etc.). Au-delà de la sélection d'un sous-ensemble de logiciels dédiés à un usage particulier, cela suppose d'améliorer les paquets existants, de mettre en paquet les logiciels manquants, d'adapter l'installateur, de fournir une documentation spécifique, etc.

SPI (*Software in the Public Interest*)
: Association dont Debian, qui ne possède aucun bien en son nom propre, n'est qu'un projet. Elle en gère les aspects matériels et financiers (dons, achat de matériel, etc.). Bien qu'initialement créée pour Debian, cette association coiffe maintenant d'autres projets du monde du logiciel libre.
▸ http://www.spi-inc.org

stable
: Distribution officielle de Debian contenant les versions stables des paquets Debian. La longue période de test qui permet d'aboutir à ce résultat implique malheureusement que la version de certains des logiciels ainsi distribués est ancienne.

Stable Release Manager
: Le gestionnaire de version stable complète le gestionnaire de version. Il gère et sélectionne les mises à jour de la version stable de Debian. Il y inclut systématiquement les correctifs de sécurité et examine au cas par cas toutes les autres propositions

d'inclusion faites par des développeurs Debian soucieux de mettre à jour un de leurs paquets dans la version stable. Cette personne est actuellement Martin Schulze.
Voir « *Release Manager* ».

Steve Langasek — Co-gestionnaire de version avec Colin Watson depuis août 2004.
Voir « *Release Manager* ».

Suggests — Champ d'en-tête décrivant des dépendances non obligatoires mais « suggérées ». Secondaires, elles indiquent que certains paquets peuvent se compléter et augmenter leur utilité respective — mais il est parfaitement raisonnable d'installer l'un sans les autres.
Il est inutile d'installer les paquets « suggérés » sauf si vous savez pourquoi vous en aurez besoin.

suivi de bogues — Voir « système de suivi de bogues ».

suivi de paquets — Voir « système de suivi de paquets ».

suivi de paquets — Voir « système de suivi de paquets ».

suppression — Voir « purge ».

`synaptic` — Interface graphique à APT permettant d'effectuer toutes les opérations courantes sur le système de paquetage (installation, mise à jour, suppression, purge, etc.).

système de suivi de bogues — Ensemble des logiciels qui permettent la gestion et le traitement des bogues des paquets Debian. Son interface web, partie émergée, permet de consulter tous les bogues répertoriés, et propose d'afficher une liste (triée) de bogues sélectionnés sur de nombreux critères : paquet concerné, sévérité, statut, adresse du rapporteur, adresse du mainteneur concerné, étiquette ou *tag*, etc.). Il est possible de consulter l'historique complet et toutes les discussions se rapportant à chacun des bogues.
Sous la surface, Debian BTS communique par courrier électronique : toutes les informations qu'il stocke proviennent de messages émis par les différents acteurs concernés. Voir l'encadré page 13 pour plus de détails techniques.
Debian BTS offre bien d'autres fonctionnalités (notamment les *tags*, ou étiquettes) ; découvrez-les dans sa documentation en ligne :
▸ http://www.debian.org/Bugs/index.fr.html

système de suivi de paquets — C'est l'une de mes réalisations. L'idée maîtresse est de rassembler sur une seule page les d'informations relatives à chaque paquet source. On peut ainsi apprécier rapidement l'état du logiciel, identifier les tâches à réaliser, et proposer son aide. C'est pourquoi cette page réunit en vrac les statistiques des bogues, les versions disponibles dans chaque distribution, la progression du paquet dans la distribution « *testing* », l'état des traductions des descriptions et des « *templates debconf* », l'éventuelle disponibilité d'une nouvelle version amont, des avertissements en cas de non conformité à la dernière version de la charte Debian, des renseignements sur le mainteneur, et toute autre information que celui-ci aura souhaité y intégrer.
▸ http://packages.qa.debian.org

Igor Genibel a créé une interface web similaire, développée autour des mainteneurs plutôt que des paquets eux-mêmes. Elle donne à chaque développeur un synoptique de l'état de tous les paquets Debian placés sous sa responsabilité.
▸ http://qa.debian.org/developer.php
Ces deux sites web constituent des outils pour *Debian QA* (« *Quality Assurance* »), le groupe en charge de l'assurance qualité au sein de Debian.

tags Étiquettes utilisées dans le suivi de bogues chez Debian.
Voir « système de suivi de bogues ».

templates Fichier décrivant les interactions avec l'utilisateur lors de la configuration d'un paquet.
Voir « **debconf** ».

testing Distribution intermédiaire entre *stable* et *unstable*. Elle consiste en une sélection automatique des paquets de *unstable* selon des critères tentant de garantir une certaine qualité.
Voir la section sur le cycle de vie des paquets, page 20.

Toy Story Film d'animation dont les noms des personnages furent employés pour baptiser les versions successives de Debian.
Voir « noms de code ».

ucf Voir « préserver la configuration existante ».

unstable Distribution où les paquets fraîchement préparés par les mainteneurs sont intégrés. Ils y subissent une campagne de tests et plusieurs mises à jour jusqu'à ce que leur qualité permette de les migrer vers *testing*.
Voir « *testing* ».

update-alternatives Script modifiant le lien établissant, sur une machine donnée, une correspondance entre un service et le logiciel l'assurant.
Voir « choix (*alternatives*) ».

Venema, Wietse Personnalité du monde du logiciel libre dont les compétences en matière de sécurité font un programmeur réputé. Il développa le programme **tcpd** et est également l'auteur principal de Postfix, serveur de messagerie électronique (SMTP — *Simple Mail Transfer Protocol*, ou protocole simple de courrier électronique) modulaire conçu pour être plus sûr et plus fiable que **sendmail**.

version Ensemble de paquets ayant tous atteint un niveau de maturité donné, regroupés dans un tout cohérent (surtout s'il est qualifié de « stable » ou de « *testing* »— les systèmes « *unstable* », comme leur nom l'indique, ne sont pas toujours très cohérents).
Voir « *Release Manager* ».

virtuel, paquet Voir « paquet virtuel ».

Watson, Colin Co-gestionnaire de version avec Steve Langasek depuis août 2004.
Voir « *Release Manager* ».

Wietse Venema — Voir « Venema, Wietse ».

`wiki.debian.net` — Voir « `debian.net` ».

wishlist — Niveau de sévérité d'un rapport de bogue exprimant un souhait plutôt qu'un problème. Voir « charte Debian ».

woody — Voir « noms de code ».

Index

Symboles

.d 87
.desktop 227
.htaccess 196
/boot/grub/menu.lst 124
/etc/X11/XF86Config-4 223
/etc/apt/apt.conf.d 86
/etc/apt/preferences 87
/etc/apt/sources.lists 82
/etc/bind/named.conf 178
/etc/default/nfs-common 200
/etc/default/nfs-kernel-server 200
/etc/default/ntpdate 126
/etc/exports 201
/etc/fstab 127
/etc/group 118
/etc/hosts 115
/etc/init.d/rcS 138
/etc/init.d/rcS.d 138
/etc/menu-methods 226
/etc/pam.d/common-account 214
/etc/pam.d/common-auth 214
/etc/pam.d/common-password 214
/etc/passwd 116
/etc/samba/smbpasswd 204
/etc/shadow 117
/etc/sudoers 127
/etc/timezone 125
/proc 114
/sys 114
/usr/lib/menu 226
/usr/share/doc 11
/usr/share/zoneinfo 125
/var/lib/dpkg 68
~ 120
>GForge 16
1000baseT 112
100baseT 112
10baseT 112
6bone 175

A

A, enregistrement DNS 176
AAAA, enregistrement DNS 177
ACPI 158
acpid 158
activité, historique 155
activité, surveillance 154
adduser 118
administration, interfaces 144
adresse IP 112
    privée 162
ADSL, modem 113
Advanced Configuration and Power Interface 158
Advanced Package Tool 82
Advanced Power Management 157
Afterstep 226
Agnula 15
AH, protocole 169
aide (paquet Debian) 72
ajout d'un utilisateur dans un groupe 119
alias 186
    domaine virtuel 186
alien 78
alioth 16
Allow from, directive Apache 197
AllowOverride, directive Apache 196
alternative 226
am-utils 128
amanda 155
amavisd-new 192
amd 128
amd64 40
amont, auteur 6
amorçable, cédérom 260
amorçage, chargeur de 46, 121
anacron 152
analog 106
analyseur de logs web 198
Anjuta 231
annuaire LDAP 210
antivirus 192
apache 194

Apache, directives 196, 197
APM 157
apmd 157
apropos 100
APT 82
    affichage des en-têtes 90
    configuration 86
    configuration initiale 56
    interfaces 91
    pinning 87
    préférences 87
    recherche de paquet 90
apt-cache 90
apt-cdrom 83
apt-check-sigs 92
apt-ftparchive 247
apt-get 84
apt-get autoclean 96
apt-get dist-upgrade 86
apt-get install 85
apt-get remove 85
apt-get update 84
apt-get upgrade 86
apt-get.org 84
apt.conf.d 86
aptitude 91
aptitude 58
ar 62
arch 18
architecture 5, 40
archive de paquets 246
artistique, licence 8
ASCII 110
association 4
assurance qualité 17
at 151
atd 149
ATI 223
atq 151
atrm 151
attribution des noms 114
auteur amont 6
authentification web 197
auto-monteur 128
autobuilder 21
autofs 128
automount 128
awk 226
AWStats 198
awtats 106
azerty 111

B

bande, sauvegarde 156
base de données
    des développeurs 10
    des groupes 116
    des utilisateurs 116
base-config 56
bash 119
Basic Input/Output System 44
BGP 174
bgpd 174
bind 177
bind9 177
BIOS 44
Blackbox 226
bloc, mode 119
bloquer un compte 118
bo 9
Bochs 237
bogue
    gravité 13
    rapport de 14, 106
    sévérité 13
    signaler un 14
boîte aux lettres, domaine virtuel 187
boot-floppies 5
bootloader 46, 55, 121
branchement à chaud 157
broadcast 112
Bruce Perens 9
BSD, licence 8
BTS 13
Bug Tracking System 13
bugs.debian.org 13
build daemon 21
buildd 21
bureau graphique 227
    déporté 142
bureautique, suite 235
buzz 9
bzip2 82

C

c++ 226
câble croisé 114
cache, proxy 57
cacti 155
caractère, mode 119
carte graphique 222
cc 226
cédérom
  amorçable 260
  businesscard 45
  d'installation 45
  netinst 45
  sauvegarde sur 156
chage 117
chaîne 163
changelog.Debian.gz 102
chargeur
  d'amorçage 46, 55, 121
  de démarrage 46
charte Debian 10
chastity-list 210
chaud, branchement 157
chfn 117
chgrp 144
chipset vidéo 223
chmod 144
choix 226
  de la langue 46
  du pays 47
chown 144
chsh 117
CIFS 203
clamav 192
clavier
  disposition 48, 111, 224
clé USB 45
client
  Jabber 234
  NFS 202
CNAME, enregistrement DNS 176
code
  binaire 5
  source 5
codename 9
CodeWeavers 236
coldplug 157
comité technique 11
commandes planifiées 149
commandes, interpréteur de 119
Common Unix Printing System 121
common-account 214
common-auth 214
common-password 214
comparaison de versions 78
compilateur 5
compilation 5
  d'un noyau 129
complétion automatique 120
Compose, touche 111
compte
  administrateur 56, 126
  bloquer 118
  création 118
Concurrent Versions System 18
conffiles 70
config, script debconf 69
configuration
  d'un logiciel 104
  de l'impression 121
  du noyau 130
  du réseau 112
    DHCP 49
    statique 49
  du système de base 56
  fichiers 70
  initiale d'APT 56
Conflicts, champ d'en-tête 65
conflits 65
connecteur RJ45 112
connexion
  à distance 140
  à la demande 113
  par modem ADSL 113
  par modem RTC 113
console, configuration du clavier 111
console-data 111
console-tools 111
constitution 11
contrat social 6
contrib, section 83
control 63

control.tar.gz 67
contrôle du trafic 172
contrôleur de domaine 204
copie de sauvegarde 156
copyleft 9
copyright 103
correctif 14
courrier électronique
    filtrage 185
    filtrage sur l'expéditeur 189
    filtrage sur le contenu 191
    filtrage sur le destinataire 190
    logiciel 229
CPAN 66
création
    de compte 118
    de groupe 118
cron 149
crontab 150
CrossOver Office 236
crypt 117
csh 120
CUPS 121
cupsys 121
    administration 121
CVS 18
cycle de vie 20

D

DAM 12
Darik Horn 170
DATA 190
DCF-77 126
dch 249
debc 249
debconf 69, 145
debhelper 250
debi 249
Debian Account Manager 12
Debian Free Software Guidelines 7
Debian policy 10
debian-admin 16
debian-cd 5
Debian-Edu 15
debian-installer 5, 44
debian-multimedia 15
debian-user-french 106
debian.net 45
debsums 71
debuild 249
démarrage
    chargeur de 46
    du système 138
démon 105
DeMuDi 15
dénis de service 181
Deny from, directive Apache 197
dépendance 64
Depends, champ d'en-tête 64
déporté, bureau graphique 142
dérivées, distributions 259
Destination NAT 162
détection d'intrusion 181
développeurs
    base de données 10
    Debian 10
devscripts 249
DFSG 7
dh-make 250
DHCP 112, 179
diald 113
diff 14
diff.gz, fichier 72
directives Apache 196, 197
DirectoryIndex, directive Apache 196
discover 222
discussion enflammée 12
disposition du clavier 48, 111, 224
disque dur, noms 121
distance, connexion 140
distribution Linux
    commerciale V, 31
    communautaire 31
    définition V
    rôle 19
distributions dérivées 259
dmesg 122
DNAT 162
DNS 115, 176
    enregistrement 177
    mise à jour automatique 180
    zone 176
DNSSEC 177

doc-linux-fr-html 104
doc-linux-html 104
documentation 100, 102
    emplacement 11
Domain Name Service 115
domaine
    nom de 115
    virtuel 186
domaine NT 204
dpkg 74
dpkg, fonctionnement interne 69
dpkg-reconfigure 145
dpkg-source 73
dput 250
droits 142
    d'auteurs 9
    masque 144
DSC, fichier 72
dselect 58
dsl-provider 114
dump 156
dupload 250
dur, lien 156
Dynamic Host Configuration Protocol 179

E

échange, partition de 53
écran, gestionnaire de 142
écriture, droit 143
edquota 152
eGroupware 232
EHLO 189
emplacement de la documentation 11
empreinte 71
émulation Windows 236
encodage 110
énergie, gestion 157
Enhances, champ d'en-tête 65
enregistrement DNS 177
environment 111
environnement
    hétérogène 36
    variable de 120
Epiphany 230
errants, profils 206
ESP, protocole 169
etch 9
été, heure 125
eth0 112
Ethernet 112
Evolution 229
evolution-exchange 230
Excel, Microsoft 235
ExecCGI, directive Apache 196
exécution
    droit 143
    niveau 139
exemples, emplacement 105
experimental 20, 83, 88
Explanation 89
exploration d'une machine Debian 39
exports 201

F

fichier
    de logs 105, 146
    de logs, rotation 125
    spécial 119
fichiers
    de configuration 70
    serveur de 199
    système de 52
fil, sans, réseau 158
file d'attente de paquets IP 172
filtrage de courrier électronique 185
filtre de paquets 163
Firefox, Mozilla 230
flamewar 12
FollowSymlinks, directive Apache 196
fonctionnement interne 10
fork 141
francisation 110
Freebox 114
FreeDesktop.org 227
Freenet6 175
freeswan 168
freeze 23
fréquence de rafraîchissement 225
Freshmeat 103
fstab 127
ftpmaster 16
fuseau horaire 56, 125
fwbuilder 166

G

GAIM 232
gatekeeper 232
gconf 228
gconftool-2 228
gdm 142, 225
Gecko 230
GECOS 116
gel 23
General Public License 8
gestion de l'énergie 157
gestion de la configuration 18
gestionnaire
  d'écran 142, 225
  de fenêtres 225
getent 118
getty 140
GForge 234
gid 116
Glade 231
GNOME 227
GNOME Office 235
gnome-apt 91
gnome-desktop-environment 228
gnome-system-monitor 154
gnome-system-tools 145
GnomeMeeting 232
GNU 4
  General Public License 8
  Info 102
  is Not Unix 4
GNU/Linux 29
Gnumeric 235
Gossip 234
GPL 8
GPS 126
gq 212
graphique, bureau déporté 142
gravité 13
GRE, protocole 169
group 118
groupadd 118
groupdel 118
groupe 116
  ajout d'un utilisateur 119
  base de données 116
  changer de 118
  création 118
  de volumes 54
  propriétaire 142
  suppression 118
groupmod 118
groupware 231
GRUB 55, 123
grub-install 123
GTK+ 227
gzip 82

H

H323 232
hamm 9
HELO 189
heure d'été 125
horloge 56
  synchronisation 126
Horn, Darik 170
host 178
hostname 114
hosts 115
hôte virtuel 195
hotplug 157
HOWTO 103
htpasswd 197
HTTP
  sécurisé 194
  serveur 194
HTTPS 194

I

i18n 13
ia64 40
Ian Murdock 4
Icewm 226
ICMP 164
ICQ 232
id 118
IDS 181
IEEE1394 157
IKE 168
impression
  configuration 121
  réseau 208
in-addr.arpa 178

Includes, directive Apache 196
Indexes, directive Apache 196
inetd 148
info 102
info2www 102
init 113, 138
installateur 44
installation
    d'un noyau 133
    de paquets 74, 84
    netboot 45
interface
    d'administration 144
    graphique 222
    réseau 112
internationalisation 13
Internet Control Message Protocol 164
Internet Printing Protocol 121
Internet Relay Chat 232
Internet Software Consortium 177
interpréteur de commandes 100, 119
Intrusion Detection System 181
intrusion, détection de 181
inverse, zone 178
invoke-rc.d 140
IP
    adresse 112
    paquets, file d'attente 172
ip6tables 176
ipmasq 163
IPP 121
iproute 172
IPsec
    IPsec Key Exchange 168
IPsec 168
iptables 163, 165
iputils-ping 175
iputils-tracepath 175
IPv6 175
    pare-feu 176
IRC 232
ISC 177
ISO-8859-1 110
ISO-8859-15 110
Itanium 40

J
Jabber 232
    clients 234
jeu de caractère 110

K
KDE 227
KDevelop 231
kdm 142, 225
kernel voir noyau
kernel-img.conf 133
kernel-package 129
kernel-patch-mppe 170
klogd 146
KMail 230
Knoppix 260
KOffice 235
Konqueror 230
Kopete 234
kwm 225

L
l10n 13
LANG 111
langue 110
Latin 0 110
Latin 1 110
Latin 9 110
LDAP 210
    sécurisé 216
LDIF 211
LDP 104
leader
    élection 11
    rôle 11
lecture, droit 143
libapache-mod-ssl 194
libnss-ldap 213
libpam-ldap 214
Libranet 261
licence
    artistique 8
    BSD 8
    GPL 8
lien
    dur 156
    symbolique 125

LILO 122
limitation de trafic 173
linda 249
Lindows 262
Linspire 262
lintian 249
Linux 29
    distribution V
    noyau V
Linux Documentation Project 104
Linux Loader 122
linuxconf 145
lire 106
listes
    de diffusion 16, 106
listmaster 16
LiveCD 260
ln 125
locale-gen 110
locales 110
localisation 13
locate 128
lockd 200
logcheck 106, 153
Logical Volume Manager 54
logiciel
    configuration 104
    libre 7
login 116
logrotate 125
logs
    affichage 154
    fichier de 105
    fichiers, rotation 125
    répartition 146
    surveillance 153
    web, analyseur 198
lpd 121
lpq 121
lpr 121
LVM 54

M

MAIL FROM 189
main, section 83
maintenance
    paquet 10
mainteneur
    nouveau 12
make-kpkg 131
    clean 131
    kernel-doc 131
    kernel-headers 131
    kernel-image 131
    kernel-source 131
    modules-image 132
makefile 245
man 100
man2html 102
mandataire HTTP/FTP 209
manuel, pages de 100
Martin Schulze 146
masque
    de droits 144
    de sous-réseau 112
masquerading 162
Master Boot Record 121
Matrox 223
MBR 121
MD5 71
md5sums 70
mdetect 222
mémoire virtuelle 53
menu 226
menu 226
menu-methods 226
menu.lst 124
Mepis Linux 261
méritocratie 12
messagerie
    électronique 184
    instantanée 232
Messenger 232
Meta, touche 111
méta-distribution 4
méta-informations d'un paquet 63
méta-paquet 64
metacity 225
Microsoft
    Excel 235
    Point-to-Point Encryption 170
    Word 235
migration 28, 36

migrationtools 212
mini-dinstall 247
mise à jour
    automatique du système 94
    du système 86
mise en veille 157
mknod 119
mode
    bloc 119
    caractère 119
modem
    ADSL 113
    RTC 113
modification, droit 143
modlogan 106
modprobe 138
module-init-tools 138
modules
    du noyau 138
    externes au noyau 132
modules-config 194
modutils 138
mondo 156
montage, point de 53, 127
mot de passe 117
mount 127
Mozilla 231
    Firefox 230
    Thunderbird 230
mozilla-browser 231
MPPE 170
mrtg 155
MultiViews, directive Apache 196
Murdock, Ian 4
MX
    enregistrement DNS 177
    serveur 185

N

Name Service Switch 118
named.conf 178
nameserver 115
NAT 162
Nat Traversal 169
NAT-T 169
navigateur Web 230
netfilter 163
Netiquette 106
Netscape 231
Network
    Address Translation 162
    File System 199
    IDS 181
    Time Protocol 126
newgrp 118
NEWS.Debian.gz 11, 102
NFS 199
    client 202
    options 201
    sécurité 200
nfs-common 200
nfs-kernel-server 200
nfs-user-server 201
NIDS 181
niveau d'exécution 139
nmap 38
nmbd 203
nom
    attribution et résolution 114
    de code 9
    de domaine 115
    des disques durs 121
    résolution 115
nommé, tube 148
non-free 7
non-free, section 83
noyau
    compilation 129
    configuration 130
    installation 133
    modules externes 132
    patch 133
    sources 129
NS, enregistrement DNS 177
NSS 118
NTP 126
    serveur 126
ntp-refclock 126
ntp-simple 126
ntpdate 126
nVidia 223

O

Open Source 9
OpenLDAP 210
OpenOffice.org 235
OpenSSH 141
openswan 168
option
    POSIX 76
Options, directive Apache 196
Order, directive Apache 197
ordonnanceur de commandes 149
organisation interne 10
OSPF 174
ospf6d 174
ospfd 174

P

package tracking system 17
Packages.gz 82
pages de manuel 100
PAM 111
pam_env.so 111
PAP 113
paquet
    binaire VII, 62
    conflit 65
    Debian VII
        archive de 246
    dépaquetage 75
    dépendance 64
    inspection du contenu 76
    installation 74, 84
    IP 162, 163
        file d'attente 172
    liste des fichiers 76
    maintenance 10
    méta-informations 63
    popularité 229
    priorité 87
    purge 75
    recherche 90
    remplacement 67
    scellement 92
    signature 92
    source VII, 72
    source de 82
    statut 76
    suppression 75, 84
    téléchargement 96
    types 244
    vérification d'authenticité 92
    virtuel 66
pare-feu 163
pare-feu IPv6 176
partage Windows 203
partition
    d'échange 53
    étendue 122
    primaire 122
    secondaire 122
partitionnement 50
    assisté 51
    manuel 52
passerelle 112, 162
passwd 116, 117
patch 14
patch noyau 133
pc105 111
PCMCIA 157, 158
pcmcia-cs 158
Perens, Bruce 9
périphérique
    droit d'accès 119
    multi-disques 53
Perl 66
permissions 142
PHPGroupware 232
Pin 89
Pin-Priority 89
ping 164
plan directeur 28
planification de commandes 149
pmud 158
poff 113
point à point 113
point de montage 53, 127
Point-to-Point Tunneling Protocol 169
policy 10
pon 113
popularité des paquets 229
popularity-contest 229
port
    TCP 162

UDP 162
port forwarding 141, 162
portmapper 200
POSIX 76
Postfix 184
postinst 67
postrm 67
potato 9
Powerbook 158
PPP 113, 168
pppconfig 113
PPPOA 113
PPPOE 114
pppoeconf 114
PPTP 114, 169
pptp-linux 169
pré-dépendance 64
Pre-Depends, champ d'en-tête 64
preferences 87
preinst 67
prelude 181
prerm 67
principes du logiciel libre 7
printcap 121
prise en main d'un serveur Debian 39
privée, adresse IP 162
proc 114
procédure type 104
processeur 5
processus 138
procmail 185
profils errants 206
Progeny 4
propriétaire
groupe 142
utilisateur 142
protocole
AH 169
ESP 169
GRE 169
Provides, champ d'en-tête 65
proxy 57
proxy cache 57, 209
pseudo-paquet 16
PTR, enregistrement DNS 177
PTS 17
purge d'un paquet 69, 75

Q

QEMU 237
QoS 172
Qt 227
Designer 231
quagga 174
qualité
assurance 17
de service 172
quality of service 172
quota 119, 152

R

racoon 168
radvd 176
rafraîchissement, fréquence 225
RAID logiciel 53
rapport de bogue 14, 106
RBL 188
RCPT TO 190
rcS 138
rcS.d 138
RDP 237
read-edid 222
README.Debian 11, 102
réception, tampon 164
recherche de paquet 90
Recommends, champ d'en-tête 65
récupération d'une machine Debian 39
redémarrage des services 140
Red Hat Package Manager 78
règle de filtrage 163, 165
réinstallation 71
release 20
Release Manager 22
Remote Black List 188
Remote Desktop Protocol 237
Remote Procedure Call 200
Remote Shell 140
remplacement 67
Replaces, champ d'en-tête 67
reportbug 14
Request For Comments 63
réseau
adresse du 112

configuration 112
configuration DHCP 179
passerelle 162
privé virtuel 168
sans fil 158
résolution 225
de nom 115
générale 12
resolv.conf 115
responsable des comptes Debian 12
restauration 155
restriction d'accès web 197
rétroportage 240
rex 9
RFC 63
RIP 174
ripd 174
ripngd 174
RJ45, connecteur 112
root 126
root-tail 154
rotation des fichiers de logs 125
routage
avancé 172
dynamique 174
route 174
routeur 112, 162
RPC 200
rpc.mountd 200
rpc.statd 200
RPM 78
RSH 140
rsync 156
runlevel 139

S

Samba 36, 203
sans fil, réseau 158
sarge 9
sauvegarde 155
copie 156
sur bande 156
sur cédérom 156
Schulze, Martin 146
scp 141
script d'initialisation 139
secrétaire du projet 11
section
contrib 83
main 83
non-free 7, 83
Secured Shell 140
serveur
de fichiers 199, 203
de noms 176
HTTP 194
MX 185
NTP 126
SMTP 184
web 194
X 222
service
qualité 172
redémarrage 140
setgid, droit 143
setquota 152
setuid, droit 143
sévérité 13
sftp 141
sg 118
SHA1 71
shadow 117
shaper 173
shell 100, 119
sid 9
signaler un bogue 14
Simple Mail Transfer Protocol 184
Simple Network Management Protocol 155
SkoleLinux 15
slapd 211
slink 9
SMB 203
smbclient 207
smbd 203
smbfs 207
smbmount 207
smbpasswd 204
SMTP 184
SNAT 162
SNMP 155
snort 181
social, contrat 6
Software in the Public Interest 6

software suspend 157
somme de contrôle 71
source (paquet) VII, 72
source de paquets 82
Source NAT 162
Sourceforge 234
sources du noyau Linux 129
Sources.gz 82
sources.list 82
souris 224
sous-projet 4
sous-réseau 112
spécial, fichier 119
speedtouch 113
SPI 6
Squid 57, 209
squidGuard 209
SSH 140, 168
stable 20
Stable Release Manager 22
StarOffice 235
sticky bit 143
subversion 18
sudo 126
sudoers 127
Suggests, champ d'en-tête 65
suite bureautique 235
super-serveur 148
supervision 153
suppression d'un paquet 75, 84
suppression de groupe 118
surveillance
    de l'activité 154
    des logs 153
swap 53
SWAT 203
symbolique, lien 125
SymlinksIfOwnerMatch, directive Apache 196
synaptic 91
synchronisation horaire 126
sys 114
syslogd 105, 146
système
    de base 55
    de fichiers 52
    de fichiers réseau 199
    de suivi de bogues 13
    de suivi de paquets 17
systemimager 156

T

tampon de réception 164
tc 172
TCO 30
TCP, port 162
tcpd 149
telnet 140
telnet-ssl 140
telnetd-ssl 140
testing 20
textes fondateurs 6
Thunderbird, Mozilla 230
tilde 120
timezone 125
top 154
ToS 174
Total Cost of Ownership 30
touche
    Compose 111
    Meta 111
trafic
    contrôle 172
    limitation 173
travail collaboratif 231
tube nommé 148
Type of Service 174
types de paquets 244
TZ 125

U

Ubuntu Linux 260
ucf 146
UDP, port 162
uid 116
umask 144
Unicode 110
unstable 20
update-alternatives 226
update-menus 226
update-modules 138
update-rc.d 140
update-squidguard 210
updatedb 128

upstream 6
USB 157
uscan 249
UTF-8 110
utilisateur
    base de données 116
    propriétaire 142

V
variable d'environnement 120
veille, mise en 157
vendeur de cédéroms 45
Venema, Wietse 149
version, comparaison 78
VESA 222
vidéoconférence 232
Virtual Network Computing 142
Virtual Private Network 168
virtuel, domaine 186
visudo 127
vmlinuz 133
VMWare 237
VNC 142
vncserver 142
volumes
    groupe de 54
    logiques 54
    physiques 54
vote 12
VPN 168

W
warnquota 153
Web, navigateur 230
web, serveur 194
webalizer 106
webmin 145
whatis 101
Wietse Venema 149
wifi 158
Wiki 45
Winbind 203
window manager 225
WindowMaker 226
Windows Terminal Server 237
Windows, émulation 236
Wine 236
winesetup 237
Wins 204
wireless-tools 158
wondershaper 173
woody 9
Word, Microsoft 235
www-browser 226
www-data 194

X
x-window-manager 226
x-www-browser 226
x11vnc 142
Xandros Linux 261
xdelta 156
xdm 142, 225
XF86Config-4 223
Xfce 229
XFree86 222
Ximian Connector 230
xkb 111
xserver-xfree86 222
xvncviewer 142

Y
yaboot 124
ybin 124

Z
zebra 174
zone
    DNS 176
    inverse 178
zoneinfo 125
zsh 120